Raisonnement sur le contexte et les croyances pour l'interaction homme-robot

Grégoire Milliez

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :
l'Institut National Polytechnique de Toulouse (INP Toulouse)

Présentée et soutenue le *18/10/2016* **par :**

GRÉGOIRE MILLIEZ

Raisonnement sur le contexte et les croyances pour l'interaction homme-robot

---

JURY

PETER FORD DOMINEY	Directeur de recherche CNRS	Président du Jury
RAJA CHATILA	Directeur de recherche CNRS	Rapporteur du Jury
DOMINIQUE DUHAUT	Professeur des universités	Rapporteur du Jury
RACHID ALAMI	Directeur de recherche CNRS	Membre du Jury
DANIEL SIDOBRE	Maître de conférences	Membre du Jury

École doctorale et spécialité :
MITT : Domaine STIC : Intelligence Artificielle
Unité de Recherche :
Laboratoire d'analyse et d'architecture des systèmes
Directeur(s) de Thèse :
Rachid Alami
Rapporteurs :
Dominique DUHAUT et *Raja CHATILA*

« Les hommes demanderont de plus en plus aux machines de leur faire oublier les machines.»
—Philippe Sollers, *Logique de la fiction*

«L'aspect le plus triste de notre vie aujourd'hui est que la science acquière les connaissances plus vite que la société n'acquière la sagesse.»
—Isaac Asimov

Remerciements

La fin d'une thèse est également la fin d'un chapitre dans le livre d'une vie. Ce chapitre représente pour moi l'accomplissement de dix années d'études et il me revient ici de remercier les personnes qui m'ont aidé à l'écrire jusqu'au bout. Mes premiers remerciements vont à Rachid Alami, mon directeur de thèse, qui par ses conseils, m'a permis de produire un travail de qualité. J'ai également apprécié les discutions et les moments partagés.

Je remercie grandement Sébastien Druon qui m'a soutenu lors de mon premier stage en robotique. Par sa patience et sa pédagogie, il m'a aidé à prendre confiance en moi et m'a donné l'envie de faire une thèse.

Je remercie également ceux qui m'ont encourager à faire des études et m'ont toujours soutenu, Pierre et Anne mes parents ainsi que ma famille.

Je remercie tous ceux qui m'ont accompagné durant ce périple, mes amis et collègues, avec qui j'ai eu le plaisir de collaborer et de partager des moments privilégiés. Le travail bien que parfois difficile, fut plus supportable en votre compagnie. J'espère avoir la chance et l'honneur de travailler à nouveaux avec vous à l'avenir.

Enfin je remercie Pierre, Domitille et Nathalie pour leur patience lors de leur relecture.

Table des matières

Résumé

Les premiers robots sont apparus dans les usines, sous la forme d'automates programmables. Ces premières formes robotiques ont le plus souvent un nombre très limité de capteurs et se contentent de répéter une séquence de mouvements et d'actions. De nos jours, de plus en plus de robots ont à interagir ou coopérer avec l'homme, que se soit sur le lieu de travail avec les robots coéquipiers ou dans les foyers avec les robots d'assistance.

Mettre un robot dans un environnement humain soulève de nombreuses problématiques. En effet, pour évoluer dans le même environnement que l'homme et comprendre cet environnement, le robot doit être doté de capacités cognitives appropriées.

Au delà de la compréhension de l'environnement matériel, le robot doit être capable de raisonner sur ses partenaires humains afin de pouvoir collaborer avec eux ou les servir au mieux. Lorsque le robot interagit avec des humains, l'accomplissement de la tâche n'est pas un critère suffisant pour quantifier la qualité de l'interaction. En effet, l'homme étant un être social, il est important que le robot puisse avoir des mécanismes de raisonnement lui permettant d'estimer également l'état mental de l'homme pour améliorer sa compréhension et son efficacité, mais aussi pour exhiber des comportements sociaux afin de se faire accepter et d'assurer le confort de l'humain.

Dans ce manuscrit, nous présentons tout d'abord une infrastructure logicielle générique (indépendante de la plateforme robotique et des capteurs utilisés) qui permet de construire et maintenir une représentation de l'état du monde à l'aide de l'agrégation des données d'entrée et d'hypothèses sur l'environnement. Cette infrastructure est également en charge de l'évaluation de la situation. En utilisant l'état du monde qu'il maintient à jour, le système est capable de mettre en œuvre divers raisonnements spatio-temporels afin d'évaluer la situation de l'environnement et des agents (humains et robots) présents. Cela permet ainsi d'élaborer et de maintenir une représentation symbolique de l'état du monde et d'avoir une connaissance en permanence de la situation des agents. Dans un second temps, pour aller plus loin dans la compréhension de la situation des humains, nous expliquerons comment nous avons doté notre robot de la capacité connue en psychologie développementale et cognitive sous le nom de "théorie de l'esprit" concrétisée ici par des mécanismes permettant de raisonner en se mettant à la place de l'humain, c'est à dire d'être doté de "prise de perspective". Par la suite nous expliquerons comment l'évaluation de la situation permet d'établir un dialogue situé avec l'homme, et en quoi la capa-

cité de gérer explicitement des croyances divergentes permet d'améliorer la qualité de l'interaction et la compréhension de l'homme par le robot. Nous montrerons également comment la connaissance de la situation et la possibilité de raisonner en se mettant à la place de l'homme permet une reconnaissance d'intentions appropriée de celui-ci et comment nous avons pu grâce à cela doter notre robot de comportements proactifs pour venir en aide à l'homme . Pour finir, nous présenterons une étude présentant un système de maintien d'un modèle des connaissances de l'homme sur diverses tâches et qui permet une gestion adaptée de l'interaction lors de l'élaboration interactive et l'accomplissement d'un plan partagé.

Abstract

The first robots appeared in factories, in the form of programmable controllers. These first robotic forms usually had a very limited number of sensors and simply repeated a small set of sequences of motions and actions.

Nowadays, more and more robots have to interact or cooperate with humans, whether at the workplace with teammate robots or at home with assistance robots.

Introducing a robot in a human environment raises many challenges. Indeed, to evolve in the same environment as humans, and to understand this environment, the robot must be equipped with appropriate cognitive abilities.

Beyond understanding the physical environment, the robot must be able to reason about human partners in order to work with them or serve them best. When the robot interacts with humans, the fulfillment of the task is not a sufficient criterion to quantify the quality of the interaction. Indeed, as the human is a social being, it is important that the robot can have reasoning mechanisms allowing it to assess the mental state of the human to improve his understanding and efficiency, but also to exhibit social behaviors in order to be accepted and to ensure the comfort of the human.

In this manuscript, we first present a generic framework (independent of the robotic platform and sensors used) to build and maintain a representation of the state of the world by using the aggregation of data entry and hypotheses on the environment. This infrastructure is also in charge of assessing the situation. Using the state of the world it maintains, the system is able to utilize various spatio-temporal reasoning to assess the situation of the environment and the situation of the present agents (humans and robots). This allows the creation and maintenance of a symbolic representation of the state of the world and to keep awareness of each agent status.

Second, to go further in understanding the situation of the humans, we will explain how we designed our robot with the capacity known in developmental and cognitive psychology as "theory of mind", embodied here by mechanisms allowing the system to reason by putting itself in the human situation, that is to be equipped with "perspective-taking" ability. Later we will explain how the assessment of the situation enables a situated dialogue with the human, and how the ability to explicitly manage conflicting beliefs can improve the quality of interaction and understanding of the human by the robot. We will also show how knowledge of the situation and the perspective taking ability allows proper recognition of human intentions and how we enhanced the robot with proactive behaviors to help the

human. Finally, we present a study where a system maintains a human model of knowledge on various tasks to improve the management of the interaction during the interactive development and fulfillment of a shared plan.

Chapitre 1

Introduction

Sommaire

1.1 Contexte général

De nombreux fantasmes ont toujours entouré la représentation que l'on se fait des robots. Le concept de créatures intelligentes confectionnées par la main de l'homme est déjà présent dans les mythologies telles que le mythe du Golem dans la mythologie juive ou l'histoire de Pygmalion et Galatée dans la mythologie grecque, ou encore les servantes androïdes en or du dieu Héphaïstos que l'on retrouve dans *l'Iliade* d'Homère.

On retrouve dans la littérature du XIXe siècle le thème de l'automate prenant vie, à travers des œuvres telles que *L'homme au sable* d'Ernst Theodor Amadeus Hoffmann ou le compte de fées *Pinocchio* de Carlo Collodi. C'est également à cette époque qu'est paru le célèbre roman de Mary Shelley : *Frankenstein ou le Prométhée moderne*. Cette œuvre met en scène la perte de contrôle du savant Frankenstein sur sa créature, cette dernière se révolte contre son créateur et les êtres humains en général dont il est rejeté.

Cette thématique de révolte contre l'homme sera très largement reprise dans les œuvres de science-fiction occidentales du XXè siècle. Un auteur cependant se démarque de cette ligne en présentant un recueil de nouvelles nommé *Les Robots*.

Il s'agit de l'auteur américain Isaac Asimov, qui à travers ces nouvelles présente les limites possibles de systèmes robotiques et comment s'assurer que les robots s'évertuent à contribuer au bien-être des humains en énonçant notamment les trois lois de la robotique. Ces lois, implantées au cœur même du système robotique, doivent forcer le robot à agir pour garantir l'intégrité physique de tout être humain, pour obéir aux ordres des hommes et enfin pour garantir sa propre intégrité physique.

De nos jours, la thématique de la machine échappant au contrôle de l'homme est toujours présente notamment dans le cinéma Hollywoodien à travers des œuvres telles que *Terminator*, *Matrix*, *The Machine* ou encore *Ex-Machina*. Il est cependant à constater que ces machines ont des comportements de plus en plus proches de l'être humain, de par leur intelligence mais aussi leur capacité d'établir des interactions sociales avec l'homme. Certaines œuvres cependant, dans la lignée d'Asimov, décrivent les robots comme des compagnons particulièrement utiles et dociles. On peut notamment cité l'androïde adolescent *DARYL* du film éponyme, David, le Pinocchio moderne du film *AI*, le robot médical Baymax du film d'animation *les nouveaux héros* ou encore TARS, le robot du film *Interstellar*.

La représentation que l'opinion publique se fait des robots a son importance car elle influence la perception des robots [Sundar 2016] et donc la direction donnée à la recherche. Ainsi, la culture japonaise est décrite comme robophile [Gilson 1998]. Dans ce même pays, on constate un investissement massif dans la recherche en robotique, notamment dans la robotique de service depuis plusieurs décennies.

Malgré les scénarios apocalyptiques de certaines œuvres et les angoisses sociétales liées au robot, tel la suppression d'emplois peu qualifiés, les robots sont entrés dans nos usines et commencent à arriver dans nos maisons. Les premiers modèles ont pris la forme d'automates programmables dans les industries et de robots aspirateurs ou de jouets dans les foyers. Ces premières formes ont une intelligence très limitée et liée à une tâche, le plus souvent assez basique, à accomplir. Ces premiers modèles, de par leur adaptabilité très limitée et leur intelligence qui tient plus de la réaction que du raisonnement, sont plus proches d'un automate que d'un système réellement intelligent. Cependant, on constate depuis quelques années une complexification des robots industriels et domestiques. Cette complexification permet aux robots les plus récents de prendre en compte l'homme. Ainsi, l'homme et la machine vont être amenés de plus en plus à se côtoyer, que se soit sur le lieu de travail avec des robot coéquipiers ou dans les foyers avec les robots compagnons ou assistants. C'est dans ce contexte que les robots sociaux arrivent sur le marché. Ainsi, dès le début des années 2000, des robots ayant pour but d'interagir avec l'homme sont commercialisés par Sony : tout d'abord Aibo [1] qui prends la forme d'un chien

1. https ://fr.wikipedia.org/wiki/Aibo

puis Qrio [2], un petit robot humanoïde. Dans la même lignée que Qrio, Aldebaran (aujourd'hui devenu Softbank Robotics) sort fin 2006 le robot NAO [3] qui deviendra rapidement une plateforme très utilisée dans les laboratoires de recherches et universités à travers le monde. Plus récemment, Softbank Robotics a également mis au point le robot humanoïde Pepper [4] qui est déjà utilisé comme plateforme de renseignements et d'informations commerciales. Il est également à constater que plusieurs campagnes de crowd-founding ont rencontré un franc succès pour des projets de plateformes robotiques sociales pour le domicile. On peut notamment citer les robots Jibo [5] ou Buddy [6]. Ces deux projets de plateforme robotique interagissant avec l'homme et destinée au grand public, ont reçu des retours très positifs et sont révélateurs de l'engouement de la population pour ce genre de plateforme. En effet, le secteur de la robotique de service et domestique (robots assistants/équipiers) est considéré comme un des enjeux économiques de ce siècle. On estime que d'ici à 2020, le marché de la robotique de service (tous secteurs confondus) pourrait représenter un volume supérieur à 15.69 milliards de dollars par an selon une étude de Grand View Research [7].

L'un des intérêts premier de la robotique d'assistance est lié au vieillissement de la population. En effet, les robots autonomes d'assistance pourraient permettre de redonner une certaine autonomie aux personnes âgées ainsi qu'un accès aux technologies modernes à travers l'utilisation intuitive des robots comme interface. Pour ce qui est des robots équipiers, ils pourraient permettre d'exécuter des tâches répétitives, pénibles, mais également des tâches dangereuses voir irréalisables par l'homme (par exemple celles nécessitant une grande précision) tout en travaillant en collaboration avec un opérateur humain.

Cependant, pour le robot coéquipier comme pour le robot assistant, le contact avec l'homme rend important, si ce n'est nécessaire, d'incorporer une représentation de l'homme et des comportements sociaux pour interagir avec celui-ci. Le but n'étant pas de copier l'homme ou de le remplacer, mais uniquement de mettre en place des mécanismes permettant une interaction efficace, agréable et intuitive pour l'homme. Pour cela le robot doit être capable d'estimer la situation et l'état de l'homme et exhiber des comportements pouvant être compris par l'homme. Dans cette thèse nous présenterons comment le robot peut créer et maintenir une représentation de l'environnent dans lequel il évolue ainsi que des individus avec lesquels

2. https ://fr.wikipedia.org/wiki/Qrio
3. https ://www.ald.softbankrobotics.com/fr/cool-robots/nao
4. https ://www.ald.softbankrobotics.com/fr/cool-robots/pepper
5. https ://www.jibo.com/
6. http ://www.bluefrogrobotics.com/fr/buddy-fr/
7. http ://www.grandviewresearch.com/industry-analysis/service-robotics-industry

il interagit, afin de pouvoir montrer des comportements socialement acceptables par l'homme. Ainsi, l'un des enjeux est de construire une sémantique liée au contexte de l'interaction afin d'interpréter correctement les actions physiques et les actes de langage de l'homme ainsi que son intention afin de pouvoir l'aider au mieux tout en agissant et dialoguant de façon compréhensible et acceptable par celui-ci.

1.2 Contexte de la thèse

1.2.1 Le projet MaRDi

Cette thèse a été réalisée essentiellement dans le cadre du projet MaRDi (MAn-Robot Dialogue) [8] de l'Agence Nationale pour la Recherche (ANR). Il a été financé par l'appel à projet Contenu et interactions. Ce projet vise à étudier l'apport d'une approche "située" du dialogue homme-robot. Le terme "situé" est ici relatif à l'incarnation physique d'un système de dialogue dans une plateforme robotique qui permet l'intégration d'informations issues de la perception du robot dans le contexte de l'interaction pour compléter ou lever des ambiguïtés introduites par le médium vocal. L'originalité de l'approche est de ne pas considérer les technologies vocales comme disponibles et dissociées de la tâche d'interaction homme-robot, mais bel et bien comme moyen d'en améliorer l'expérience et les performances.

1.2.2 Environnement de travail

Cette thèse a été réalisée au LAAS-CNRS dans l'équipe RIS (Robot InteractionS). Elle s'inscrit dans la continuité de travaux précédemment réalisés par cette même équipe et contribue à l'architecture globale visant à construire un système robotique complet. Les problématiques abordées sont liées à l'interaction homme-robot, du raisonnement sur les données issues de la perception à la planification de mouvements en passant par la prise de décision et la planification de tâches.

Ainsi, l'architecture cognitive globale envisagée par notre équipe pour un robot d'interaction est compatible avec le modèle d'architecture en trois niveaux décrite dans [Pacherie 2012] concernant le suivi des actions jointes entre humains. Cette architecture consiste en un premier niveau appelé niveau distal partagé (*Shared Distal level*) qui s'occupe des problèmes liés à l'intention (engagement, élaboration et suivi d'un plan partagé). Un second niveau appelé niveau proximal partagé (*Shared Proximal level*) qui s'occupe de l'exécution des actions planifiées provenant du niveau distal à haut niveau. Enfin le dernier niveau appelé intention motrice couplée (*Coupled Motor intention*) qui assure la coordination au niveau de l'espace et

8. http ://mardi.metz.supelec.fr

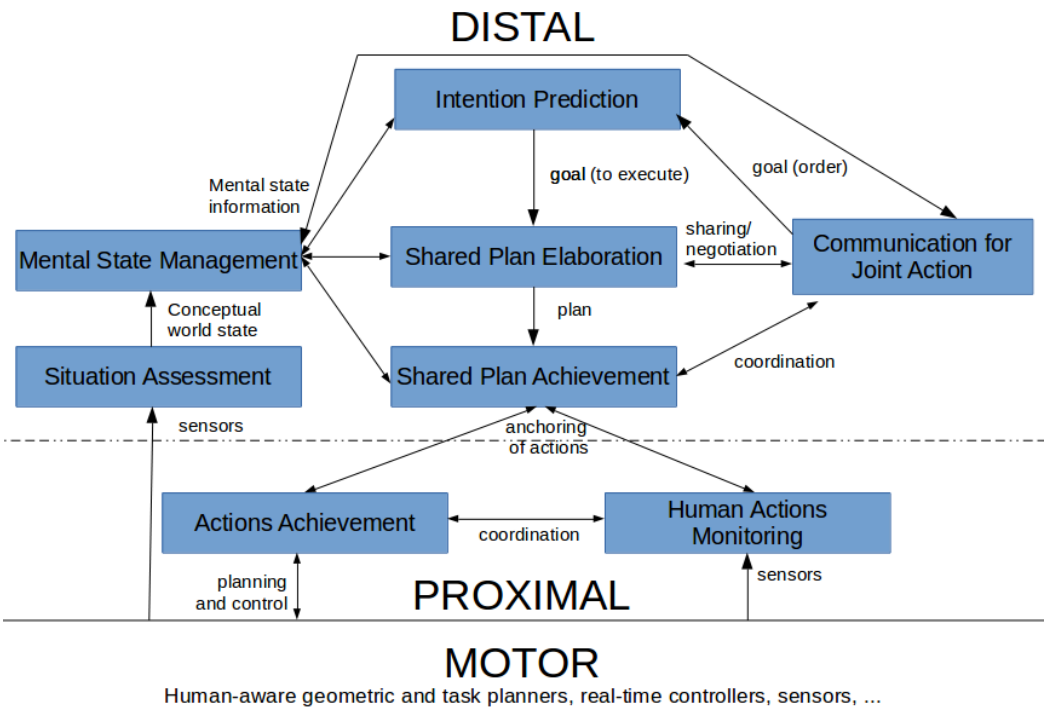


FIGURE 1.1 – Architecture cognitive pour l'interaction homme-robot mise en place par l'équipe RIS du LAAS-CNRS.

du temps durant l'exécution de l'action. Le modèle d'architecture cognitif de notre équipe est décrit dans [Alami 1998b], qui présente une architecture robotique avec trois niveaux. Cette architecture a par la suite été développée pour l'interaction Homme-Robot [Alami 2011, Alami 2013, Fiore 2014] afin d'intégrer des capacités de prise en compte de l'homme dans les différents niveaux. Basé sur ces travaux, nous avons présenté dans [Devin 2016] les compétences qui nous semblent essentielles pour un robot interagissant avec l'homme ainsi que l'architecture cognitive qui permet de doter le robot de ces capacités. Cette architecture est présentée à la figure 1.1.

1.3 Motivations

1.3.1 Scénario

La robotique d'assistance et plus généralement la recherche concernant la conception et la confection de systèmes interagissant avec l'homme a encore de nombreux défis à relever. Pour exposer certains de ces défis, nous proposons un scénario d'illustration.

Imaginons un couple, Bob et Alice, vivant dans un appartement avec un robot

d'assistance. Prenons la situation où Alice rentre du travail et décide de cuisiner une nouvelle recette avec l'aide de son robot. Avant de commencer la préparation, Alice demande à son robot de l'aider à réunir les ingrédients et ustensiles pour faire la recette. Pour pouvoir aider efficacement Alice, le robot a donc besoin de capacités élémentaires telles que la navigation, la perception et la préhension d'objets.

Par exemple, Alice peut demander au robot d'apporter un objet qui est sur la table du salon. Pour comprendre Alice, à supposer que le robot soit capable de reconnaissance vocale, il lui faut comprendre également les concepts énoncés par Alice et les relier à la réalité du contexte situé. Ceci implique pour le robot de raisonner sur l'environnement afin de produire ce genre de relations spatiales (ici un objet *"sur"* un autre). Ceci afin de comprendre et d'être compris de l'humain avec lequel il interagit.

Imaginons à présent qu'Alice ait besoin de son batteur à œufs qui est dans la commode du salon. Imaginons également que pendant la journée, Bob ait utilisé le batteur de la commode sans le ranger à son emplacement habituel, et que le robot ait perçu ce changement. Un comportement utile et proactif pour le robot serait, en voyant Alice se déplacer vers la commode du salon, de la prévenir du changement de position du batteur. Pour parvenir à cela, le robot doit être capable de se représenter les états de connaissances des humains qui l'entourent ainsi que d'interpréter leurs intentions en fonction du contexte (ici chercher les ustensiles pour réaliser une recette), de leur état mental (ici Alice pense que le batteur est dans la commode) et de leurs actions (Alice se dirige vers la commode du salon).

Une fois que les ustensiles et les ingrédients sont réunis, il reste à confectionner le plat. Imaginons qu'Alice demande au robot de la guider pour effectuer une tarte aux pommes. Le robot doit alors être capable d'élaborer un plan collaboratif prenant en compte les compétences d'Alice mais aussi pouvoir négocier ce plan avec Alice en fonction de ses préférences et des capacités d'exécution de chacun. Pour ne pas être perçu comme ennuyeux, le robot doit également pouvoir adapter son niveau de détail explicatif au niveau de compétence d'Alice sur les tâches du plan qui lui incombent. Cela nécessite pour le robot d'avoir une représentation et une capacité de suivi des connaissances des humains concernant les tâches à accomplir et de mettre en place un raisonnement pour exploiter ces informations au mieux durant l'interaction.

1.3.2 Défis liés à la thèse et contributions associées

Parmi les défis soulevés par le scénario présenté dans la partie précédente, certains seront traités dans cette thèse.

Le premier défi est de permettre au robot d'obtenir une représentation de l'environnement qui l'entoure. La contribution n'est pas au niveau de la perception proprement dite mais au niveau des raisonnements mis en place pour permettre au robot, à partir de l'agrégation des données de perception, de maintenir une représentation tridimensionnelle du monde qui l'entoure et des différentes entités présentes (objets, humains et robots). À partir de cette représentation tridimensionnelle de l'environnement maintenue par le robot, celui-ci est capable de mettre en œuvre divers raisonnements spatio-temporels permettant d'estimer la situation de l'environnement. Cette couche de représentation symbolique de l'état du monde permet de réduire l'écart entre les données de perception (sub-symboliques) avec la couche décisionnelle (la supervision). Cette contribution a été présenté dans l'article [Milliez 2014a] et sera présentée au chapitre 2.

La seconde contribution reliée également à cette estimation de la situation est la mise en place d'un modèle d'état mental pour chaque agent de l'interaction. Ainsi, en plus de connaître la situation de l'environnement, le robot est capable d'estimer la situation pour les autres. Cette capacité, appelée prise de perspective dans la littérature de psychologie, est une capacité essentielle pour de nombreux aspects de l'interaction entre agents sociaux. Il est donc crucial pour le robot d'être lui aussi doté de ce type de capacité cognitive. Cette contribution a été présentée dans l'article [Milliez 2014a] et une implémentation de ce concept a été utilisé au cours d'interactions avec des humains. Cette contribution est détaillé dans le chapitre 3.

En plus de ces deux contributions scientifiques liées à l'estimation de la situation, nous avons développé une infrastructure logicielle nommée TOASTER (Tracking Of Agent and Spatio-TEmporal Reasonning) [9]. TOASTER est l'implémentation des deux contributions scientifiques décrites ci-dessus. Il est disponible en Open-Source et a été conçu de façon générique afin de profiter au plus grand nombre. En effet les calculs et raisonnements réalisés sont indépendants des données d'entrées (des capteurs) et de la plateforme robotique.

L'établissement d'une couche symbolique représentant l'état du monde permet au robot de mettre en place un dialogue situé de qualité. Cette contribution est complétée par une étude montrant comment nous avons amélioré l'efficacité et le taux de succès du dialogue situé en utilisant l'état mental de l'homme pour mieux comprendre ses propos. Au cours de cette étude, nous avons été amenés à utiliser le simulateur robotique Open-Source MORSE et à contribuer au développement de certaines fonctionnalités pour l'interaction homme robot dans ce simulateur décrit dans [Lemaignan 2014]. Cette étude a fait l'objet d'une autre publication dans laquelle nous avons utilisé le simulateur robotique Open-Source MORSE pour en-

9. http ://www.gregoire.milliez.fr/toaster/index.html

traîner le système de dialogue à interagir avec l'homme en utilisant les données contextuelles [Milliez 2014b]. Une description de l'utilisation de l'état mental de l'homme au cours du dialogue situé ainsi qu'une évaluation de l'amélioration de l'interaction est faite dans l'article [Ferreira 2015b]. Ces travaux sont décrits au chapitre 4 de cette thèse.

Pour aller plus loin dans l'estimation de la situation de l'homme, et basé sur son état mental modélisé et maintenu par la capacité de prise de perspective, nous avons mis en place un mécanisme permettant d'estimer l'intention de l'homme. Ainsi, il est possible pour le robot d'interpréter les actions de l'homme par rapport à son état mental, et ainsi d'améliorer les performances dans les situations où l'homme possède une croyance erronée sur l'environnement. Cette contribution est complétée par l'utilisation qui est faite de cette estimation de l'intention pour donner au robot un comportement proactif afin d'aider au mieux l'humain. Cette étude est présentée au chapitre 5.

Le dernier défi relevé par cette thèse est de mettre en place une modélisation de l'état mental de l'humain, cette fois-ci au niveau des connaissances qu'il peut avoir concernant certaines tâches d'un plan partagé. Cette modélisation permet d'adapter la génération du plan collaboratif et l'exécution de ce plan au niveau d'expertise de l'humain. Cette modélisation des connaissances de l'homme et l'utilisation qui en est fait, ainsi qu'une étude utilisateur visant à mesurer la perception de l'amélioration de l'interaction par les utilisateurs sont présentées dans l'article [Milliez 2016]. Ces travaux sont rapportés au chapitre 6.

1.4 Publications

Les publications suivantes sont liées aux contributions scientifiques de cette thèse (présentée de la plus récente à la plus ancienne) :

- **Grégoire Milliez**, Raphaël Lallement, Michelangelo Fiore et Rachid Alami, *"Using human knowledge awareness to adapt collaborative plan generation, explanation and monitoring"*, The Eleventh ACM/IEEE International Conference on Human Robot Interaction, (**HRI 2016**)
- Sandra Devin, **Grégoire Milliez**, Michelangelo Fiore, Aurélie Clodic et Rachid Alami, *"Some essential skills and their combination in an architecture for a cognitive and interactive robot"*, The Eleventh ACM/IEEE International Conference on Human Robot Interaction, (**2nd Workshop on Cognitive Architectures for Social Human-Robot Interaction, HRI 2016**)
- Michelangelo Fiore, Harmish Khambhaita, **Grégoire Milliez** et Rachid Alami, *"An Adaptive and Proactive Human-Aware Robot Guide"*, Interna-

tional Conference on Social Robotics, (**ICSR 2015**)

— Emmanuel Ferreira, **Grégoire Milliez**, Fabrice Lefèvre et Rachid Alami, *"Users' Belief Awareness in Reinforcement Learning-Based Situated Human–Robot Dialogue Management"*, International Workshop on Spoken Dialog Systems, (**IWSDS 2015**)

— **Grégoire Milliez**, Emmanuel Ferreira, Michelangelo Fiore, Rachid Alami et Fabrice Lefèvre, *"Simulating human-robot interactions for dialogue strategy learning"*, Simulation, Modeling, and Programming for Autonomous Robots, (**SIMPAR 2014**)

— Séverin Lemaignan, Marc Hanheide, Michael Karg, Harmish Khambhaita, Lars Kunze, Florian Lier, Ingo Lütkebohle et **Grégoire Milliez**, *"Simulation and HRI recent perspectives with the MORSE simulator"*, Simulation, Modeling, and Programming for Autonomous Robots, (**SIMPAR 2014**)

— **Grégoire Milliez**, Matthieu Warnier, Aurélie Clodic et Rachid Alami, *"A framework for endowing an interactive robot with reasoning capabilities about perspective-taking and belief management"*, Robot and Human Interactive Communication, 2014 RO-MAN : The 23rd IEEE (**RO-MAN 2014**)

1.5 Organisation du manuscrit

Dans ce manuscrit, nous présentons tout d'abord dans le chapitre 2, comment notre système est capable de construire et maintenir une représentation de l'état du monde. Nous montrerons également l'évaluation de la situation de l'environnement et des agents faite à partir de calculs spatio-temporels sur l'état du monde qu'il maintient. Dans un second temps, pour aller plus loin dans la compréhension de la situation des humains, nous expliquerons dans le chapitre 3 comment nous avons doté notre robot de prise de perspective. Par la suite nous expliquerons dans le chapitre 4 comment l'évaluation de la situation permet d'établir un dialogue situé avec l'homme, et en quoi la capacité de gérer explicitement des croyances divergentes permet d'améliorer la qualité de l'interaction et la compréhension de l'homme par le robot. Dans le chapitre 5 nous montrerons comment la connaissance de la situation et la possibilité de raisonner en se mettant à la place de l'homme permet une reconnaissance d'intentions appropriées de celui-ci et comment nous avons pu grâce à cela doter notre robot de comportements proactifs pour venir en aide à l'homme. Pour finir, nous présenterons en chapitre 6 une étude présentant un système de maintien d'un modèle des connaissances de l'homme sur diverses tâches et qui permet une gestion adaptée de l'interaction lors de l'élaboration interactive et l'accomplissement d'un plan partagé.

Chapitre 2

Construction et Maintien de l'État du Monde

Sommaire

2.1 Contexte

2.1.1 Défis à relever

Le robot est un système pouvant percevoir, comprendre et agir sur son environnement. Comprendre son environnement suppose de pouvoir raisonner sur les données de perception afin d'en extraire des données de plus haut niveau d'abstraction, permettant d'avoir une estimation de la situation et ainsi de prendre les décisions adéquates pour accomplir la tâche qui lui incombe. C'est la célèbre boucle perception-décision-action.

Avec le progrès récent de la robotique, les robots commencent à arriver dans les usines pour travailler aux côtés d'humains, ainsi que dans les foyers comme robot d'assistance. Adapter les capacités de raisonnements du robot à ce nouveau monde, qui est par définition configuré pour l'homme, représente un défi pour la recherche. Cet environnement humain est composé d'éléments statiques tel que les murs, d'éléments pouvant évoluer au cours d'une interaction (déplacement, remplissage, activation...) que nous appellerons les objets, et enfin d'éléments pouvant se mouvoir, agir sur les objets, et interagir entre eux que nous appellerons les agents. Ces agents peuvent être robotiques ou humains. Pour pouvoir agir dans un environnement humain, le robot doit donc pouvoir percevoir tous ces éléments énumérés précédemment et en extraire une représentation symbolique afin de décrire au mieux la situation et permettre à la couche décisionnelle d'agir de façon appropriée.

2.1.2 Évaluation de la situation

Le concept de "Situation Awareness", qui pourrait se traduire par la "reconnaissance de situation", a été identifié d'après Gilson [Gilson 1995] durant la première guerre mondiale par Oswald Boelke qui a réalisé "l'importance d'obtenir une connaissance de l'ennemi avant que l'ennemi n'obtienne une connaissance similaire, et conçu des procédés pour y parvenir". Le terme de Situation Awareness est couramment utilisé dans le domaine de l'Interaction Homme-Machine. Parmi les scientifiques ayant étudié ce principe, Mica Endsley en a énoncé une définition générale. Elle définit la connaissance de la situation comme étant "la perception d'éléments de l'environnement dans un volume de temps et d'espace, la compréhension de leur signification et la projection de leur état dans un futur proche" [Endsley 2000]. Le chapitre 7, "Situation Awareness" du livre [Pew 1998] présente une discussion et une étude sur les différentes définitions (données dans divers articles tel que [Stiffler 1988, Noble 1989, Fracker 1988]...) et systèmes pour la connaissance de la situation. Dans l'article [Stanton 2001], la connaissance de la situation est abor-

dée du point de vue de la sécurité. L'article présente et discute trois théories de la connaissance de la situation : le modèle en trois niveau d'Ensley [Endsley 1995], l'approche basée sur les sous-systèmes interactifs [Bedny 1999] et le cycle perceptuel [Smith 1995].

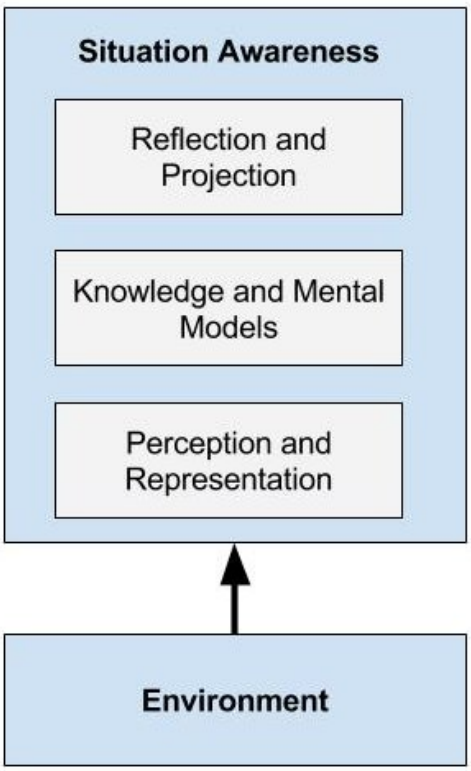


FIGURE 2.1 – Schéma présentant le modèle en trois niveaux pour la connaissance de la situation présentée par Ensley.

Nous détaillons ici les trois niveaux d'Ensley.

— Premier niveau : perception des éléments de l'environnement. Ce niveau est l'identification des éléments ou événements clés qui, en les combinant servent à définir la situation. Ce premier niveau marque sémantiquement les éléments important de la situation pour les niveaux d'abstraction supérieure dans les processus suivants.

— Deuxième niveau : compréhension de la situation courante. La compréhension provient de la combinaison des éléments du premier niveau pour avoir une représentation plus globale. Ce niveau permet de définir l'état courant en des termes pertinents qui permettent la prise de décision et d'entreprendre d'agir sur l'environnement.

— Troisième niveau : prédiction des états futurs. Ce niveau est la projection

de la situation courante vers le futur, dans l'environnement, afin d'essayer de prédire l'évolution de la situation. La précision de la prédiction dépend largement de la précision des deux niveaux inférieurs de connaissance de la situation. L'anticipation du futur projeté permet à l'agent décisionnel (l'opérateur ou le superviseur) d'avoir le temps de résoudre certains problèmes avant qu'ils n'arrivent et de mettre en place un plan d'action pour atteindre l'objectif.

Cette composition est résumée à la figure 2.1.

Dans le cas de systèmes en évolution et d'environnements dynamiques, le processus permettant d'acquérir et de maintenir la connaissance de la situation est appelé évaluation de la situation (situation assessment). Endsley explique dans [Endsley 1995] que "la connaissance de la situation incorpore la compréhension d'un opérateur sur la situation globale, formant ainsi la base pour la prise de décision". Les processus d'évaluation de la situation sont donc fortement liés aux processus de prise de décision (qui constituent la supervision dans un système robotique). L'une des principales préoccupations pour l'Interaction Homme-Machine est de concevoir des interfaces utilisateurs qui permettent à un opérateur d'acquérir la connaissance de la situation de façon pertinente afin qu'il puisse décider quelle décision prendre.

De la même manière, en robotique, le robot a besoin de décider quelle action entreprendre et quand agir sur son environnement. Pour prendre ce genre de décisions, la supervision repose sur des processus de plus bas niveau pour obtenir des informations à haut niveau d'abstraction sur l'environnement. Cela implique, pour l'évaluation de la situation, de raisonner sur ce qui entoure le robot à partir des données des capteurs afin de modéliser la configuration de l'environnement physique autour du robot et de produire un ensemble de propriétés concernant les divers éléments. Ceci permet de fournir une représentation symbolique de l'état du monde au superviseur afin de faciliter son raisonnement et sa prise de décision. Pour créer cette représentation symbolique, il est d'abord nécessaire au robot d'avoir une bonne estimation de l'état du monde et d'en avoir une représentation tridimensionnelle en regroupant efficacement les données de perception issues des divers capteurs.

2.1.3 État de l'art

Dans le domaine de l'Intelligence Artificielle, plusieurs méthodes ont été développées pour effectuer l'évaluation de la situation. Nous allons brièvement décrire ces différentes méthodes.

2.1.3.1 Évaluation de la situation basée sur des règles expertes

Un système d'évaluation de la situation basé sur des règles (rule-based en Anglais) est un système de connaissance qui repose sur des règles préétablies pour l'inférence et sur une base de faits. L'expertise liée au domaine est directement intégrée au système par la définition des différentes règles spécifiques au domaine d'application. L'état du monde symbolique en découlant est représenté comme un ensemble de faits dans une base de faits. Généralement, les règles utilisent des données d'environnements (pouvant provenir de calculs sur les données capteurs) ou des faits provenant d'autres règles, et en fonction de certaines conditions logiques, produisent un nouveau fait qui sera ajouté à la base de faits. Par exemple, un système météorologique pourrait avoir la règle suivante :
$HIGH\_TEMPERATURE \wedge NO\_CLOUD \implies GOOD\_WEATHER.$
Dans cet exemple, les propriétés $HIGH\_TEMPERATURE$ et NO_CLOUD sont issues de capteurs. Si ces deux propriétés sont vraies, le fait $GOOD\_WEATHER$ sera produit et ajouté à la base de faits. Dans le cas d'un système basé sur des règles expertes, cette implication serait directement ajoutée au système par l'expert du domaine (par exemple ici un expert en météorologie). Les modules responsables de la production des faits basés sur ces règles sont appelés des raisonneurs.

En identifiant les divers éléments et leurs relations, les règles peuvent être utilisées pour déduire des informations de plus haut niveau correspondant à des faits décrivant des situations particulières. Ceci permet à la partie décisionnelle, c'est à dire l'opérateur pour une Interface Homme-Machine (IHM) ou le superviseur pour un système autonome tel qu'un robot, de baser sa décision sur des faits décrivant la situation de façon pertinente.

Ce type d'approche a été utilisé dans divers domaines, comme par exemple l'évaluation de la situation pour la surveillance côtière [Edlund 2006], l'évaluation de procédure d'approche pour l'atterrissage d'avion [Baron 1980, Milgram 1984], le dépannage de voiture ou encore le diagnostic médical [Swartout 1985, Miller 1982]. L'un des avantages principaux des systèmes basés sur des règles est qu'ils sont relativement simples et naturels dans leur implémentation. Cependant, l'un des inconvénients de tels systèmes est le manque de modélisation de l'incertitude. Même si certaines variantes de systèmes basés sur des règles incorporent des valeurs d'incertitudes dans les règles, cette approche implique certaines restrictions et une attention toute particulière lors du développement des règles comme présenté au chapitre 15 du livre [Russell 2003].

2.1.3.2 Évaluation de la situation basée sur les cas

Le raisonnement basé sur les cas (Case-based reasoning en Anglais) est un paradigme dans lequel la connaissance est représentée comme un ensemble de cas individuels qui est utilisé pour raisonner sur de nouvelles situations. Tous les cas partagent une même structure et sont définis par un ensemble de caractéristiques. Lorsqu'un nouveau problème, défini comme le cas recherché, est présenté au système, il est comparé à tous les autres cas de la bibliothèque, et le cas le plus proche est utilisé pour résoudre le nouveau problème. La proximité est définie par une mesure basée sur un sous-ensemble des caractéristiques du cas appelé indice de caractéristiques. Il existe différents types de mesures de similarité : distance Euclidienne, distance d'Hamming, la métrique de la valeur de différence de Stanfill et Waltz (1986)...

En général, le cas le plus proche retrouvé ne correspond pas parfaitement au cas recherché. C'est pourquoi, les systèmes basés sur les cas ont souvent recours à des règles pour adapter le cas retrouvé au cas recherché. L'un des intérêts d'utiliser le raisonnement basé sur les cas pour l'estimation de la situation et la prise de décision est que cette démarche est proche de celle utilisée par les hommes pour interagir sur leur environnement. Dans le domaine de la théorie de la mémoire humaine, ce cas de connaissance liée à un épisode spécifique est appelé la mémoire épisodique [Tulving 1972].

L'approche basée sur les cas peut également être préférable lorsqu'il n'y a pas de modèles valides du domaine. En effet, dans certaines applications l'espace d'états peu être trop vaste pour permettre la construction d'un domaine complet et pertinent. En l'absence de la modélisation du modèle, les solutions basées sur les cas offrent une alternative aux solutions basées sur le modèle.

2.1.3.3 Évaluation de la situation basée sur des Réseaux Bayésiens

Pour réaliser l'estimation de la situation de façon probabiliste, de nombreux systèmes ont recours à divers variantes des réseaux Bayésiens (BN pour Bayesian Networks) [Bladon 2002, Das 2002, Higgins 2005]. Les réseaux Bayésiens (aussi appelés réseaux de croyance (Belief Networks) ou réseaux de croyance Bayésien (BBN pour Bayesian Belief Networks) sont des structures probabilistes qui permettent un raisonnement cohérent à partir d'informations incertaines [Pearl 1986].

Les BNs sont basés sur des règles Bayésiennes qui permettent le calcul d'une probabilité postérieure associée à une proposition particulière en fonction des probabilités antérieures et des probabilités conditionnelles qui y sont associées. Il est possible de représenter un BN sous la forme d'un graphe où les nœuds correspondent

aux propositions et les arrêtes directionnelles représentent le lien causal entre les propositions. Lors de la mise en place d'un BN, les probabilités à priori et conditionnelles doivent être spécifiées pour chaque nœud. Les probabilités postérieures pour chaque nœud peuvent être mises à jour selon les règles Bayésiennes lorsque une nouvelle information est présentée au réseau.

La capacité de gérer l'incertitude et une implémentation relativement simple rendent les réseaux de croyance Bayésiens très attractifs. Cependant, l'ensemble des variables est fixe et fini, et chaque variable a un ensemble fixe de valeurs possibles [Russell 2003]. Les BNs classiques ne peuvent représenter des concepts abstraits tel que les relations entre éléments, ce qui limite l'application des BNs pour des domaines complexes. Pour gérer ces contraintes, des extensions ont été créées. Ces extensions s'appellent RPM pour modèles relationnels probabilistes (Relational Probabilistic Models) [Howard 2005, Russell 2003].

Un autre inconvénient lié à l'usage des BNs classiques est qu'ils n'ont pas de mécanisme propre pour le raisonnement temporel. De nombreuses propositions existent pour palier à ce manque [Russell 2003]. Il est également possible d'utiliser un BN classique en ajoutant des noeuds multi-états correspondant à une "mémoire" [Higgins 2005].

Enfin, dues à l'aspect probabiliste, des valeurs statistiques adéquates sont nécessaires pour obtenir des modèles probants. Ceci peut nécessiter des connaissances à priori sur la répartition de probabilité ou un large panel de données pour entraîner le système, sans quoi une approximation des distributions doit être réalisée ce qui peut s'avérer compliqué. De plus, modéliser les relations causales peut également s'avérer complexe selon la complexité du domaine concerné.

Higgins [Higgins 2005] propose une approche pour la reconnaissance d'événements en utilisant les réseaux Bayésiens. Nous reviendrons sur ce type d'utilisation de réseaux Bayésiens pour tenter de reconnaître les actions de l'homme et estimer son intention au chapitre 5.

2.1.3.4 Évaluation de la situation en robotique

En robotique, l'aspect dynamique de l'environnement et le besoin d'agir et réagir en fonction de celui-ci rend l'évaluation de la situation indispensable. L'estimation de la situation en robotique est fortement liée au choix de l'architecture robotique. Ainsi Shakey [Nilsson 1969], l'un des tout premiers robots, avait une architecture en trois parties : perception, planification, exécution. Cette approche est appelée le paradigme sense-plan-act (SPA). La partie "perception" avait en charge de fournir une représentation du monde à la planification. L'équipe de Stanford, ayant réalisé

cette étude, avait déjà identifié l'importance de mettre à jour le modèle d'environnement dynamique : "comme le monde évolue, soit par l'action du robot lui-même ou pour d'autres raisons, le modèle doit être mis à jour pour enregistrer ces changements. De même, les nouvelles informations apprises sur l'environnement doivent être ajoutées au modèle". L'une des caractéristiques de ce type d'architecture est que les données capteurs sont utilisées pour créer un modèle du monde, réalisant ainsi une évaluation de la situation *basée sur des règles expertes* afin de permettre la planification pour atteindre un but donné. L'un des inconvénients de ce système est que les capteurs utilisés pour créer le modèle d'environnement servant de base pour la planification ne sont pas directement utilisés lors de l'exécution.

Par la suite, pour pallier le manque de sécurité lors de l'exécution d'actions d'architectures robotiques basées sur le paradigme SPA, et pour éviter une planification coûteuse en temps et puissance de calcul, une nouvelle forme d'architecture robotique, plus réactive et plus connectée aux données capteurs a été mise en place. Il s'agit de l'architecture Subsumption ou "behavior-based" [Brooks 1986]. Dans ce type d'architecture, les actionneurs sont reliés aux capteurs de façon beaucoup plus directes, ce qui permet d'obtenir un système plus réactif. Cependant, l'inconvénient d'une telle architecture robotique est que le système n'a pas de représentation globale de la situation et a donc un comportement principalement réactif. L'absence de représentation globale du monde qui entoure le robot ne permet pas de prendre des décisions à long terme ou d'optimiser la prise de décision. Dans une telle architecture l'évaluation de la situation est de type *basée sur le cas* (case-based).

La plupart des architectures robotiques récentes visent à intégrer à la fois la réactivité des architectures basées sur le comportement (Subsumption) et la capacité de délibération des architectures SPA. Afin d'atteindre cet objectif, l'architecture 3T (de l'Anglais "3 tiers", trois niveaux) a été mise en place [Bonasso 1995]. En terme d'évaluation de la situation, ces trois niveaux sont comparables aux trois niveaux d'Ensley présentés en section 2.1.2 comme nous allons le montrer ci-dessous.

Le niveau le plus bas appelé "Behavioral layer" est en charge des capteurs pour la perception et des actionneurs pour l'exécution. À ce niveau, les données capteurs sont traitées pour fournir des données pertinentes. Par exemple, pour le dialogue, les données brutes du microphone sont transformées en mots, cependant ceci ne sont pas encore interprétés. L'interprétation sémantique de cette entrée sera faite dans le niveau supérieur. Ce premier niveau prend également en charge les actionneurs qui reçoivent des commandes de la couche supérieure et qui les exécutent tout en utilisant les données capteurs de cette même couche. En terme d'estimation de la situation, cette couche est comparable au niveau "perception et représentation" pour la connaissance de la situation présentée par Ensley.

Le second niveau, aussi appelé "Executive layer" est en charge de l'interprétation sémantique des données issues de la couche inférieure et du séquençage d'actions à accomplir pour atteindre la tâche donnée par le niveau supérieur. Ce niveau pilote donc les sous composants d'exécution de la couche inférieure en fonction de l'estimation de la situation et de séquences d'actions pour accomplir une tâche. En terme d'estimation de la situation, si on se réfère à nouveau aux trois niveaux d'Ensley, ce second niveau par l'interprétation sémantique des données réalisée correspond bien au second niveau d'Ensley (le niveau de connaissance et d'états mentaux).

Le dernier niveau appelé "Planning layer" est en charge de la vision à long terme du système pour accomplir un but. Elle utilise les données issues de la couche inférieure pour décider de la suite de tâches à entreprendre pour accomplir un but. De nouveau, en terme d'estimation de la situation, ce dernier niveau correspond au niveau "Reflection et projection" d'Ensley.

Comme présenté en section 1.2.2 et dans [Alami 1998a], notre architecture est également en trois niveaux. Le plus bas niveau, appelé niveau fonctionnel, est constitué d'un ensemble de modules qui contiennent des algorithmes de contrôle et de perception dynamiquement paramétrable. La couche exécutive quand à elle ne contient pas de séquence d'actions. Elle se contente de recevoir les commandes du niveau supérieur et de paramétrer les modules inférieurs en conséquence. Le dernier niveau est constitué d'un superviseur [Fiore 2014] et d'un planificateur de tâches de type HTN [Guitton 2012]. Le superviseur pilote le planificateur de tâches en demandant de résoudre un but, puis pilote les autres modules pour exécuter chaque action du plan et surveiller l'évolution de celui-ci. Le superviseur peut ainsi, lorsqu'il détecte une incohérence (une action n'ayant pas eu l'effet prévu, l'échec d'une action ou encore l'impossibilité d'exécuter l'action) demander un nouveau plan au planificateur.

Concernant l'évaluation de la situation, de nombreux systèmes robotiques ont recours à un composant d'évaluation de la situation qui correspond aux besoins du robot dans une tâche applicative particulière. Dans [Beck 2011], le système d'évaluation de la situation est basé sur une chaîne de Markov dynamique pour modéliser les états de l'environnement et leur évolution. Les auteurs présentent une application pour un robot mobile navigant dans un passage étroit. [Kluge 2001] présente une évaluation de la situation empirique pour robot mobile dans une zone à forte influence, permettant de reconnaître les situations d'obstruction volontaire.

Notre système permet non seulement d'effectuer l'agrégation des données de perception pour modéliser l'état du monde afin de pouvoir réagir à l'environnement et exécuter les actions en prenant en compte l'aspect dynamique de ce qui l'entoure, mais aussi la production de faits symboliques basés sur des calculs géométriques,

ce qui permet le raisonnement symbolique au niveau des modules supérieurs dans l'architecture, à savoir la supervision et la planification. Nos travaux peuvent être comparés au "Grounded Situation Model" (GSM) introduit par Mavridis et Roy [Mavridis 2005] dans le sens où ils fournissent tous les deux une représentation amodale du monde utilisée comme un médiateur entre les capteurs et le modèle symbolique. [Coradeschi 2013] présente une étude sur ces systèmes de fondement des symboles (symbol grounding [Harnad 1990]) en robotique. Le fondement de symboles permet de relier la représentation symbolique de concepts avec leurs équivalents non symboliques dans le monde réel. [Daoutis 2009] [Lemaignan 2011] présentent des exemples d'utilisations de ce genre de systèmes. Les applications pour le raisonnement spatial [O'Keefe 1999] sont multiples. Par exemple, cela a été utilisé pour le traitement du langage naturel pour des applications telles que la reconnaissance de direction [Kollar 2010, Matuszek 2010] ou le fondement du langage (language grounding) [Tellex 2010]. [Skubic 2004] présente un raisonneur spatial intégré dans un robot qui calcule la position symbolique des objets.

Nos travaux peuvent également être comparés aux travaux de Heinz [Heintz 2009], dans le sens où contrairement à beaucoup d'applications robotiques, nous ne cherchons pas à relier des capteurs à un raisonneur pour une application particulière (de façon "had-hoc") mais nous cherchons à créer une infrastructure logicielle permettant le raisonnement sur l'environnement du robot pour diverses applications d'interactions homme-robot et pour combler l'écart entre les données capteurs et le raisonnement symbolique. Heinz définit le terme de "knowledge processing middleware"[Heintz 2008], ou intergiciel de traitement de données, comme étant "une infrastructure logicielle systématique et fondée sur des principes pour combler le fossé entre détection et raisonnement dans un agent physique". Ainsi, nos travaux répondent à de nombreuses caractéristiques requises pour être considérés comme un intergiciel de traitement de données. En effet, Heinz déclare qu'une telle infrastructure doit fournir à la fois un aspect conceptuel et une implémentation permettant d'intégrer une large variété de fonctionnalité et de gestion de l'information. Ainsi, Heinz dans [Heintz 2010] décrit DyKnow. DyKnow est un intergiciel (ou middleware) de traitement de connaissances basé sur des flux. Le but de DyKnow est de combler l'écart entre "les données sur le monde collectées généralement par des capteurs et les données supposées disponibles qui sont utilisées par les fonctionnalités délibératives".

En plus de la description conceptuelle que nous allons faire ici, nous présenterons, en section 2.7, TOASTER, une infrastructure logicielle open-source implémentant les concepts énoncés. De plus, notre système crée une représentation de l'environnement à différents niveaux d'abstraction (sub-symbolique et symbolique),

et peut être configuré ou reconfiguré de façon flexible et en ligne (par exemple par le superviseur). De plus il incorpore un raisonnement "bottom-up" pour la production de données symboliques et "top-down" pour raisonner sur des données perceptives potentiellement incomplètes (voir section 2.5) et permet de prendre en compte l'incertitude.

Cependant, nos travaux portent principalement sur l'aspect "représentation" du premier niveau décrit par Ensley et sur les connaissances et les modèles mentaux constituant le second niveau plutôt que sur la "perception" à proprement parler. Ainsi, notre système n'effectue pas de traitement de données brutes comme le traitement d'images mais utilise les sorties de tels logiciels (par exemple le nom et la position des objets). Nous utilisons une approche modulaire où chaque module partage une représentation tridimensionnelle du monde. Cette représentation est utilisée pour le calcul de divers faits et primitives concernant notamment les relations spatiales, la cinématique et l'état des divers éléments de l'environnement. Une valeur de "confiance" est associée à chaque fait, permettant ainsi le raisonnement probabiliste. Différents modules peuvent donc utiliser différentes techniques d'évaluation de la situation, probabiliste ou basé sur des règles expertes. Notre infrastructure pour l'évaluation de la situation comporte également certaines spécificités telles des raisonnements "top-down" permettant de pallier certains manques liés à la perception. Notre étude ne concerne pas la fusion des données capteurs (voir chapitre 25 du livre [Siciliano 2008]) ni les différentes méthodes permettant de réaliser la perception à proprement parlée des divers éléments de l'environnement, cependant, afin de donner un contexte à notre étude, nous présenterons brièvement certaines de ces méthodes. En effet, nos travaux sur l'estimation de la situation pour l'interaction homme-robot se concentrent sur les processus de plus haut niveau, à savoir l'agrégation et l'interprétation contextuelle des données issues de la perception pour évaluer la situation.

L'architecture et la composition de notre infrastructure logicielle permettant de réaliser l'évaluation de la situation ayant grandement évolué au cours du temps, nous présenterons les concepts et la configuration correspondant à la version la plus récente du système.

2.2 Perception et données capteurs

2.2.1 Éléments statiques

Un robot amené à interagir avec des humains est également amené à partager leur environnement. Que se soit dans un foyer ou une entreprise, cet environnement

est composé de murs, d'escaliers de meubles... autant d'éléments statiques pouvant être des obstacles pour se mouvoir et manipuler les objets dans l'environnement. Ces éléments étant considérés comme "statiques", leur configuration peut être stockée dans la mémoire du robot (par exemple sous forme de carte 2D pour la navigation, comme dans la figure 2.2) et directement utilisée au niveau de l'exécution lorsque l'action à accomplir nécessite une navigation ou une manipulation, sans avoir à passer par le niveau décisionnel. Connaître l'empreinte au sol des différents éléments statiques n'est pas toujours suffisant. Ainsi pour effectuer une manipulation d'objet sur des meubles, le robot a besoin de connaître le modèle tridimensionnel du meuble afin d'éviter toute collision lors de la manipulation. Cette connaissance peut provenir soit d'une perception tridimensionnelle de l'environnement, soit d'une connaissance à priori du modèle tridimensionnel du meuble.

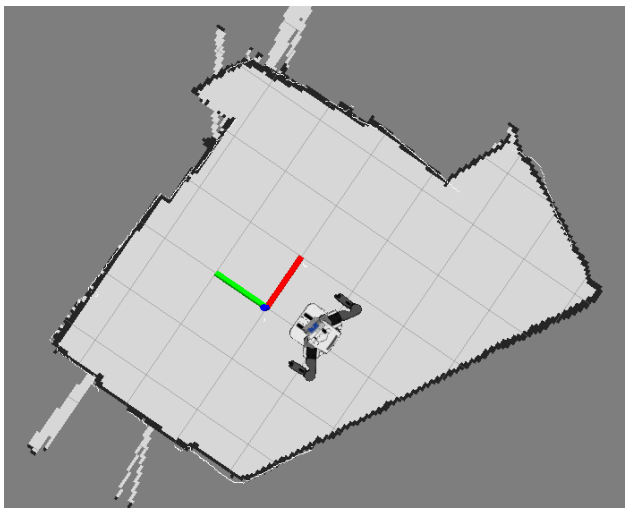


FIGURE 2.2 – Exemple de carte pour la navigation 2-D et la localisation du robot.

2.2.2 Objets

Les lieux de travail et de vie sont également composés d'objets divers. Récipients, outils, vêtements, appareils électroniques... autant d'éléments qui peuvent prendre des formes et fonctionnalités variées. Certains de ces objets peuvent également avoir des états. Par exemple, une bouteille peut être vide ou pleine, un vêtement peut être propre ou sale, un appareil électronique peut être éteint ou allumé. Ces objets sont pour la plupart manipulables au cours d'une interaction. Il est donc important que le robot puisse, afin d'agir correctement sur son environnement, reconnaître ces objets et dans la mesure du possible, connaître et suivre l'évolution de leur état.

La vision permet d'obtenir des informations riches concernant les objets, en

répondant aux questions "où ?" et "quoi ?". L'article [Besl 1985] propose une définition du problème de reconnaissance d'objets 3-D. Pour effectuer cette reconnaissance d'objets, diverses méthodes existent. Ces méthodes sont basées sur des solutions capteurs comme la stéréo vision [Murphy-Chutorian 2005] ou en utilisant un capteur RGBD tel la Kinect [Tang 2012, Han 2013]. Ainsi, le robot est capable de détecter, reconnaître et positionner, moyennant une projection dans le repère global, les objets perçus par rapport au reste de l'environnement.

2.2.3 Robots

Pour comprendre la situation, le robot doit également pouvoir comprendre la situation des agents, à commencer par lui-même. Pour connaître sa situation, dans le cas où l'environnement est connu, le robot doit pouvoir se localiser dans l'environnement. Ceci est essentiel pour lui permettre de naviguer ou de se situer par rapport aux autres éléments. La problématique de construire une carte de l'environnement et de localiser un robot mobile en même temps est un problème ayant été largement étudié en robotique. Ce problème est appelé SLAM (Simultaneous Localization And Mapping). Différentes méthodes existent et font l'objet de chapitre entiers dans des ouvrages tels que [Siciliano 2008, Thrun 2005].

Le robot est généralement composé de plusieurs membres, lui permettant de manipuler des objets, de changer son champ de vision ou de se déplacer. Il lui est donc nécessaire de connaître la configuration de ses propres membres. Cette faculté est appelée proprioception. Sur les robots, des capteurs de position permettent de transmettre la configuration des articulations, et ainsi de connaître la posture globale du robot. Le package ROS "tf" [1] peut par exemple être utilisé pour accéder aux coordonnées des articulations d'un robot, comme illustré par la figure2.3 où les repères des articulations d'un Pr2 sont affichées.

Dans le cas où l'interaction comporte plusieurs robots, différentes solutions sont possible pour connaître la configuration et la position des autres entités robotiques. L'une des méthodes consiste à utiliser la perception. Cette méthode étant coûteuse en calcul et pas toujours fiable, l'échange direct de données est généralement privilégié. En effet, si chaque robot connaît sa position et sa configuration par rapport à un repère commun, il leur est possible de communiquer ces informations à travers le réseau et ainsi permettre à chaque robot de pouvoir accéder directement aux données des autres. Grâce à cela, il est possible à chaque système robotique de connaître la position et la configuration de chaque robot par rapport à l'environnement global.

1. http ://wiki.ros.org/tf

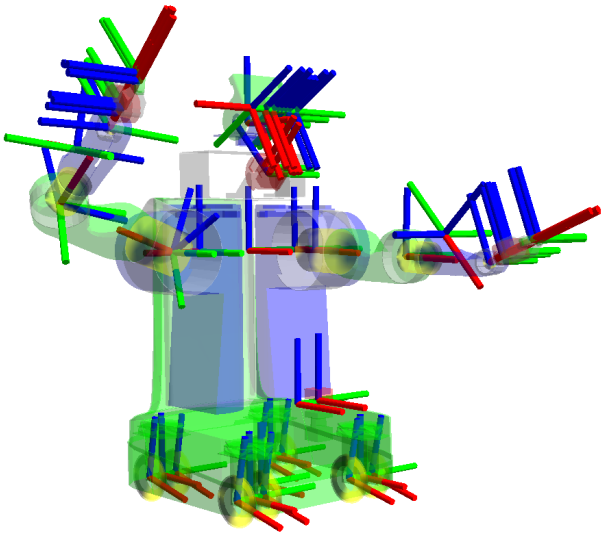


FIGURE 2.3 – Représentation des différents repères articulaires du robot Pr2 provenant de la documentation du package ROS tf.

2.2.4 Humains

Lorsqu'un ou plusieurs humains sont présents dans l'environnement, il est essentiel pour le robot de pouvoir au moins les détecter, voire de les reconnaître. Tout comme le robot, l'homme possède plusieurs membres ! Il est donc important de connaître la position de chaque homme mais aussi sa posture, et donc de percevoir la position de ses membres. Un capteur largement utilisé pour détecter les humains est la Kinect. Elle permet de connaître directement la position des différents membres de chaque humain. La figure 2.4 présente la perception tridimensionnelle de la Kinect et le suivi des membres de chaque humain ainsi que l'estimation de son squelette (de la position de ses membres) calculée à partir de cette perception.

De même, les équipements de capture de mouvements, tel que ceux utilisés dans le cinéma d'animation, peuvent permettre d'obtenir la position et la configuration des membres des humains. Cependant cette dernière solution nécessite aux humains de s'équiper de combinaison ayant des repères visuels pour les caméras de la capture de mouvements (voir figure 2.5).

2.3 Représentation du monde et regroupement des données

La seconde partie de ce premier niveau d'estimation de la situation consiste à établir une "représentation" géométrique du monde qui entoure le robot. Une

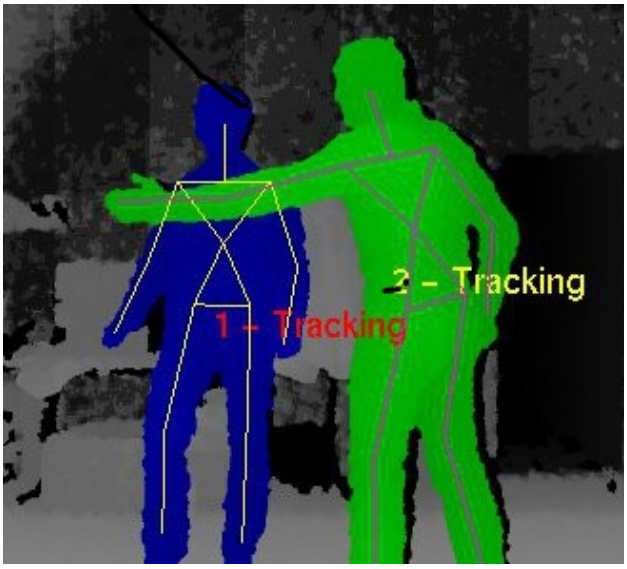


FIGURE 2.4 – Image provenant de la Kinect et illustrant le suivi des humains et de leurs membres réalisé par la librairie OpenNi.

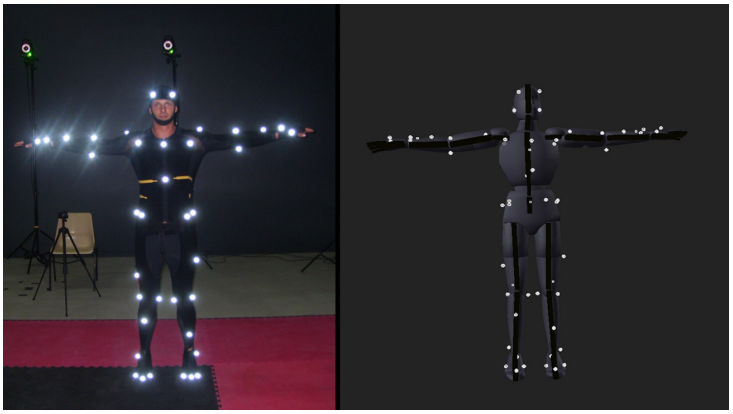


FIGURE 2.5 – Exemple d'équipement de motion capture pour le suivi de l'homme et de sa posture. À gauche un homme équipé et à droite son modèle tridimensionnel.

fois que le système robotique peut accéder aux données de perception issues des capteurs lui permettant d'obtenir la configuration des divers éléments qui composent l'environnement, il est nécessaire, pour en extraire les informations pertinentes, de regrouper, combiner et interpréter les données. Pour ce faire, il faut dans un premier temps regrouper les données de position dans un repère commun afin que le robot puisse avoir une représentation globale de la scène. En effet, selon le capteur utilisé la position des éléments peut se faire dans le repère du capteur, du robot ou directement dans un repère global. La figure 2.6 illustre cette unification avec une représentation en trois dimensions des divers éléments de l'environnement tel qu'ils

sont perçus par le robot.

FIGURE 2.6 – Exemple d'environnement d'interaction homme robot (à gauche), et la représentation tridimensionnelle de cette même scène tel qu'elle est perçue par le robot (à droite). La représentation utilise les données collectées des capteurs pour positionner et orienter les divers éléments dans l'environnement 3d. La représentation faite utilise le nœud de visualisation de ROS : Rviz.

L'unification des données capteur va permettre au système de maintenir un état du monde tridimensionnel comme dans la figure 2.6. Pour améliorer le suivi de l'état du monde notamment au niveau des positions des divers éléments, il est parfois nécessaire d'établir un raisonnement basé sur des hypothèses, par exemple pour éviter de perdre la position d'un objet lorsqu'il n'est plus perçu. Ces raisonnements et hypothèses reposant sur des calculs géométriques et sur l'évaluation de la situation, ils seront présentés plus tard (en section 2.5).

La construction et le maintien du modèle tridimensionnel par agrégation des données provenant des divers capteurs va permettre de calculer des relations entre les différents éléments de l'environnement. D'un point de vue architectural, le module chargé de l'unification des données capteurs est donc le module auprès duquel d'autres modules vont récupérer les données de position des éléments de l'environnement, notamment les modules d'estimation de la situation mais également d'autres tel la planification de mouvement. Il est donc important qu'il définisse les structures de données permettant de représenter l'état du monde géométrique et qu'il les rende accessibles ou les distribue aux autres modules qui en ont besoin pour raisonner sur l'environnement. Ces problématiques relèvent de l'implémentation et seront donc discutées en section 2.7.1.

2.4 Connaissances sur l'environnement et calculs de faits

2.4.1 Représentation de la connaissance

Pour accéder au niveau deux de connaissance de la situation décrit par Ensley, le robot doit pouvoir établir des connexions entre les divers éléments de l'environnement, connaître leur état, leur mouvements... Pour acquérir ces connaissances il est nécessaire au robot de raisonner sur la représentation du monde qu'il a construit et qu'il maintient à jour. Idéalement il doit également être capable de raisonner non seulement sur les relations spatiales et les états des éléments de l'environnement, mais également sur l'évolution de ces éléments (prise en compte de l'aspect temporel). Une fois que ces vérités relatives à l'environnement ont été calculées, le robot doit pouvoir les représenter pour les utiliser lors de la planification de tâche, de la prise de décision ou lors de dialogue avec l'homme.

Nous définissons donc un formalisme pour représenter tous ces types de vérités concernant les divers éléments de l'environnement. Dans notre système, nous représentons l'état de connaissance du robot, aussi appelé état du monde symbolique, par une liste de *faits*. Un *fait* est la représentation d'une connaissance sur l'environnement traduit par une structure de données contenant divers champs. Le fait contient toujours au moins une propriété qui peut décrire l'état d'une entité, un attribut, une relation entre entité, une affordance ou tout autre vérité se rapportant à un élément de l'environnement. Afin de décrire plus en détail cette structure de données essentielles pour notre architecture, nous énumérons ci-dessous les divers champs possibles.

— *Subject*[2] : l'élément de l'environnement sur lequel la propriété s'applique. Il peut s'agir d'un agent (humain ou robotique), d'un objet ou du membre d'un agent (e.g. : *Red_Mug*, *Human1*, *PR2_ROBOT*, *Human1_Right_Hand*).

— *Property* : la propriété attachée au sujet du fait (*Subject*). Il peut s'agir d'une relation entre l'entité-sujet (*Subject*) et l'entité-cible (*Target*), d'un état ou un attribut du sujet ou plus généralement de toute vérité pouvant s'y rapporter. Par exemple les propriétés suivantes sont utilisées dans notre système : *isOn*, *isIn*, *isNextTo*, *isInRoom*, *isInArea*, *isFacing*, *areaDensity*, *canSee*, *canReach*, *distance*.

— *Target* : il se peut que la propriété soit une relation qui lie l'entité-sujet *Subject* avec une entité-cible *Target*. Par exemple, si une entité *RED_BOOK*

2. Pour éviter les confusions avec la catégorie d'éléments matériels faisant parti de l'environnement, le terme de "sujet" a été privilégié à celui "d'objet"

se trouve sur une entité *KITCHEN_TABLE*, *RED_BOOK* sera alors le sujet du fait tandis que *KITCHEN_TABLE* sera l'entité-cible.

— *PropertyType* : ce paramètre définit la catégorie dans laquelle on peut classer la propriété. Il peut s'agir, dans notre système, d'une propriété de position, d'état, de mouvement, de posture ou d'accessibilité. En utilisant ce paramètre, lorsqu'un module extérieur ajoute un fait dont la propriété est inconnue, il reste possible de savoir quel type de propriété est décrite par ce fait. Cela permet également d'appliquer certains traitements aux faits en fonction de leur type. À l'heure actuelle, cette catégorie est utilisée pour le rafraîchissement des données gérées par notre infrastructure logicielle d'estimation de la situation dans la base de données associée.

— *Value* : selon la propriété, il est possible qu'une valeur y soit rattachée. Par exemple, la propriété décrivant l'état d'un container *isFull* ou le déplacement d'une entité *isMoving* peuvent prendre la valeur *TRUE* ou *FALSE*. Un autre exemple, si on représente la distance entre deux membres, comme la main (ou pince) du robot et la tête de l'homme, le paramètre *Value* peut contenir la valeur *DANGER*, *CLOSE* ou *FAR*. Dans certains cas, il est également possible de donner une valeur numérique à ce paramètre. Dans d'autres cas, pour représenter le manque de connaissances sur une propriété et le fait que ce manque est connu, la valeur du fait peut aussi être *unknown*.

— *Confidence* : ce nombre entre 0 et 1 représente la fiabilité du fait. Cette valeur peut être liée à la fiabilité des capteurs ou résulter du calcul de la propriété.

— *Time* : ce paramètre permet d'enregistrer le moment auquel la propriété a été produite (perçue, calculée ou inférée).

— *FactObservability* : ce dernier paramètre représente la probabilité qu'un agent acquière la connaissance du fait s'il est capable de voir le sujet du fait. Par exemple, pour un container, la propriété de contenance "isFull" aura une observabilité directement associée à l'opacité du container. En effet si le container est transparent, en observant celui-ci, l'agent devrait être capable de connaître la valeur de la propriété de contenance. Cette propriété de contenance aura alors une valeur d'observabilité proche de 1. En revanche, si le container est opaque, l'agent qui observera le container ne pourra être informé par cette simple observation visuelle de l'état de la propriété de contenance. La valeur d'observabilité sera donc proche de 0. Pour certaines propriété, l'observation peut être possible ou non ou plus ou moins compliquée. L'utilisation d'une probabilité permet de tenir compte de cette complexité, et, par la suite, de déterminer si la probabilité qu'un agent ait observé un fait est suffisant ou non selon le cas. Plus de détails sur ce paramètre et

son utilisation seront donnés dans le chapitre suivant.

Par exemple, le vecteur $< Subject = Dan, Property = isMovingToward, Target = TABLE\_4, PropertyType = motion, Confidence = 0.8, time = 146962814568, FactObservability = 0.7 >$ représente le fait qu'à un temps *time*, l'agent *Dan* se dirige vers la table du salon (*TABLE_4*). La situation dans laquelle ce fait est produit est illustrée par l'image 2.7. Un autre exemple de fait correspondant à la même illustration représentant le fait que Dan est dans le salon (la zone *Livingroom* en rose) prendra les champs suivants : $< Subject = Dan, Property = isInRoom, Target = Livingroom, PropertyType = position, Confidence = 0.9, time = 146962814568, FactObservability = 0.9 >$

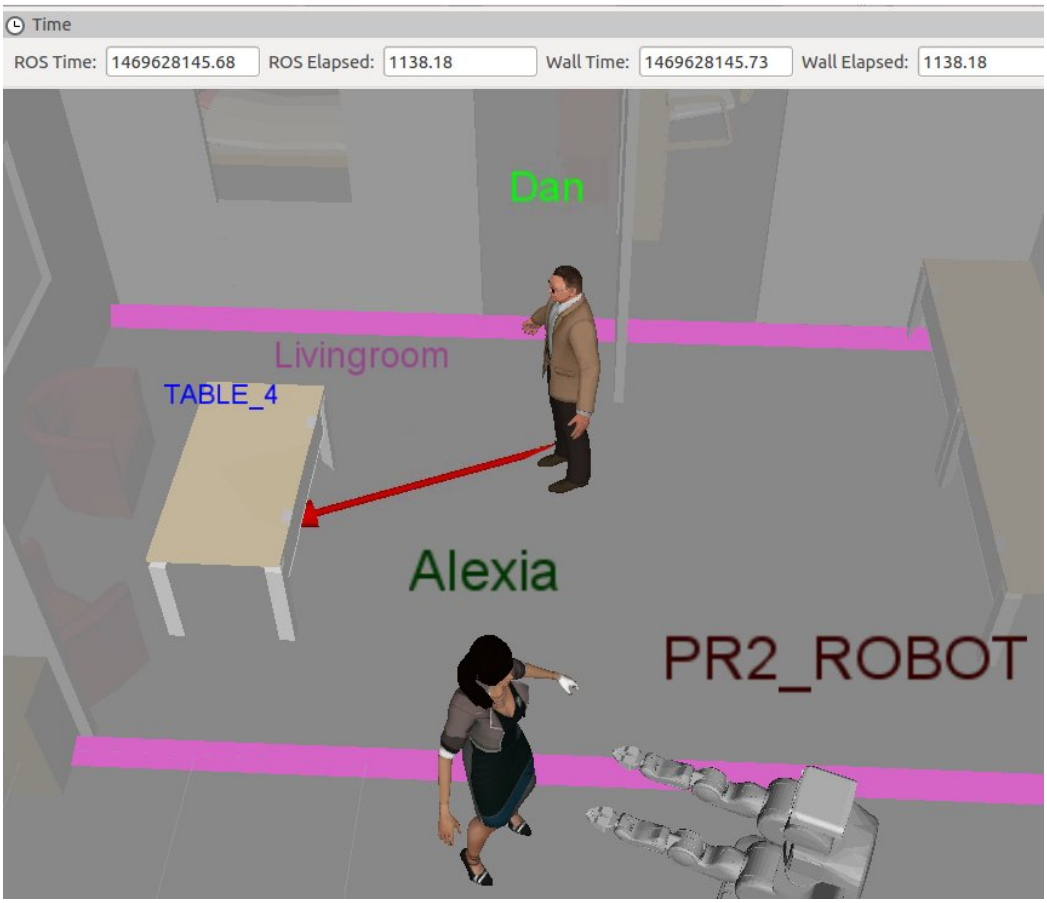


FIGURE 2.7 – Image issue de la visualisation de l'environnement et des faits TOASTER dans le module de visualisation Rviz. Le fait que Dan se déplace vers la table *TABLE_4* est représenté par une flèche rouge.

Les sections suivantes décrivent les raisonnements géométriques et temporels permettant de produire ce genre de faits.

2.4.2 Calculs de faits

Pour acquérir des connaissances sur l'environnement, il ne suffit pas de connaître la position des objets et des divers agents. En effet, il est également nécessaire d'en estimer certaines propriétés permettant de décrire leur état ou le lien qu'il peut exister entre eux. Pour ce faire, nous avons recours à une infrastructure modulaire qui en utilisant les données provenant de la perception va pouvoir calculer différentes propriétés permettant d'avoir une estimation non seulement spatiale mais également conceptuelle de l'environnement.

Nous décrivons dans les sous parties suivantes les divers outils utilisés pour produire les faits permettant de représenter cette estimation de la situation.

2.4.2.1 Utilisation de zones sémantiques

Pour avoir une première estimation de la situation, il est possible dans un premier temps de regarder l'emplacement des divers éléments. Pour ce faire, nous utilisons des zones ayant une signification sémantique particulière pour avoir une première catégorisation de la situation en fonction de la répartition des éléments par rapport à ces zones. Cette première estimation peut ainsi permettre de déclencher d'autres traitements afin d'optimiser les calculs nécessaires. Par exemple il est inutile de calculer si un humain regarde le robot lorsque celui-ci est trop éloigné ou dans une autre pièce.

Nous définissons comme zone un emplacement délimité de l'environnement ayant une signification particulière. Ces zones peuvent être bidimensionnelles ou tridimensionnelles selon l'utilisation qui en est faite. Ces zones peuvent être statiques, c'est à dire liées à une position fixe dans le repère global de l'interaction, ou dynamiques, c'est à dire que la position et l'orientation de la zone est liée à un élément dynamique de l'environnement et évolue donc avec la position et l'orientation de cet élément. Pour avoir une définition générique des zones et une utilisation s'adaptant aux différents cas d'usage possible, ces zones sont paramétrables. Les paramètres permettant de définir ces zones sont :

— le type de la zone : ce paramètre permet de catégoriser la zone. Ainsi une zone peut-être une pièce (par exemple un salon, comme la zone rose *Livingroom* à la figure 2.7), une zone d'interaction avec un agent (par exemple en face de l'agent en question, comme à la figure 2.8), une zone de danger... Ce paramètre permet donc de donner un contexte sémantique à la zone concernée en l'associant à une catégorie. À l'heure actuelle, notre système prends en compte le type "Room" pour lequel il va générer un fait avec la propriété *isInRoom* pour les entités présentes dans la zone et répondant au bon type.

Pour toutes les autres zones, le fait aura la propriété *isInArea*.

— le type d'éléments concernés par cette zone : une zone peut ne concerner que certains éléments de l'environnement. Ce paramètre permet de définir quel type d'éléments doivent être considérés. Il peut s'agir de toutes les entités (objets, et agents), ou simplement des agents ou d'un seul type d'entité (humain, robot ou objet). Les éléments ne faisant pas partie de la catégorie choisie seront ignorés des calculs effectués en lien avec cette zone.

— le type de calcul lié à cette zone : ce paramètre permet de définir quels calculs doivent être faits. Ces calculs seront appliqués aux éléments concernés et se trouvant à l'intérieur de la zone. Dans notre système, ce type peut être *interaction*, au quel cas le fait ayant la propriété *isFacing* sera calculé. Ce fait permet de savoir si les entités présentes dans la zones sont orientées vers le "propriétaire" de la zone. L'autre type est *density*. Pour les zones ayant ce paramètre, on calcule un fait ayant pour propriété *areaDensity* permettant de connaître le pourcentage d'entité du type concerné par la zone étant présent dans cette zone. Il est prévu d'ajouter d'autres types pour permettre notamment d'avoir un calcul conditionnel au niveau des relations spatiales définies dans la section 2.4.2.2.

— le "propriétaire" de cette zone : une zone peut avoir un "propriétaire". Si un propriétaire est défini, la zone est alors une zone dynamique dont la position et l'orientation évoluera avec la position et l'orientation de l'entité désignée comme propriétaire. Par exemple, il est possible de définir une zone d'interaction liée au robot par un trapèze positionné devant celui-ci. Si le robot bouge, la zone bougera avec lui pour toujours rester devant lui.

Pour reprendre l'exemple précédent et pour illustrer les différents paramètres, il est possible de définir une zone d'interaction devant le robot pour savoir si un humain se trouve dans cette zone et activer conditionnellement certains calculs, tel que l'orientation de l'humain pour savoir si celui-ci est dans une configuration permettant l'interaction. Dans cet exemple le robot a pour identifiant *PR2_ROBOT*. La zone aura alors le vecteur de paramètres :

<interaction, humans, orientation, PR2_ROBOT>

Cette zone est illustrée par la figure 2.8 par une zone en jaune. Dans l'exemple illustré, les faits (simplifiés ici) *Alexia isInArea interaction* et *Alexia isFacing PR2_ROBOT* sont créés. Le robot (*PR2_ROBOT*) peut donc en déduire qu'il lui est directement possible de parler à *Alexia* mais pas à *Dan* (il devra par exemple commencer par l'appeler).

Les faits produis par ce module peuvent avoir la propriété *isInRoom* si la zone

a comme type particulier *Room*, pour tous les autres types de zone le fait produit aura la propriété *isInArea*.

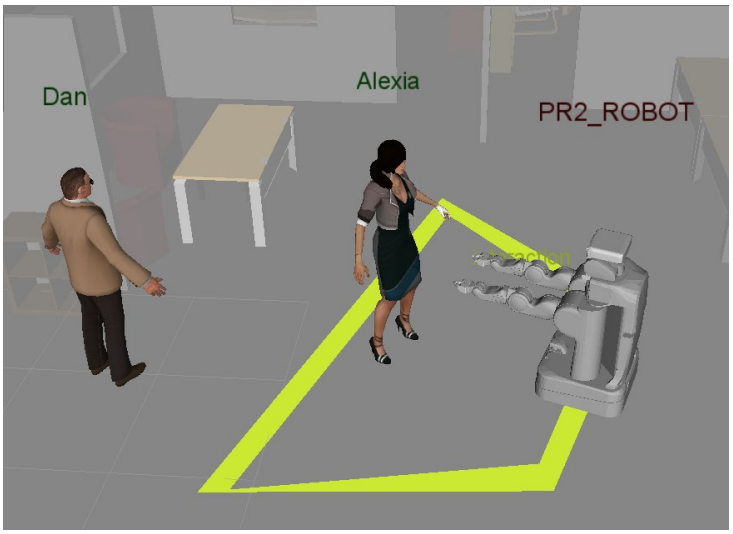


FIGURE 2.8 – Zone liée au robot Pr2 pour savoir si un humain est dans sa "zone d'interaction". Ici Alexia se trouve à l'intérieur de la zone et donc à un emplacement propice pour interagir avec le robot. Le robot va alors calculer si celle-ci lui fait face afin par exemple d'entamer un dialogue.

Ces zones peuvent être créées, mises à jour et supprimées pendant l'interaction (par exemple par le superviseur) et sont utiles pour avoir une première discrimination de la situation et des calculs conditionnels.

2.4.2.2 Agencement des objets

Afin de pouvoir raisonner sur la position des objets il est nécessaire de connaître leur situation par rapport aux autres objets. Par exemple, si le but du robot est de mettre un objet sur une table, il est nécessaire qu'il puisse représenter le fait qu'un objet soit sur un autre (ici la table).

Il s'agit donc de passer d'une représentation numérique des positions à une représentation symbolique. Pour ce faire, un composant de l'infrastructure logicielle du robot calcule les relations spatiales entre les objets. Pour ce faire, ce composant utilise les modèles tridimensionnels des différents éléments stockés en mémoire pour effectuer une représentation tridimensionnelle de l'environnement. Les modèles des différents éléments sont positionnés en utilisant les données de position et d'orientation de ces éléments et leur mise à jour en temps réel pour tenir compte de l'état du monde courant (issu du processus décrit en section 2.3. À partir de cette représentation 3d de l'état du monde, il calcule des propriétés d'agencement spatial pour

décrire la situation des objets.

Ce composant permet de calculer si un objet O1 se trouve au dessus d'un autre objet O2 (*O1 isOn O2*), dans un objet O2 (*O1 isIn O2*) ou à côté d'un objet O2 (*O1 isNextTo O2*) comme illustré par la figure 2.9. Les détails de calculs et la méthode d'implémentation proviennent de travaux précédents et sont décrits dans [Sisbot 2011].

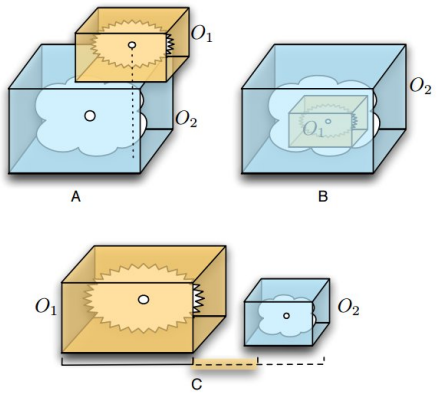


FIGURE 2.9 – Relation spatiale entre deux objets : A) propriété *isOn*, B) propriété *isIn*, et C) propriété *isNextTo*

Grâce à ce composant il est donc possible de décrire la position de chaque objet par rapport aux autres objets de l'environnement en créant des faits avec les propriétés d'empilement (*isOn*), de proximité (*isNextTo*) et d'inclusion (*isIn*).

2.4.2.3 Situation des agents

Dans un contexte d'interaction homme-robot, le robot doit accomplir des tâches en collaborant avec des humains. Il est primordial pour le système robotique d'identifier la configuration de l'humain : est-il en train de bouger ? Est-ce qu'il se déplace vers un objet particulier ? À quelle distance est-il de cet objet ? Pour répondre à ces questions, nous utilisons un composant qui permet de surveiller chaque agent en utilisant des calculs de faits concernant les déplacements, la posture et la distance par rapport à certains points d'intérêt.

Le composant permettant de calculer ces faits enregistre en permanence les positions des entités. Afin de pouvoir stocker ces données temporairement et de les renouveler tout en évitant de surcharger la mémoire, nous avons créé une structure de données appelée buffer circulaire temporel (time stamped circular buffer). La différence avec les buffers circulaires traditionnels est que les données y sont labellisées en fonction d'un marqueur temporel et qu'il est possible d'y accéder en utilisant le

temps qui leur a été attribué. Ceci permet d'accéder à la donnée de position des entités à un temps défini (moyennant de ne pas dépasser la taille du buffer).

Comme le suivi d'un agent peut s'avérer pertinent seulement dans certaines situations, le calcul des faits concernant n'importe quel agent peut être activé ou désactivé par n'importe quel autre module (en utilisant un système de requête). Ceci permet d'éviter de calculer en permanence des faits concernant tout les agents en ciblant seulement les agents qui se trouvent dans une situation nécessitant un suivi particulier. Il est également possible d'activer le calcul de faits concernant un ou plusieurs des membres d'un agent. Lorsque le suivi d'un agent est activé, le composant calcule le déplacement de l'agent à partir des données présentes dans le buffer circulaire associé, afin de déterminer si l'agent est en train de se déplacer. En utilisant le buffer circulaire associé à l'agent et ceux associés aux autres entités, il est possible de calculer si l'agent se déplace en direction d'une entité en particulier (même si cette dernière est également en cours de déplacement) et l'évolution temporelle de la distance entre l'agent et un point d'intérêt. Ce calcul peut être aussi bien fait sur la position globale de l'agent ou sur l'un de ses membres (par exemple sur la main de l'homme). Pour effectuer ce calcul, nous combinons deux méthodes : la première consiste à calculer la trajectoire de l'agent (respectivement de son membre) et de considérer que les entités se trouvant dans cette direction, moyennant une marge d'erreur paramétrable, sont des candidats potentiels. La seconde méthode utilise le différentiel de distance entre l'agent et les entités candidats. Si la distance diminue, l'agent se rapproche de l'entité. En combinant ces deux méthodes de calculs, on peut avoir une liste de candidats avec un coefficient de confiance pour les entités vers lesquelles l'agent se déplace. Ce calcul est évidemment simpliste car il ne prend pas en compte les éventuels obstacles qui pourraient faire qu'un agent fait un détour pour se diriger vers une entité. Cependant il permet dans de nombreux cas d'avoir une idée de l'entité vers laquelle l'agent se déplace.

En plus du mouvement, nous calculons également les distances entre les agents surveillés (respectivement les membres surveillés) et les autres entités. De cette manière il est possible de savoir si un humain, ou l'un de ses membres, est proche d'un point d'intérêt (comme le robot ou un objet avec lequel il veut interagir). Par exemple, ce module peut produire les faits (simplifiés ici) *<HUMAN1 isMoving TRUE>* pour indiquer que l'humain *HUMAN1* est en mouvement ou *<HUMAN1_RIGHT_HAND isMovingToward BLUE_BOOK>* pour indiquer que la main droite de l'humain se dirige en direction du livre bleu.

Le dernier type de fait concerne la posture de l'agent. En utilisant le buffer circulaire il est possible de savoir vers quoi l'humain pointait le doigt à un instant donné. Ceci peut être utile par exemple pour un module de fusion d'un système de

dialogue. Ainsi, si un humain demande au robot "donne-moi ça", si la reconnaissance de parole est capable de fournir le temps pour lequel le mot "ça" a été prononcé, il est possible de demander au module quel(s) objet(s) étai(en)t pointé(s) par l'humain au moment où il a prononcé le mot "ça". Pour alléger la charge de calcul, ce fait est calculé seulement sur demande (et non en permanence comme les autres). Le module utilise la position de la main de l'agent par rapport à son épaule pour calculer la direction de pointage. Tous les éléments se trouvant dans cette direction, à un angle d'erreur prêt, sont renvoyés dans une liste d'entités avec un indice de confiance basé sur l'angle de déviation par rapport à la direction pointée, pour chaque élément de la liste.

2.5 Hypothèses pour le maintien de l'état du monde

2.5.1 Hypothèses de position

Dans leur quotidien, les humains sont occupés par diverses activités (cuisine, nettoyage, bricolage...). Ces activités impliquent très souvent l'utilisation d'objets. Comme les robots sont fait pour assister les hommes dans leurs tâches quotidiennes, il est crucial qu'ils puissent non seulement manipuler les objets, mais également suivre leur changement de position. Notre but ici n'est pas de discuter des différentes méthodes de suivi d'objet basées sur la perception mais d'expliquer comment le raisonnement sur les données de perception peut améliorer la détection et le suivi d'objet.

Localiser et suivre un objet n'est pas une tâche simple. En effet, l'objet peut être de petite taille et par conséquent être fréquemment occulté ou hors du champ visuel. Les humains sont capables de connaître la localisation d'un objet même sans perception directe. Ils font en permanence des hypothèses sur la position des objets qu'ils ne peuvent pas voir et effectuent des raisonnement basés sur ces hypothèses. Pour travailler efficacement avec des humains, le robot doit être capable lui aussi de faire ce genre de raisonnements.

Pour ce faire, nous avons ajouté au composant qui maintient à jour l'état du monde, la possibilité d'émettre des hypothèses de position lorsqu'un objet n'est pas visible.

Comme hypothèse de base, lorsque le robot est dans l'incapacité de voir l'objet (l'objet est hors du champs de vision ou est caché), le système suppose que l'objet est à la même place et a la même orientation que la dernière fois qu'il l'a perçu. Dans la figure 2.10, dans la configuration où le robot perçoit les objets à partir de caméras au niveau de sa tête, le livre bleu étant caché par la boîte rose, il n'est

pas visible par le robot. Cela signifie que le capteur de vision n'est plus capable de fournir la donnée de position de cet objet. Cependant, en utilisant la représentation 3d et des calculs de visibilité, il est possible pour le robot de savoir que l'objet n'est pas visible à sa position courante. Le système de maintien de l'état du monde le conserve tel quel dans le modèle de l'environnement car il sait que l'objet est caché, et qu'il est donc normal qu'il ne soit pas détecté. Cette hypothèse de maintien de position est utilisée comme hypothèse par défaut. Cela signifie que s'il y a une autre hypothèse concernant la position d'un objet, nous choisirons cette dernière plutôt que celle par défaut. Ainsi, si l'on reconnaît une action d'un agent qui consiste à prendre cet objet, l'hypothèse que l'objet est dans la main sera privilégiée.

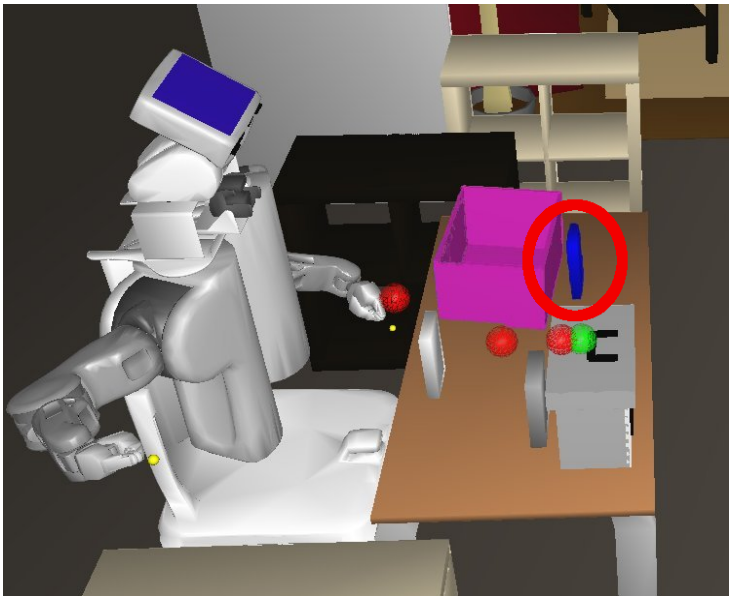


FIGURE 2.10 – Exemple d'État du monde où le système maintient les objets en position même s'ils ne sont pas directement perçus (comme le livre bleu). Ici le robot a une perception uniquement basée sur des caméras placées dans sa tête.

Les autres hypothèses de position sont créées à partir des actions du robot et de l'homme. Si le robot attrape un objet, l'objet sera probablement caché par sa propre main, mais nous savons où l'objet se trouve (dans la main du robot). Par ailleurs, sur certaines plateformes robotiques, ce type d'hypothèse peut être confirmer en utilisant des capteurs d'effort au niveau de la main du robot ou en mesurant l'ouverture après fermeture pour confirmer la présence d'un objet.

Nous avons le même type d'hypothèse pour l'humain, en utilisant les observations sur la situation de l'humain (proximité avec un objet, posture, mouvement) présentées à la section 2.4.2.3, il est possible de suivre les actions de l'homme, et ainsi de savoir quand un humain prend un objet ou le dépose.

En dernier lieu, nous avons également une hypothèse permettant de connaître la position des objets lorsque ceux-ci sont contenus dans d'autres objets. Si un agent laisse tomber un objet dans un autre, même si il n'est pas possible de voir l'objet lâché, il est possible d'approximer sa position (par exemple en le mettant au centre du contenant).

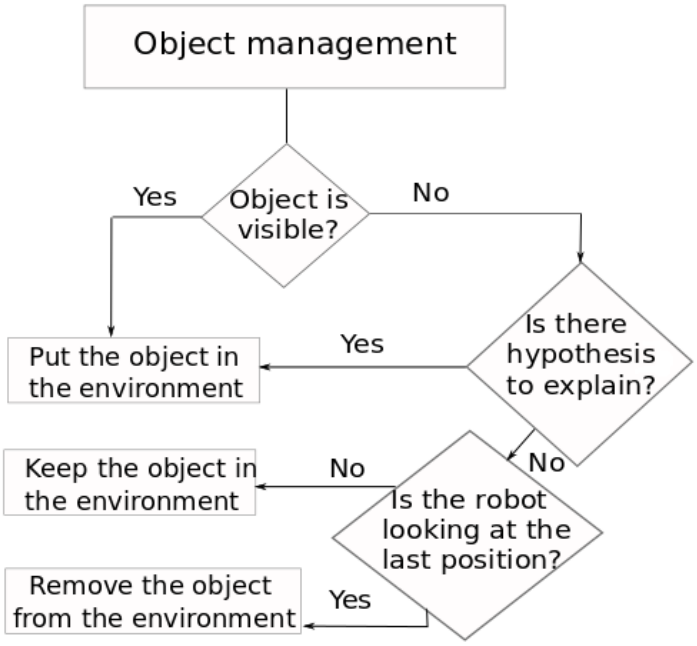


FIGURE 2.11 – Schéma de raisonnement pour gérer la position d'un objet. Les hypothèses sur la position des objets peuvent provenir d'une occlusion, d'un objet dans un conteneur ou dans la main d'un agent.

Pour résumer, le robot utilise en premier lieu la perception, puis s'il n'est pas possible de percevoir l'objet il utilise des hypothèses de position. Si aucun des deux n'est disponible, il utilise la dernière position de l'objet, comme indiqué par la figure 2.11. Les hypothèses de position doivent être gérées en temps réel pour maintenir un modèle du monde cohérent. Dans le cas où un objet a une hypothèse de position et est perçu en même temps, la position perçue est comparée avec celle donnée par l'hypothèse de position. Si la position indiquée est incompatible (distance trop importante) l'hypothèse est alors supprimée car elle est considérée comme impossible. Si un objet devrait pouvoir être perçu et ne l'est pas, alors l'hypothèse par défaut est annulée et la position de l'objet prend alors la valeur "unknown" pour indiquer que le robot ne sait pas où se trouve l'objet en question.

Ces hypothèses concernant la position des objets liés à un raisonnement top-down (symbolique vers sub-symbolique) permettent au robot d'avoir un suivi plus

efficace des objets, spécialement quand ils sont occultés ou non perçus et ainsi avoir une représentation de l'état du monde plus fiable en palliant à certaines limitations de la perception.

2.5.2 Généralisation

Pour aller plus loin, nous travaillons actuellement à généraliser ces hypothèses. En effet, il est important de pouvoir suivre la position des objets, et donc d'établir des hypothèses pour pallier le manque de visibilité et avoir un suivi efficace de la position des objets, mais il est également important de connaître l'état des diverses propriétés liées aux objets. Contrairement aux hypothèses de position, ces hypothèses n'influencent pas la représentation tridimensionnelle de la scène mais uniquement les données symboliques.

Dans l'environnement, nous considérons que seul les agents peuvent agir pour modifier l'état du monde. Ainsi dans cette première approche, nous ne prenons pas en compte l'évolution temporelle de certaines propriétés (comme par exemple le fait qu'une boisson chaude refroidisse). Le maintien des hypothèses concernant les faits symboliques décrivant l'état du monde est donc effectué à partir du suivi des agents et de l'estimation de leurs actions. Nous utilisons pour cela un module de reconnaissance d'actions appelé AIR (Action and Intention Recognizer), dont le fonctionnement est présenté au chapitre 5. Ce module transmet au module de gestion d'hypothèses (Hypotheses Manager) les actions détectées par le système. De nombreux travaux traitent de la question de la mise à jour du monde à partir de la reconnaissance d'actions [Ginsberg 1988a, Ginsberg 1988b, Rao 1989, Jackson 1990, Brewka 1993]. Ainsi, dans l'infrastructure de représentation de connaissances KnowRob, des *règles de projection* sont utilisées pour connaître l'état du monde après qu'une action se soit réalisée [Tenorth 2015]. Les actions peuvent changer la position des objets mais également le détruire ou créer un nouvel objet en combinant deux autres (par exemple en mélangeant deux ingrédients). Ici, nous traitons uniquement des actions pouvant modifier la position (comme présenté précédemment) ou l'état d'un objet, comme par exemple le fait de remplir ou vider un container, de nettoyer un objet ou encore de l'allumer ou l'éteindre. L'idée est d'associer à chaque action un effet (se traduisant par la modifications de certains faits symboliques) dont l'action est la cause. Cet effet est aussi appelé post-condition. Basé sur les posts-conditions de l'action reconnue, le module de gestion d'hypothèses est capable de mettre à jour certains faits liés à l'état symbolique du monde pour prendre en compte l'effet de l'action reconnue.

Par exemple, si le module AIR détecte qu'un humain a versé le contenu C

d'un récipient B (comme une bouteille) dans un autre récipient V (comme un verre), le rôle du module de gestion des hypothèses sera d'envoyer une requête à la base de données pour mettre à jour l'état des propriétés liées à V et B. Ainsi, la propriété *isEmpty* sera mise à faux pour V et la propriété *hasLiquid* sera ajoutée avec C comme valeur (C peut être par exemple de l'eau). Le module de gestion d'hypothèses sera également chargé de décrémenter la quantité de C contenue dans B. Pour résumer, ce module de gestion d'hypothèses permet d'appliquer les effets (ou postconditions) liés à une action lorsque celle-ci est détectée. Il sera également en charge d'assurer la cohérence globale de l'état du monde.

Le schéma 2.12 présente l'architecture permettant cette gestion d'hypothèses. Le module de gestion d'hypothèses peut communiquer avec le module d'agrégation de données et de maintien du monde (Perceived Data Gathering) pour modifier la position d'un objet, ou avec la base de données temporelle (présentée à la section suivante) pour modifier certains faits, et donc l'état du monde symbolique.

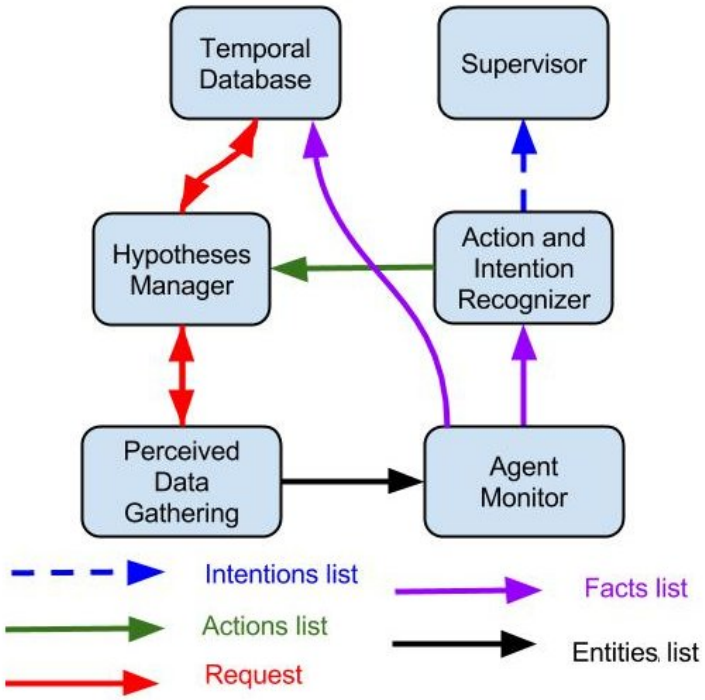


FIGURE 2.12 – Schéma de l'architecture prévisionnelle pour le module de gestion des hypothèses.

2.6 Base de données temporelle

Nous avons présenté précédemment comment divers composants contribuent à maintenir un état du monde géométrique (sub-symbolique). Nous avons également présenté comment d'autres composants, à partir de l'état du monde géométrique et de raisonnements et calculs sur celui-ci, produisent une liste de faits permettant une représentation symbolique de l'état du monde et une estimation de la situation.

Afin de regrouper ces données symboliques et permettre un accès standardisé et efficace, nous avons ajouté une base de données SQL à notre système.

2.6.1 Gestion des données

La base de données peut gérer deux types de données : les données statiques et dynamiques. En effet, il est possible de connaître à priori certaines informations sur l'environnement tel que la couleur, le nom de certains objets ou leur type ou leur propriétaire. Nous avons donc dans notre base de données une première table permettant d'avoir des informations sur les diverses entités de l'environnement. Cette table peut être complétée en ligne si le système reçoit des informations complémentaires durant l'interaction (si le robot découvre de nouvelles propriétés statiques ou de nouveaux objets, que se soit par la perception ou le dialogue).

Concernant les données symboliques et notamment les *faits* produits par les divers composants d'évaluation de la situation, la base de données récupère en permanence la liste de faits produite et publiée par chacun des modules. Cela permet de vérifier les changements apparus dans l'environnement pour mettre à jour la table de représentation symbolique de l'état du monde tel qu'il est perçu par le robot (*ROBOT_WS_TBL* pour Robot World State Table). Cette table est composée de faits qui sont considérés comme vrais et mis à jour en temps réel. Ces faits proviennent des divers modules d'évaluation de la situation et ont comme valeur temporelle la date où le fait a été détecté.

2.6.2 Gestion de l'historique

L'une des valeurs ajoutées de cette base de données est l'aspect temporel. Ceci permet de doter le système d'une mémoire des faits passés. Dans [Vernon 2014], Vernon décrit la mémoire des systèmes cognitifs comme un moyen de "faciliter la persistance des connaissances et forme un réservoir d'expérience". Ainsi, certains systèmes de gestion de connaissances intègrent des mécanisme de mémoire et de raisonnement sur la mémoire, tel KnowRob [Tenorth 2015] ou encore Open-Ease [Beetz 2015]. Open-Ease est une base de connaissance en ligne fournissant des don-

nées sur les activités des humains et des robots. Open-Ease fournit également un langage pour effectuer des requêtes sur la base de données ainsi qu'un système d'inférences. Cela permet au robot de répondre aux questions sur leur propres actions en fournissant les raisons et en décrivant les différentes étapes.

Dans notre implémentation, la mémoire est principalement réalisée en terme de sauvegarde d'état précédent des propriétés et des événements s'étant produits dans l'environnement. En effet, lors d'une interaction homme-robot, de nombreux événements peuvent se succéder et pour pouvoir comprendre et interpréter les actions et les élocutions de l'homme, il est important de considérer l'historique de l'interaction.

Nous avons donc ajouté une gestion de l'aspect temporel. Pour ce faire, lorsqu'un fait est considéré comme vrai à partir d'un temps t1 et donc présent dans la table de représentation symbolique de l'état du monde perçu par le robot, le temps t1 lui est attribué jusqu'à ce que ce fait soit considéré comme faux à un temps t2.

Lorsque le fait est considéré comme faux en t2, ou lorsqu'il change de valeur, il est retiré de la table *ROBOT_WS_TBL* pour être mis dans une table mémoire *ROBOT_MEM_TBL*. Cette table contient également une liste de faits. À la différence de la table *ROBOT_WS_TBL*, les faits de cette table peuvent être en double (deux faits identiques peuvent être présents dans la table si ils ont des temps différents) et représentent des faits passés. Pour ce qui est du temps associé, chaque fait a deux dates : la date t1 à laquelle le fait a commencé a être vrai (ou plutôt détecté comme étant vrai) et la date t2 à laquelle le fait a cessé d'être vrai (détecté comme étant faux). Il est donc possible, à l'aide d'une requête SQL adéquate, de demander la liste de faits étant vrais à un instant t donné (ce qui correspond à tous les faits dont $t1 < t$ et $t < t2$). Pour avoir un accès direct aux transitions, nous avons également ajouté une table d'événements *EVENT_TBL*. Lorsqu'une transition de fait est détectée (un fait n'est plus vrai ou change de valeur), en plus de le retirer de la table de faits courant et de l'ajouter à la table mémoire, nous ajoutons la transition avec la date t2.

Pour illustrer, nous prenons l'exemple où un homme H prend un objet O sur une table T à un moment t2. Trois opérations seront faites dans la base de données :

- Le fait O *isOn* T sera déplacé de la table *ROBOT_WS_TBL* vers la table *ROBOT_MEM_TBL*, avec t2 comme temps de fin.
- Le fait H *hasInHand* O sera ajouté à la table *ROBOT_WS_TBL* avec un temps t2.
- L'événement H *picks* O sera ajouté à la table *EVENT_TBL*.

Cette gestion du temps, des événements et de la mémoire permet de pouvoir réaliser des raisonnements temporels complexes sur les données symboliques et les

liens temporels entre les différents faits. L'une des améliorations à venir est de permettre "d'oublier" les événements en fonction de leur date et de leur ancienneté pour éviter de saturer la mémoire dans le cas d'une utilisation longue durée.

2.7 Implémentation

Les composants décrits précédemment ont été implémentés dans une infrastructure logicielle open-source appelée TOASTER [3] (Tracking Of Agents and Spatio-TEmporal Reasonning).

L'un des avantages de TOASTER repose sur sa généricité et son adaptabilité grâce à une conception modulaire qui permet également d'étendre facilement l'infrastructure. En effet, comme présenté en section 2.2, les données d'environnement peuvent parvenir de divers capteurs.

2.7.1 Collecte de données et généricité

Pour que l'infrastructure logicielle soit utilisable avec n'importe quelle configuration de capteurs, un premier module permet de collecter toutes les données et de les mettre au format TOASTER. Tous les autres modules utiliseront ensuite l'état du monde au format exporté par ce module. Ce module chargé de collecter les données des divers capteurs est appelé *PDG* pour Perceived Data Gathering. Il permet à l'heure actuelle de gérer quatre types d'entrées pour l'homme, deux pour les objets, deux modèles de robot. Il gère également, à l'heure actuelle, la lecture de deux types de messages : les topics ROS [4] et les posters pocolibs [5].

Un autre avantage de ce module est également sa simplicité d'extension. Ainsi, grâce au système d'héritage du C++ il est possible d'ajouter une entrée capteur pour détecter des objets, pour suivre l'homme ou un modèle de robot en seulement 50 à 200 lignes de code C++ (estimation basée sur les entrées déjà existantes). Au démarrage du module ou à tout moment de l'interaction, il est possible de (re)configurer se module pour modifier les entrées à prendre en compte.

Durant la phase de développement de nouvelles fonctionnalités il est coûteux d'un point de vue matériel tout comme d'un point de vue temporel de faire des tests dans le monde réel. Pour permettre de tester rapidement et sans monopoliser de plateforme robotique, le module *PDG* peut prendre en entrée des données de simulateur robotique tel que Gazebo [6] pour la configuration du robot,

3. TOASTER est disponible sur github à l'adresse https ://github.com/Greg8978/toaster
4. http ://www.ros.org/
5. https ://www.openrobots.org/wiki/pocolibs
6. http ://wiki.ros.org/gazebo

MORSE[Echeverria 2011] pour la configuration de l'homme, du robot et la vision des objets. En plus de ces simulateurs, un module intégré à TOASTER pour créer et tester des environnements d'interaction homme robot appelé *toaster_simu* a été créé. *PDG* permet également d'avoir une configuration hybride. Par exemple, il est possible d'utiliser les capteurs d'un robot réel ainsi que la détection d'objets dans le monde réel et de simuler le déplacement d'un humain (avec l'interface clavier). Le module *PDG* permet donc une grande flexibilité dans la configuration des données d'entrées. En exportant l'état du monde dans un format unique et invariable, *PDG* permet aux autres modules TOASTER de s'abstraire de cette diversité et de conserver les mêmes formes de raisonnement, indépendamment de la configuration des capteurs et des données d'entrées de TOASTER.

2.7.2 Outils de développement

Pour aider au développement de nouvelles fonctionnalités et à la configuration de scénarios, un module de visualisation a été ajouté à l'infrastructure TOASTER. Ce module utilise le système de visualisation intégré à ROS, Rviz [7] pour afficher les différents éléments en trois dimensions, ce qui permet de comprendre la situation en affichant l'état du monde. Chaque entité est associée à un modèle 3D qui est positionné dans un environnement virtuel à la position et à l'orientation données par le module PDG. Le module affiche également les noms de chaque entité au dessus de leur modèle 3D. Le module permet également d'afficher les différentes zones présentes dans le module de gestion de zone (*area_manager*) avec leurs noms, comme présentée sur la figure 2.8. Concernant les faits exportés par le module de suivi d'agents (*agent_monitoring*), afin d'éviter la surcharge de l'interface, le fait *isMoving* rend l'affichage du nom de l'agent plus intense selon si le système considère qu'il bouge. La donnée numérique de sa vitesse associée à ce fait permet également de faire varier cette intensité. Pour le fait *isMovingToward*, une flèche reliant l'agent à l'entité vers laquelle il se déplace est tracée. L'intensité de la couleur de cette flèche varie également en fonction de l'indice de confiance du fait associé. Ce retour visuel est illustré par un exemple à la figure 2.13

Cette visualisation permet de comprendre l'état du monde tel qu'il est perçu par le robot et permet donc de faciliter les explications liées aux agissements du robot, mais aussi de faciliter la configuration d'une nouvelle expérimentation et la recherche d'erreurs lors de l'implémentation de nouvelles fonctionnalités.

Un autre composant déjà évoqué dans la partie 2.7.1, permettant de rapidement paramétrer et tester des expérimentations est le module de test *toaster_simu*. Grâce

7. http ://wiki.ros.org/rviz

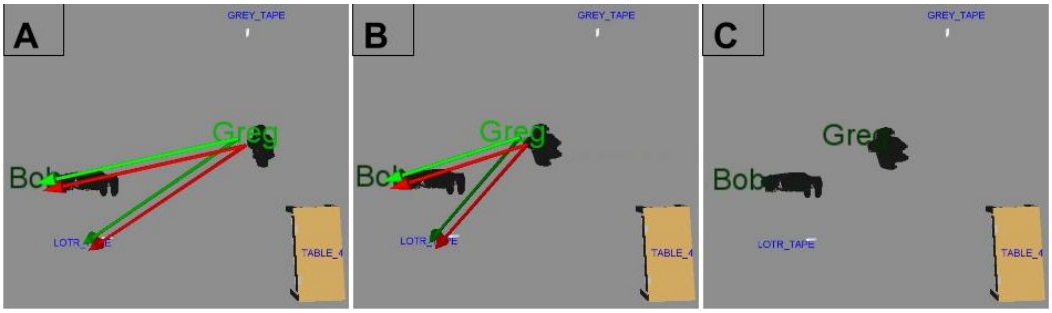


FIGURE 2.13 – Affichage dans Rviz des faits liés au module *agent_monitoring*. La flèche rouge indique ce fait calculé en terme de différentiel de distance alors que la flèche verte indique ce même fait en terme de direction de trajectoire. Sur l'image A et B *Greg* se déplace vers *Bob* ainsi que vers un objet (*LOTR_TAPE*). Sur l'image A et de façon plus prononcée sur l'image B, les flèches allant vers *LOTR_TAPE* sont plus sombres, indiquant que la confidence pour ce candidat est moins grande que pour *Bob*. En C Greg s'est arrêté. Les flèches de directions sémantiques ont disparues et le nom de Greg s'est assombri (pour devenir comme celui de Bob qui est immobile).

à ce module, il est possible par une simple requête d'ajouter et positionner des entités dans l'environnement. Il est également possible de contrôler ces entités par une interface clavier. Le module exportera alors en temps réel les données liées à l'état du monde créé par l'utilisateur et ces données seront lues par le module PDG. Il est donc possible, à partir d'un simple script envoyant quelques requêtes, de configurer un environnement d'interaction avec des robots, des humains et des objets. Des scripts Python pour configurer un environnement de simulations sont disponibles à titre d'exemple [8].

2.7.3 Architecture globale

Les autres modules compris dans l'infrastructure logicielle TOASTER ont une implémentation qui suit les descriptions données précédemment. Ainsi, le module *area_manager* permet d'ajouter et de supprimer des zones dans l'environnement à l'aide de requêtes. Le modèle et les raisonnements développés dans ce module suivent la description faite en 2.4.2.1. Pour le calcul de l'agencement des différents éléments, un module nommé *SPAR* (pour SPAtial Reasonning) est utilisé. Enfin, pour le suivi des agents décrit en 2.4.2.3, le module *agent_monitoring* a été développé. Concernant la base de données temporelle, nous avons conçu celle-ci en utilisant la bibliothèque SQLite [9]. Ce choix a été fait car cette base de données est très répandue,

8. https ://github.com/Greg8978/toaster-scripts
9. https ://www.sqlite.org/

respecte les conventions SQL et permet une gestion in-memory des données qui augmente les performances en terme de vitesse d'accès à la donnée.

Chacun de ces modules peut être démarré et arrêté à n'importe quel moment de l'interaction. Il peut également recevoir des requêtes ou des messages provenant d'autres composants mais ne dépend pas directement des autres. Par exemple, il est possible d'utiliser le module de raisonnement sur les zones (*area_manager*) sans *PDG*. Cependant il est nécessaire, si l'on veut que les calculs sur la présence d'entités dans les zones soit effectif, que le module *area_manager* reçoive les données correspondantes. De même, la base de données temporelle a été conçue pour pouvoir lire des listes de faits en entrée mais il est possible de paramétrer celle-ci afin de définir quelles listes de faits doivent être lues. Ceci permet de facilement remplacer un module de l'architecture sans impacter les autres modules, dans la mesure où ce nouveau module respecte les conventions de formats de données d'entrée et sortie.

Pour aider à la compréhension, nous proposons le schéma d'architecture 2.14 qui présente comment les données des différents composants peuvent s'interfacer.

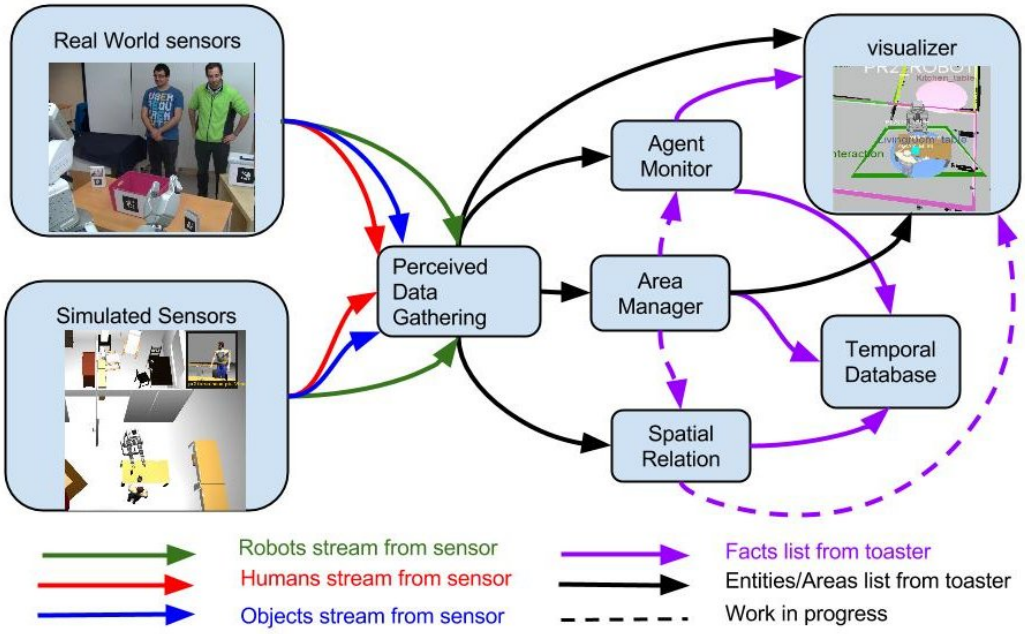


FIGURE 2.14 – Architecture de l'infrastructure logicielle TOASTER avec les différents flux d'échanges de données

2.8 Résultats expérimentaux

Pour illustrer l'utilisation possible de TOASTER, nous présentons ici deux expérimentations utilisant l'infrastructure logicielle.

2.8.1 Le robot guide

L'une des premières applications de TOASTER a été de l'utiliser comme composant d'estimation de la situation pour un robot guide adaptatif, proactif et prenant en compte les humains [Fiore 2015].

Pour cette application, le robot doit guider un groupe d'utilisateurs à travers une zone de transit (potentiellement bondé), pour les amener à l'endroit choisi. L'un des défis de ces travaux était de guider les personnes de façon socialement acceptable et en prenant en compte le confort des usagers. Le robot adapte sa vitesse au groupe, afin de tous les amener à leur destination avec une allure qui satisfasse la plupart des usagers et évite que certains utilisateurs n'abandonnent.

Pour atteindre ces objectifs, nous avons utilisé : (1) le module PDG pour regrouper les données capteurs afin de créer et maintenir le modèle de l'état du monde ; (2) les faits produits par le module de gestion des zones (area_manager) ; (3) les faits produits par le module de suivi des agents (agent_monitoring), pour comprendre la situation du groupe.

Concernant le module de gestion des zones, deux types de zones ont été crées : les zones de guidage et les zones d'activité. Les premières sont liées au robot et évoluent donc avec sa position et son orientation. Elles sont utilisées pour évaluer la configuration des utilisateurs par rapport au robot guide, ce qui lui permet d'adapter sa vitesse en fonction de la configuration du groupe. L'idée est de créer trois zones de guidage autour du robot pour savoir si la configuration des utilisateurs indique qu'ils souhaitent que le robot aille plus vite ou si l'un d'entre eux a du mal à suivre. Les trois zones sont appelées *slow*, *following* et *pushing* et sont configurées comme indiqué dans la figure 2.16. L'idée est que, (1) si l'un des membres du groupe est dans la zone *slow* (loin du robot) le robot *ralentit*. (2) Si 1) est faux et la majorité du groupe est dans la zone *pushing* (proche ou sur les côtés du robot) le robot *accélère*. (3) Si 1) et 2) sont fausses, signifiant qu'à la fois aucun humain ne se trouve dans la zone *slow* et que la majorité des humains se trouvent dans la zone *following*, le robot *continue* avec la même allure.

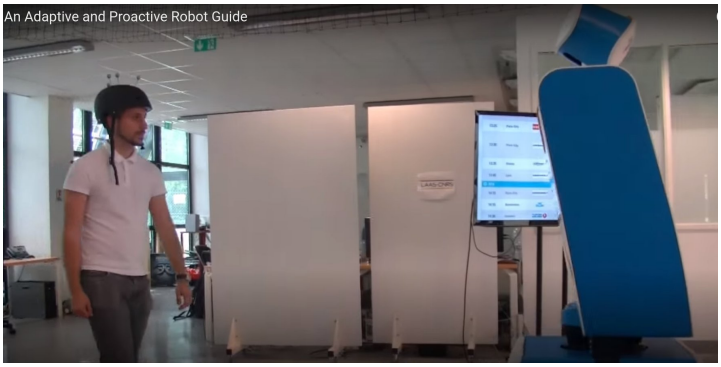
Le deuxième type de zones est statique, lié à une position de l'environnement, et déclenche des calculs sur les humains s'y trouvant. Ces calculs concernent les distances, le mouvement et l'orientation par rapport à un point d'intérêt (comme par exemple un écran). En utilisant ces zones, le système peut détecter l'activité humaine (par exemple que l'humain regarde un écran d'information, comme dans l'illustration à la figure 2.15, ou l'humain se dirige vers les toilettes). Cette estimation de la situation donne au robot la capacité de détecter quand un ou plusieurs des humains du groupe guidé, est investi dans une activité temporaire. Le superviseur

peut ensuite distinguer entre les situations où il devrait abandonner le guidage de certains usagers de situations où il devrait attendre les humains, voir même les aider de façon pro-active (par exemple proposer des informations s'il détecte qu'un humain regarde un panneau d'informations). Une vidéo de présentation de ces travaux est disponible à l'adresse : `https://youtu.be/9lI1_Qecjmc`.

2.8.2 Le robot coéquipier

L'un des autres projets utilisant l'infrastructure logicielle TOASTER visait à construire un robot pour travailler avec un humain sur une chaîne d'assemblage. Dans ce scénario, le robot partage l'espace de travail avec un humain. Par conséquent, il est essentiel de pouvoir estimer à tout moment la situation spatiale de l'homme et son activité. Le système se repose sur le module *PDG* de TOASTER pour regrouper les données depuis les capteurs et construire le modèle d'environnement. Il utilise également le module *SPAR* pour avoir une représentation symbolique de l'état du monde, le module de gestion des zones pour savoir à quelle station de travail l'humain se trouve et le module de suivi des agents pour estimer l'action courante de l'homme. En utilisant les faits provenant de ces modules, lui procurant une représentation symbolique de l'état du monde et de la situation de l'interaction, le robot est capable d'adapter son comportement à l'humain avec lequel il collabore. La première façon d'utiliser les informations de TOASTER est d'assurer la sécurité en arrêtant le déplacement du robot lorsque l'homme est dans la même zone de travail et lorsqu'il est en train de se déplacer. Un autre moyen de s'adapter est de transmettre la représentation symbolique de l'état du monde (l'ensemble des faits) à un planificateur afin d'aider l'homme de façon proactive. En effet, avec la connaissance de l'état du monde actuel, reliée à la compréhension de l'activité de l'homme, le robot peut estimer l'étape actuelle du plan à accomplir et agir afin d'aider l'homme pour les étapes suivantes du plan. Par exemple, lorsque l'homme est en train de nettoyer une pièce à assembler, si l'étape suivante est de visser cette pièce, le robot peut apporter la visseuse à l'homme. L'image 2.17 illustre la situation où le robot détecte que l'homme tend le bras et est orienté en direction du robot. En utilisant ces faits, le robot suppose que l'homme est prêt à recevoir l'objet et le lui donne.

Ces deux exemples d'utilisation de l'infrastructure logicielle dans deux projets différents illustrent la généricité, la simplicité de configuration grâce à l'architecture modulaire, ainsi que la polyvalence de TOASTER. Dans ces deux projets d'interaction homme robot, les faits produits par TOASTER ont permis de fournir les informations nécessaires à la couche décisionnelle pour pouvoir s'adapter à l'homme



(a) Le robot guide l'homme.

(b) L'homme s'arrête pour consulter un écran d'informations

(c) Le robot donne des informations complémentaire de façon proactive.

FIGURE 2.15 – Dans cet exemple, le robot qui doit guider l'humain estime que celui-ci est pris dans une tâche secondaire temporaire (ici consulter un écran d'informations). Plutôt que d'abandonner la tâche, le robot décide alors de s'approcher de l'homme pour lui donner des informations complémentaires avant de reprendre la tâche de guide.

et donner un comportement socialement acceptable au robot.

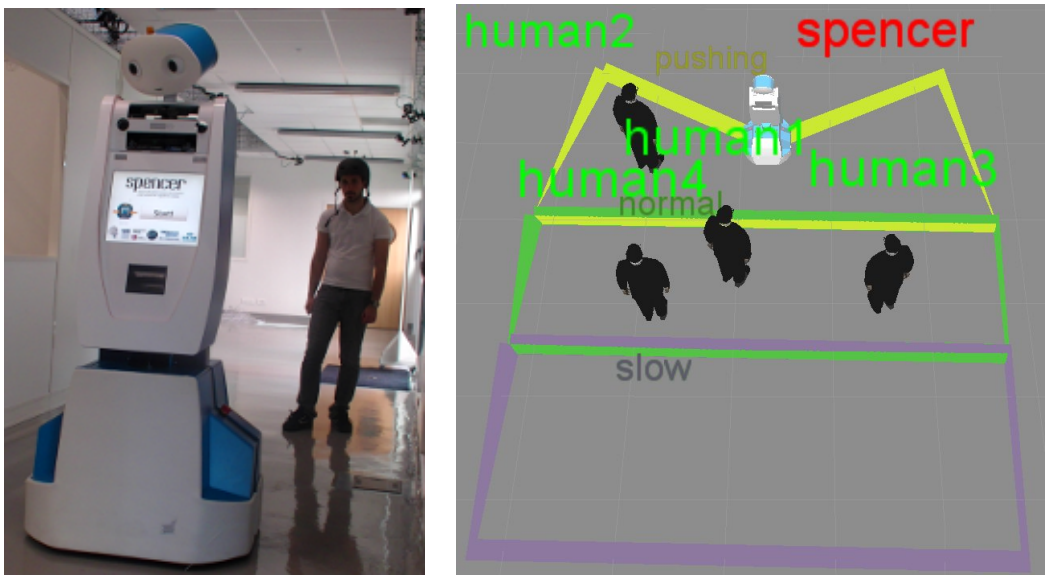


FIGURE 2.16 – Le robot Spencer, dans le monde réel (à gauche) et dans TOASTER (à droite) avec les zones qui lui sont attachées pour guider les humains.

FIGURE 2.17 – Le robot détecte que l'homme demande l'objet en utilisant la distance de la main par rapport à son corps, grâce au module de suivi d'agent de TOASTER, pour comprendre que l'homme est en train de tendre le bras. Ici, un système de capture de mouvement est utilisé pour suivre les positions de la tête et du bras de l'homme.

Chapitre 3

Prise de perspective et maintien de l'état mental des agents

Sommaire

3.1 Motivation

Comme présenté dans le chapitre précédent, il est primordial que le robot comprenne son environnement pour pouvoir agir. Il doit avoir connaissance de l'état du monde en utilisant la perception qu'il en a à travers ses capteurs et utiliser cette perception pour en extraire, à l'aide de divers raisonnements, une estimation de la situation. Cependant, dans le cas où le robot agit en collaboration, ou tout du moins en présence d'humains, il est nécessaire que le robot ne considère pas ces derniers comme de simples obstacles ou uniquement comme des systèmes agissant sur l'environnement. En effet, lorsque l'homme est présent, pour mesurer la qualité

de l'interaction, il est important de considérer non seulement le succès et l'optimalité de l'exécution, mais également la satisfaction et le confort des humains présents dans l'environnement ou impliqués dans la tâche. L'homme étant une créature sociale, le robot doit pouvoir faire preuve de capacité d'interprétation de la situation de l'homme pour le comprendre et exhiber des comportements "sociaux" pour être compris et accepté par les individus avec lesquels il doit interagir. Ainsi il est primordial que le robot comprenne son environnement et sa situation mais également celle des humains présents dans cet environnement. Nous avons déjà présenté au chapitre précédent, comment certains composants du système d'évaluation de situation permettent d'obtenir des informations sur l'homme, notamment en 2.4.2.3. Ici nous allons aller plus loin en considérant non seulement la situation spatiale de l'homme mais aussi une estimation de son état mental.

3.2 Théorie de l'esprit

3.2.1 Littérature en psychologie

Afin de savoir comment le robot doit interagir avec l'homme, il est important d'étudier tout d'abord comment les hommes interagissent entre eux. La psychologie est la discipline étudiant les comportements humains. L'un des domaines de cette discipline, appelé psychologie du développement, a pour but de comprendre comment et pourquoi l'humain se développe. Cela concerne aussi bien l'étude des processus mentaux, des comportements, des performances que l'évolution des habiletés au cours de la vie humaine. La psychologie du développement s'attache à caractériser la théorie de l'esprit (Theory of mind en anglais, ou ToM). La théorie de l'esprit désigne la capacité qu'a un individu d'attribuer un état mental (en terme de pensées, ressentis, désirs, motivations et intentions) aux autres avec lesquels il interagit. La théorie de l'esprit inclut la notion de prise de perspective. Cette capacité humaine permet à un individu de raisonner en prenant le point de vue d'un autre.

Étudiée dans la littérature en psychologie[Flavell 1992, Tversky 1999], cette habileté humaine est cruciale pour interagir avec autrui en permettant de raisonner sur ce que l'autre comprend du monde en terme de perception visuelle, de description spatiale, d'affordances et de croyances. Des études menées sur des individus ne possédant pas les mécanismes cognitifs nécessaires pour la prise de perspective, comme les jeunes enfants ou les personnes autistes [Frick 2014], ont permis de mettre en évidence les difficultés que ces personnes ont dans leurs relations sociales quotidiennes, ce qui confirme l'importance de cette capacité pour interagir convenablement avec

d'autres humains.

Flavell dans [Flavell 1977] décrit deux niveaux de prise de perspective. Il distingue : le niveau un étant la prise de perspective perceptuelle et le niveau deux, la prise de perspective conceptuelle. La prise de perspective perceptuelle désigne la capacité d'un humain à comprendre que les autres ont une perception différente du monde, illustré par l'image 3.1.

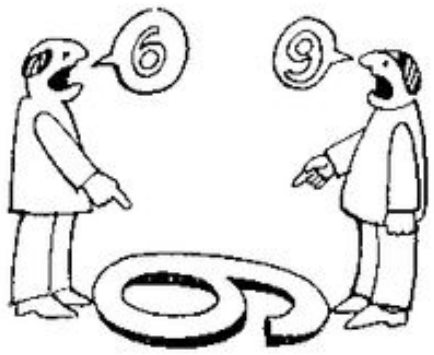


FIGURE 3.1 – Illustration du fait que deux individus peuvent avoir une interprétation divergente de l'environnement en fonction de leur situation spatiale. La capacité de comprendre que l'autre a une vision différente est liée à la prise de perspective perceptuelle (ou prise de perspective de niveau un).

La prise de perspective conceptuelle va plus loin et désigne la capacité d'un humain à attribuer des croyances et des sentiments aux autres[Baron-Cohen S 1985]. Ainsi, dans l'illustration faite dans la figure 3.2, Bob fait une supposition sur l'état de croyance d'Alice concernant la contenance de la boîte. Cette supposition que Bob fait de l'état de croyance d'Alice peut différer de l'état réel.

Ainsi, dans la psychologie du développement, la faculté de comprendre qu'un autre puisse avoir une croyance erronée sur l'environnement qui l'entoure est considérée comme une étape importante dans l'évolution et l'acquisition de la théorie de l'esprit. Dans la littérature en psychologie, la tâche de reconnaissance de fausse croyance ("false belief task" en anglais) a été formulée dans [Wimmer 1983]. Heinz Wimmer et Josef Perner définissent un test (le test de Sally et Anne) qui permet d'évaluer les aptitudes d'une personne à comprendre qu'autrui possède des états mentaux différents des siens. Le test a été par la suite réalisé par Simon Baron-Cohen et Alan M. Leslie et rapporté dans [Baron-Cohen S 1985] et une seconde version dans [Leslie 1988]. Ce test est expliqué dans la figure 3.3.

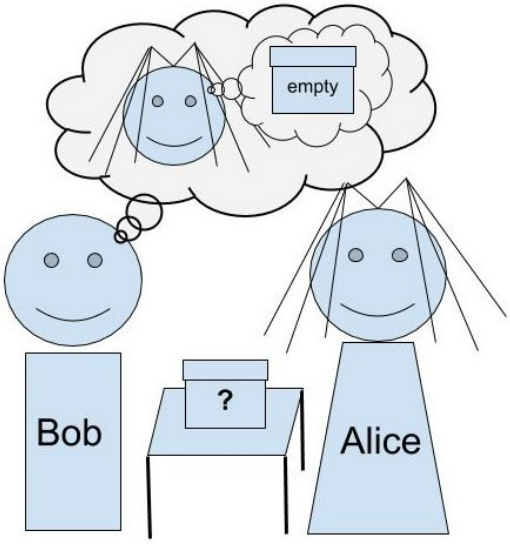


FIGURE 3.2 – Illustration de la prise de perspective conceptuelle. Ici Bob attribue une croyance à Alice concernant la contenance de la boîte. Il suppose qu'elle croit que la boîte est vide. Cette faculté d'imputer des croyances aux autres s'appelle la prise de perspective conceptuelle (ou prise de perspective de niveau deux).

3.2.2 Usage en robotique

Comme indiqué dans la partie 3.1, pouvoir percevoir et raisonner sur l'environnement qui l'entoure sont des capacités nécessaires pour le robot mais non suffisantes lorsqu'il interagit avec des humains. Pour pouvoir comprendre la situation de l'humain, des recherches récentes ont tenté d'implémenter une sorte de théorie de l'esprit, en permettant au robot d'avoir la faculté de prise de perspective de niveau un (perceptuelle). Cynthia Breazeal présente dans [Breazeal 2006] un algorithme d'apprentissage qui prend en compte l'information concernant le point de vue visuel de l'instructeur afin d'apprendre convenablement une tâche. Dans [Johnson 2005], les auteurs appliquent la prise de perspective visuelle pour la reconnaissance d'action entre deux systèmes robotiques. Gregory Trafton dans [Trafton 2005] utilise à la fois la prise de perspective visuelle et spatiale pour identifier le référent indiqué par un partenaire humain. Dans [Ros 2010] la prise de perspective visuelle est utilisée pour simplifier la clarification de déclarations référentielles (referential utterances) dans des scénarios impliquant plusieurs objets.

Pour pleinement comprendre et se faire comprendre par l'homme, certaines études ont également visé à donner le niveau deux (conceptuel) de prise de perspective au robot. Cynthia Breazal dans [Breazeal 2009] propose l'une des premières implémentations faisant intervenir un humain et un robot ainsi que quelques capa-

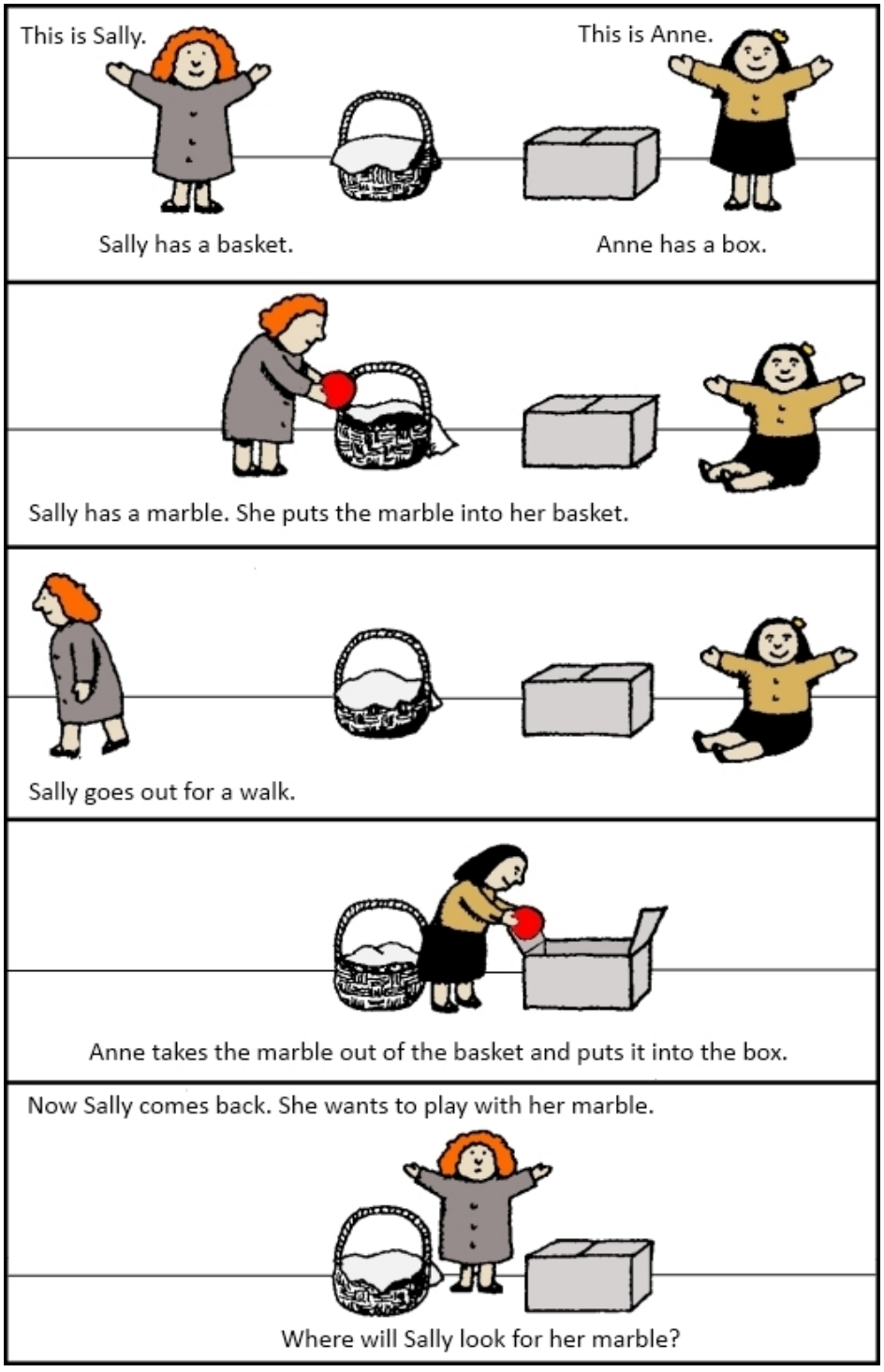


FIGURE 3.3 – Test de Sally et Anne permettant de vérifier la capacité d'un individu à identifier un état de connaissance erroné chez autrui. Illustration basée sur l'œuvre d'Axel Scheffler.

cités plus avancées de reconnaissance de but basées sur cette détection de fausse croyance.

3.3 Prise de perspective et état mental

3.3.1 Prise de perspective perceptuelle

Dans notre système, pour donner au robot la capacité de se mettre à la place de l'homme et de comprendre ce qu'il est capable de percevoir et les éléments qui sont à sa portée, nous avons ajouté le calcul de deux faits basés sur la représentation tridimensionnelle de l'état du monde expliquée au chapitre précédent. Le premier permet de connaître ce qui est visible par l'homme, c'est à dire ce qui se trouve dans son champ visuel et n'est pas caché (ou tout du moins qui est suffisamment visible).

Ainsi, dans la figure 3.4, l'humain en bleu est incapable de voir l'objet à droite (*WHITE_BOOK*) car il est caché par la boîte grise (GREY_BOX). Le système robotique est capable, en utilisant la position de la tête de l'homme et le modèle de l'environnement, de savoir quels objets sont visibles de l'homme et quel pourcentage de l'objet est visible. En utilisant ce pourcentage, le système peut produire un fait concernant la visibilité de l'objet avec une confiance associée. Par exemple ici, le fait serait :

```
Subject   property  target     value

ROBOT     canSee  WHITE_BOOK  true
HUMAN     canSee  WHITE_BOOK  false
```

Le robot est donc capable de savoir ce qu'il est capable de percevoir (moyennant éventuellement un mouvement de tête), mais également ce que l'homme peut ou non percevoir. Nous verrons dans le chapitre suivant comment cela peut être utilisé durant une interaction, notamment pour améliorer le dialogue situé en terme d'efficacité.

Le second fait ajouté concerne le calcul permettant de savoir si les objets qui entourent l'homme sont à sa portée. En utilisant la bibliothèque 3D Move3D [Siméon 2001], il est en effet possible de calculer la cinématique inverse d'un agent et de planifier ses mouvements. Grâce à cela, il est donc possible de savoir quels objets sont accessibles aux agents de la scène en prenant en compte les collisions éventuelles comme expliqué dans [Sisbot 2011]. Ce calcul est illustré par l'image 3.5, où le robot calcule l'accessibilité d'un objet par rapport à lui-même et à l'homme présent dans la scène.

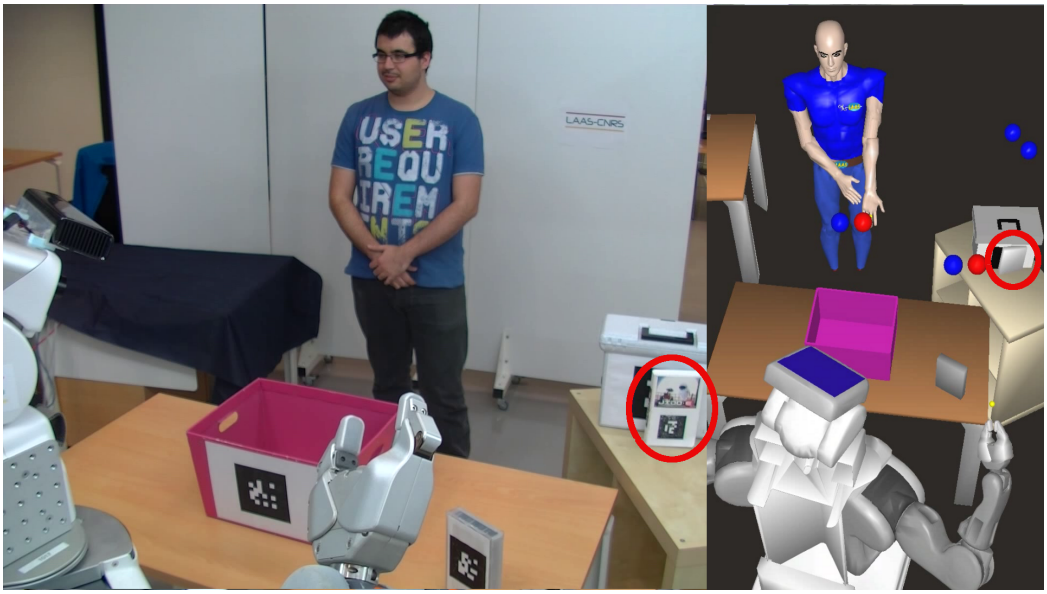


FIGURE 3.4 – Exemple d'état du monde où le système est capable de calculer la visibilité des objets pour chaque agent. Ici, le livre blanc n'est pas visible par l'homme.

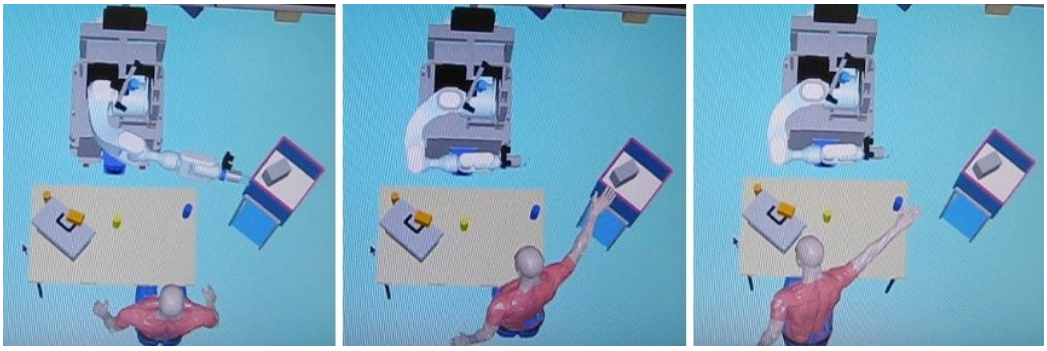


FIGURE 3.5 – Exemple d'état du monde où le système calcule si les agents sont capables d'atteindre un objet. Sur l'image de gauche, le robot est capable de calculer qu'il peut attraper l'objet gris. Il calcule également que l'homme peut attraper l'objet gris, comme illustré par l'image centrale. Si l'homme avait été placé plus loin, le robot aurait pu calculer que l'objet était inaccessible pour l'homme, comme illustré par l'image de droite.

La liste des faits (simplifiés) produits pour cette scène concernant l'accessibilité de l'objet gris serait pour les différentes configurations :

```
left and center pictures                    right picture

Subject   property  target    value         Subject   property  target    value

ROBOT     canReach  GREY_OBJ  true          ROBOT     canReach  GREY_OBJ  true
HUMAN     canReach  GREY_OBJ  true          HUMAN     canReach  GREY_OBJ  false
```

Pouvoir connaître quels objets sont à la portée de l'homme permet au robot de produire des plans collaboratifs prenant en compte cette donnée pour l'allocation de tâche, comme dans [Gharbi 2015]. Par exemple, si la tâche collaborative est de ranger des livres dans une boîte, si on a les faits :

```
Subject   property  target      value

ROBOT     canReach  BLUE_BOOK   true
ROBOT     canReach  WHITE_BOOK  false
ROBOT     canReach  BASKET      false
HUMAN     canReach  BLUE_BOOK   false
HUMAN     canReach  WHITE_BOOK  true
HUMAN     canReach  BASKET      true
```

le plan créé prendra en compte que le livre bleu (BLUE_BOOK) et le panier (BASKET) sont atteignables par le robot pour lui assigner la tâche de ranger ce livre. De même, comme le livre blanc (WHITE_BOOK) et le panier sont atteignables par l'homme, la tâche de le ranger lui sera assignée. Ce fait peut également aider à lever l'ambiguïté, par exemple si l'homme demande au robot d'apporter un objet, il est peu probable que l'homme face référence à un objet qui est à sa portée.

Pour comprendre l'homme, il est également important de comprendre sa situation spatiale, à savoir comment celui-ci est entouré et comment il est susceptible de décrire ce qui l'entoure par rapport à sa position.

Nous avons donc ajouté dans notre système d'évaluation de la situation la possibilité de faire des requêtes pour obtenir la position d'une entité du point de vue d'un agent. Il est par exemple possible de calculer de quel côté une entité se trouve (gauche, droite, avant, arrière). De plus, il est aussi possible d'obtenir la position d'une entité comparativement à une autre du point de vue d'un agent. Par exemple, il est possible pour le robot de dire à l'homme "la tasse que vous cherchez est **à votre droite**" ou "Donnez moi le livre qui est **pour vous à gauche de la tasse rouge**". Ces requêtes permettent donc au robot d'avoir une prise de perspective géométrique et ainsi de parler à l'homme en prenant son propre référentiel, améliorant ainsi les performances sociales du robot.

3.3.2 Prise de perspective conceptuelle

3.3.2.1 Modélisation des états mentaux

Nous avons expliqué dans le chapitre 2 qu'il est important de faire des hypothèses sur la position des objets pour pouvoir en assurer le suivi. Bien que nécessaire, le fait de créer des hypothèses implique la possibilité de se tromper. Dans cette section nous allons expliquer comment la prise de perspective va permettre au robot d'avoir

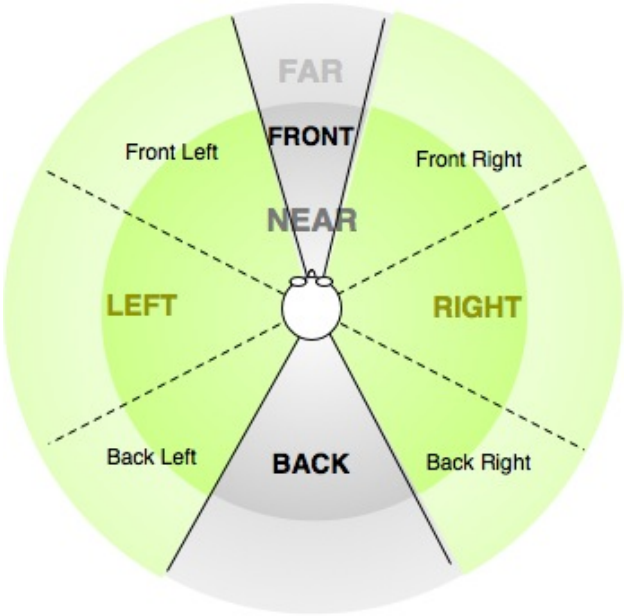


FIGURE 3.6 – Illustration du découpage géométrique pour désigner les emplacements en prenant l'homme comme référentiel. Cette illustration provient de la documentation de SPARK, un logiciel pour l'estimation de situation ayant été développé avant TOASTER : `https://www.openrobots.org/wiki/spark/what_spark_computes`.

connaissance de l'état de croyances des hommes avec lesquels il interagit et ainsi savoir lorsqu'ils ont une croyance erronée concernant l'environnement.

La première étape pour pouvoir connaître l'état mental d'un humain est d'avoir connaissance des informations qu'il reçoit. L'humain peut recevoir des informations sur son environnement de deux façons : soit il les reçoit directement (ou les déduit) de ses capacités perceptives, soit il les reçoit d'un autre agent à travers le dialogue. Pour donner au robot la capacité de prise de perspective décrite en 3.2.1, nous nous focalisons dans un premier temps sur la perception et sur le dialogue homme-robot pour déduire quelles informations l'humain peut acquérir.

Grâce aux calculs des affordances des agents expliqués en 3.3.1, il est possible de savoir lorsqu'un agent perçoit un objet. Lorsqu'un humain voit un objet, cela implique qu'il acquiert la connaissance de sa localisation, et donc met à jour cette information dans son état mental. Cela permet donc de suivre l'état de connaissance de chaque agent.

L'état mental de chaque agent (tel qu'il est estimé par le robot) est ainsi conservé et mis à jour dans un modèle séparé de l'état de connaissance que le robot lui-même possède sur l'environnement. Ainsi, le modèle représentant l'état mental de chaque agent se traduit par une liste de faits. Chaque modèle est indépendant et cohérent

d'un point de vue logique.

3.3.2.2 Gestion de croyances divergentes

Dans certains cas, l'homme et le robot peuvent avoir un modèle qui contient des valeurs différentes. Cela peut provenir d'une vision différente de la scène (e.g. certaines propriétés sont accessibles uniquement au robot, donc l'homme n'en a pas connaissance). Cela peut provenir de croyances divergentes entre l'homme et le robot (e.g. l'homme croit qu'un objet a la propriété P alors que le robot sait que cette propriété est fausse). Dans ce second cas, la prise de perspective n'est pas suffisante pour comprendre la croyance erronée de l'homme. Il est nécessaire de mettre en place la gestion de croyance divergente (divergent belief management en Anglais).

Ici, nous traitons uniquement de propriétés n'ayant pas d'évolution prédictibile. En effet, certaines propriétés ont une évolution prédictible, comme la température d'un breuvage, la fraîcheur d'une peinture... Ceci signifie que la propriété évolue et que cette évolution peut-être estimée sans observation. Par exemple, si un humain sait qu'une tasse de thé est chaude, en revenant une heure plus tard, sans avoir à observer la tasse, il va pouvoir estimer qu'elle est froide. Pour simplifier, nous considérons donc uniquement les propriétés dont l'évolution ne peut être prédite, comme c'est souvent le cas lorsqu'un autre agent agit sur l'environnement et modifie l'état d'une propriété (déplacement d'objet, remplir ou vider un container...).

Par exemple, si on considère qu'un humain appelé *GREG* quitte la zone d'interaction avec la croyance qu'un objet *Obj* a la propriété *Prop* dans un état *e1*.

On présente ci-dessous l'état de connaissance du robot et celui de *GREG* tel qu'il est modélisé par le robot.

```
    ROBOT                GREG
Obj Prop e1          Obj Prop e1
```

Durant son absence, imaginons que la propriété *Prop* appliquée à *Obj* a évolué pour atteindre un état *e2*. Lorsque *GREG* revient, même si la propriété a évolué, il pensera probablement que la propriété est dans le même état que lorsqu'il a quitté la scène, jusqu'à ce qu'il soit capable de réévaluer ses croyances en utilisant le dialogue ou la perception. Pour prendre en compte ce type de situation, le robot garde inchangé le modèle d'un agent lorsque celui-ci n'est pas présent pour observer les actions ou constater les changements induits, et ce jusqu'à son retour.

Avant que l'homme ne revienne, la croyance du robot a donc évolué mais pas celle de l'homme.

```
    ROBOT                GREG
Obj Prop e2          Obj Prop e1
```

Lorsque l'homme revient, on identifie trois situations différentes :

— Dans le premier cas, l'homme remarque le changement d'état de la propriété et corrige directement sa croyance erronée. Le robot corrige donc automatiquement le vecteur de faits représentant les croyances de *GREG* avec le nouvel état de la propriété.

```
        ROBOT                GREG
    Obj Prop e2          Obj Prop e2
```

Pour illustrer, on peut prendre comme objet une radio (*RADIO*) et comme propriété l'état (*turnedOn*) qui peut avoir comme valeur allumée (*TRUE*) ou éteinte (*FALSE*). Si *GREG* quitte la scène en sachant que la radio est allumée, lorsqu'il revient, n'entendant plus la radio, il pourra directement déduire la nouvelle valeur de la propriété d'état liée à la radio (éteinte).

```
        ROBOT                GREG
  RADIO turnedOn FALSE  RADIO turnedOn FALSE
```

— Dans le second cas, l'homme remarque le changement mais est dans l'incapacité de connaître le nouvel état de la propriété. L'humain étant au courant de son ignorance, le robot supprime sa croyance erronée. Pour ce faire, il met à jour le vecteur de faits avec un état *unknown* qui traduit le fait que le robot ait connaissance que l'homme sait qu'il ignore le nouvel état de la propriété.

```
        ROBOT                GREG
    Obj Prop e2          Obj Prop unknown
```

On peut prendre cette fois comme exemple un livre (*BOOK*) ayant comme propriété une position (*isOn*) et pouvant avoir pour valeur le meuble sur lequel il se trouve (*LIVINGROOM_TABLE*, *BEDSIDE_TABLE*, *BEDROOM_SHELF*). Si *GREG* quitte la scène en sachant que le livre est sur la table du salon, lorsqu'il revient, s'il observe que le livre n'est plus à sa place, il aura alors connaissance que la propriété de position a changé de valeur sans être capable de savoir la nouvelle valeur de celle-ci tant qu'il n'aura pas vu l'objet.

```
        ROBOT                      GREG
  BOOK isOn LIVINGROOM_TABLE  BOOK isOn unknown
```

— Dans le troisième cas, l'homme est incapable de remarquer que la propriété a changé. L'humain va alors garder sa croyance erronée concernant la propriété jusqu'à ce qu'il puisse constater un changement ou que le robot l'informe de l'évolution de la propriété.

```
        ROBOT                GREG
    Obj Prop e2          Obj Prop e1
```

Pour prendre un dernier exemple, si l'objet est une boîte de cookies (*CO-*

OKIE_BOX) opaque et que la propriété décrit la présence de cookies à l'intérieur de la boîte (*isFull*), pouvant prendre pour valeur pleine (*TRUE*) ou vide (*FALSE*). Si lorsque *GREG* quitte la scène il pense que la boîte est vide, si celle-ci a été remplie pendant son absence, *GREG* sera incapable de savoir que la boîte est à présent remplie tant que celle-ci sera fermée (et donc l'état de la propriété *isFull* non observable).

```
          ROBOT                        GREG
COOKIE_BOX isFULL TRUE       COOKIE_BOX isFULL FALSE
```

Pour résumer et formaliser ces différentes situations pour la mise à jour de l'état mental d'un autre agent, nous notons :

- *HB* le modèle de croyance de l'humain H et *RB* celui du robot R.
- p représente une propriété de l'environnement.
- *value(p,m)* représente la valeur de la propriété p dans le modèle m.
- *evo(p,A)* représente la capacité de l'agent A à percevoir l'évolution de la propriété p. *evo(p,A)* peut prendre les valeurs *UNSEEN* lorsque l'agent n'est pas au courant de l'évolution, *INVALIDATE* lorsqu'il sait que la propriété a évolué sans connaître la nouvelle valeur et *UPDATE* lorsqu'il est capable d'observer ou estimer la nouvelle valeur.

Dans le cas où il y a une croyance divergente, c'est à dire :
if $p \in RB, \quad p \in HB \wedge value(p, HB) \neq value(p, RB)$

On a alors les trois situations précédentes qui peuvent être représentées par les règles :

- if $evo(p, HB) = UNSEEN \rightarrow value(p, HB) = value(p, HB)$
- if $evo(p, HB) = INVALIDATE \rightarrow value(p, HB) = unknown$
- if $evo(p, HB) = UPDATE \rightarrow value(p, HB) = value(p, RB)$

Il est alors à constater que la partie complexe pour mettre à jour correctement les croyances de l'homme est de modéliser convenablement sa capacité à percevoir l'évolution de la propriété et donc l'obtention de la valeur de *evo(p,HB)*. Dans la section suivante nous présentons notre implémentation pour l'état de croyance des agents et nous proposons une solution pour l'estimation de cette valeur.

3.4 Implémentation

3.4.1 Traitement générique de la mise à jour de l'état de croyance

Pour pouvoir avoir un état du monde symbolique pour chaque agent afin de représenter la conception mentale que celui-ci a de l'environnement, il est possible d'ajouter des tables dans la base de données temporelle. Chaque table est comparable à la table *ROBOT_WS_TBL* décrite en section 2.6.1 et qui contient une

liste de faits décrivant l'état du monde perçu par le robot lui-même. En plus de ces tables contenant la représentation symbolique de l'environnement du point de vue d'un agent tel qu'elle est estimée par le robot, nous ajoutons également une table mémoire pour chaque agent. Cette table représente l'état de croyance passée de l'agent courant. Elle est comparable à la table *ROBOT_MEM_TBL* décrite en section 2.6.2.

Cependant, pour le robot la mise à jour de ces tables repose directement sur la perception de l'environnement basée sur les différents modules de TOASTER décrits en section 2.4 et sur le module de gestion des hypothèses décrit en section 2.5. Pour pouvoir mettre à jour les croyances de l'homme, il faut pouvoir mettre en place une gestion des croyances telle que décrite en section 3.3.2.2.

Pour ce faire, comme expliqué en section 3.3.2.2, lorsqu'un agent ne peut percevoir un événement, son état mental reste inchangé. Toute la difficulté ici est de parvenir à estimer lorsque l'homme est capable ou non de mettre à jour son état de croyance (c'est à dire la valeur de *evo(p,HB)* décrit dans 3.3.2.2). En effet, selon la propriété concernée, l'état peut être plus ou moins observable ou déductible selon divers moyens de perception.

Par exemple, la propriété de localisation d'un objet est une propriété directement observable lorsque l'objet est visible. En effet, si un homme observe un objet, il aura directement la connaissance de sa position. Par contre, si l'on reprend l'exemple de la boîte de cookies, l'observabilité de la propriété de contenance dépend de la propriété relative à l'ouverture de cette boîte. Ainsi, il est possible d'observer la contenance de la boîte de cookie seulement si celle-ci est ouverte. De plus, la perception peut passer par diverses modalités. Par exemple pour savoir qu'une tasse est chaude il est possible, soit de percevoir de la fumée émanant de la tasse, soit de ressentir la chaleur de celle-ci dans la main. Enfin, l'observabilité est également liée aux capacités de perception de chaque agent. Ainsi, si l'on reprend le cas de la radio, une personne sourde ne sera pas en capacité de savoir si celle-ci est allumée ou éteinte.

Afin de proposer une première implémentation et de mettre en place une gestion générique des hypothèses et du maintien de l'état mental de chaque agent, nous avons ajouté à chaque fait une notion d'observabilité, correspondant au champ *factObservability* déjà présenté en section 2.4.1 lors de la présentation de la structure de *fait*. L'observabilité représente la probabilité qu'un agent puisse acquérir l'état d'une propriété lorsque le sujet (*subject*) du fait est visible.

Pour gérer la mise à jour, nous utilisons les règles suivantes qui sont une adaptation des règles énoncées en section 3.3.2.2. Nous rappelons les notations précédemment utilisées et en introduisons de nouvelles.

— *HB* le modèle de croyance de l'humain H et *RB* celui du robot R.
— p représente une propriété de l'environnement.
— *value(p,m)* représente la valeur de la propriété p dans le modèle m.
— *obs(p)* signifie que la propriété p est observable.
— *probObs(p)* est l'observabilité de la propriété comme définie précédemment.
— *valid(p,A)* signifie que la propriété p ne contredit pas la perception courante de l'agent A.
— *vis(p,A)* signifie que l'agent A est capable de voir (lié à la propriété *canSee* évoqué en section 3.3.1) l'entité sujet de la propriété p.
— *conf(p,A)* représente la confiance que le robot a dans le fait que l'agent A ait effectivement mis à jour la valeur de la propriété p.

L'évaluation de la validité d'une propriété étant hautement variable, il n'a été implémenté pour l'instant que pour les propriétés de position. En effet, pour les propriétés de position, le fait de pouvoir observer l'emplacement précédent suffit à savoir que la propriété n'est plus valide. Nous avons donc l'ensemble des règles de mise à jour suivant, pour chaque propriété $p \in HB \cup RB$:

— if $p \in RB$, $p \notin HB$, $obs(p)$, $vis(p, H) \rightarrow value(p, HB) = value(p, RB)$.
— if $p \notin RB$, $p \in HB$, $obs(p)$, $vis(p, H) \rightarrow remove$ p $from$ HB.
— if $p \in RB$, $p \in HB$ then :
  — if $value(p, HB) \neq value(p, RB)$,
    $obs(p)$, $vis(p, H) \rightarrow value(p, HB) = value(p, RB)$
    $conf(p, H) = vis(p, H) * probObs(p)$.
  — if $value(p, HB) \neq value(p, RB)$,
    $!obs(p)$, $!valid(p, H) \rightarrow value(p, HB) = unknown$.

L'idée de cet ensemble de règles est de mettre à jour le modèle d'état mental d'un agent A seulement si le sujet de la propriété (pouvant être n'importe quel entité à savoir un homme, un robot, un objet ou le membre d'un agent) est observable par A, ou, dans le cas d'invalidité, s'il n'est pas observable mais que les données de perception contredisent la valeur actuelle de la propriété. Par exemple si une tasse a été déplacée de la table de la cuisine à une étagère, en l'absence de A, même si A ne peut voir où est la tasse, il peut voir qu'elle n'est plus sur la table.

3.4.2 Réflexion sur d'éventuelles améliorations

La méthode d'implémentation, basée sur les règles définies ci-dessus, permet un traitement "automatique" et générique des diverses propriétés. Cependant, cette implémentation est simplifiée car, comme indiqué précédemment, l'observation de

l'invalidité d'une propriété ou l'acquisition du nouvel état d'une propriété peut être très variable d'une propriété à l'autre.

L'une des façons d'améliorer simplement les règles déjà existantes serait de considérer l'observabilité d'une propriété non seulement en la basant sur la visibilité du sujet de la propriété par l'agent concerné, mais aussi de la cible pour certaines propriétés telles que les propriétés de relations entre objets (isOn, isIn, isNextTo) ou de capacités des agents (canReach, canSee). Ainsi selon la propriété, la mise à jour de l'état pourrait être liée à la possibilité de voir le sujet ou la possibilité de voir la cible ou la possibilité de voir les deux à la fois.

Une autre façon d'améliorer la gestion des croyances serait de créer un ensemble de règles définissant les différents moyens de mise à jour d'une propriété. Ainsi, pour le problème de la propriété de contenance de la boîte opaque, il faudrait vérifier que : la boîte est visible par l'agent, que son contenu est visible et que la boîte est ouverte. Un autre exemple, pour savoir qu'un agent *H1* est capable de savoir qu'un agent R peut voir un objet O il lui faut non seulement pouvoir voir la position de R mais aussi connaître (ou voir) la position de l'objet O. Cette méthode, bien qu'envisageable, nécessiterait de définir des règles pour chaque propriété, ou tout du moins chaque type de propriétés, et donc nécessite de définir un domaine réduit avec des règles données au robot par un expert. Un moyen d'éviter cela serait de développer des mécanismes d'apprentissage afin que le robot puisse produire lui-même ces règles.

3.5 Résultats expérimentaux

Nous présentons ici l'une des expériences menées au LAAS permettant de mettre en valeur la capacité du système à maintenir et mettre à jour l'état mental des humains présents grâce aux mécanismes de suivi et de mise à jour d'état mental présentés précédemment.

3.5.1 Configuration expérimentale

Tout d'abord, nous présentons brièvement les capteurs et modules que nous utilisons pour les expérimentations.

Ces expérimentations ayant été réalisées en 2014, l'infrastructure logicielle utilisée pour l'estimation de la situation est SPARK[Sisbot 2011, Warnier 2012, Milliez 2014a], qui est une version antérieure à TOASTER mais qui a un fonctionnement similaire pour le suivi de l'état mental des humains concernant les propriétés de position des objets.

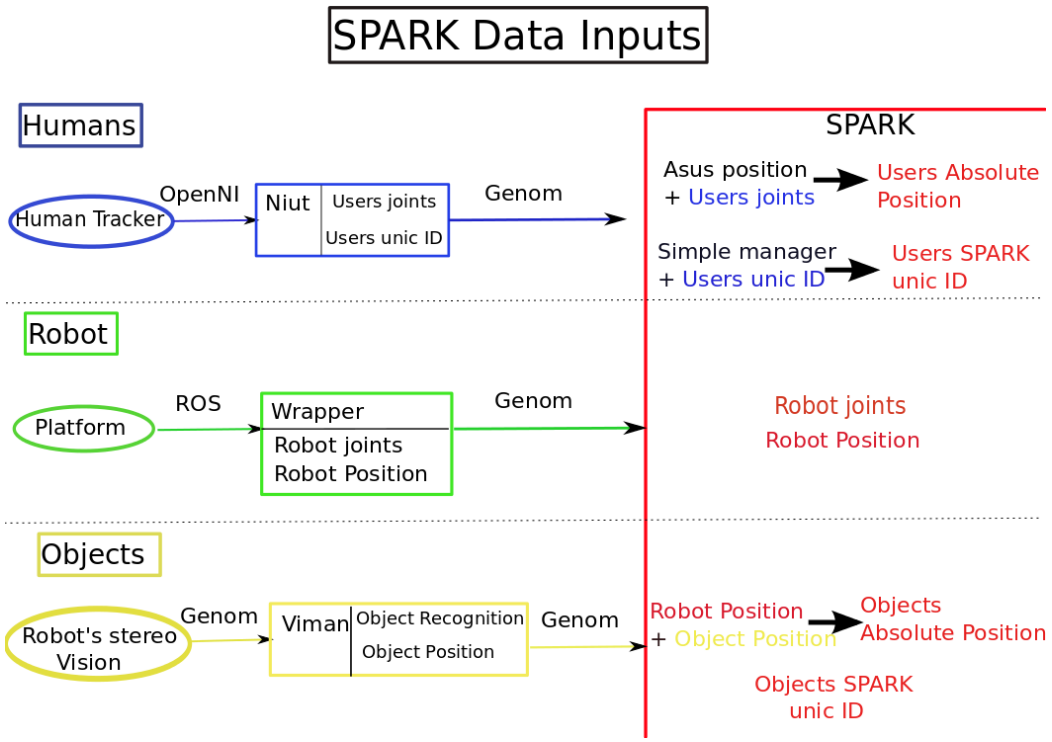


FIGURE 3.7 – Schéma représentant les entrées de SPARK.

Pour assurer le suivi des humains, nous utilisons une Asus xtion [1]. Comme le fait de bouger la tête est une partie importante de la communication pour permettre au robot de montrer son attention, et pour éviter la perte du suivi, nous avons fixé l'Asus xtion sur une base stable derrière le robot. Un module, appelé *niut*, est responsable de la gestion du suivi des humains, en utilisant l'API OpenNI. Ce module fait également une identification des humains basée sur la teinte. Pour obtenir la valeur moyenne de la teinte de chaque humain, on extrait la position du torse, puis nous la projetons dans les coordonnées de l'image RGB. Si les données sont disponibles, le module utilise également les coordonnées des épaules et des hanches pour définir le rectangle dans lequel la teinte sera calculée, sinon le module crée un rectangle autour de la position du torse, en adaptant l'échelle en fonction de l'éloignement de l'humain. En donnant au module les valeurs des teintes de chaque humain, il est possible de reconnaître l'humain et d'envoyer ses données de posture au module d'estimation de la situation avec un id unique. Cela permet également de filtrer les faux positifs dans la détection d'humains et évite que des humains non enregistrés perturbent le déroulement des expériences. Pour que les hommes soient à la bonne position dans le modèle du monde, une projection est effectuée à partir

1. `https://www.asus.com/fr/3D-Sensor/Xtion_PRO`

de la position et l'orientation de l'Asus xtion.

La posture et la position du robot sont directement obtenues grâce à ses capteurs internes en utilisant l'intergiciel ROS.

Enfin, pour obtenir la position des objets, nous utilisons le module *Viman* qui est basé sur la bibliothèque de réalité augmentée ARToolKit [2]. Ce module utilise la vision stéréo du robot pour reconnaître et localiser les objets. Pour ce faire, *viman* scanne des RTags placés sur les objets. Une fois que la position relative (à la caméra) de l'objet est connue, elle est envoyée au module d'estimation de la situation qui, en utilisant la position de la tête du robot pourra obtenir la position de l'objet dans le monde par projection. La figure 3.7 illustre cette implémentation.

3.5.2 Résultats

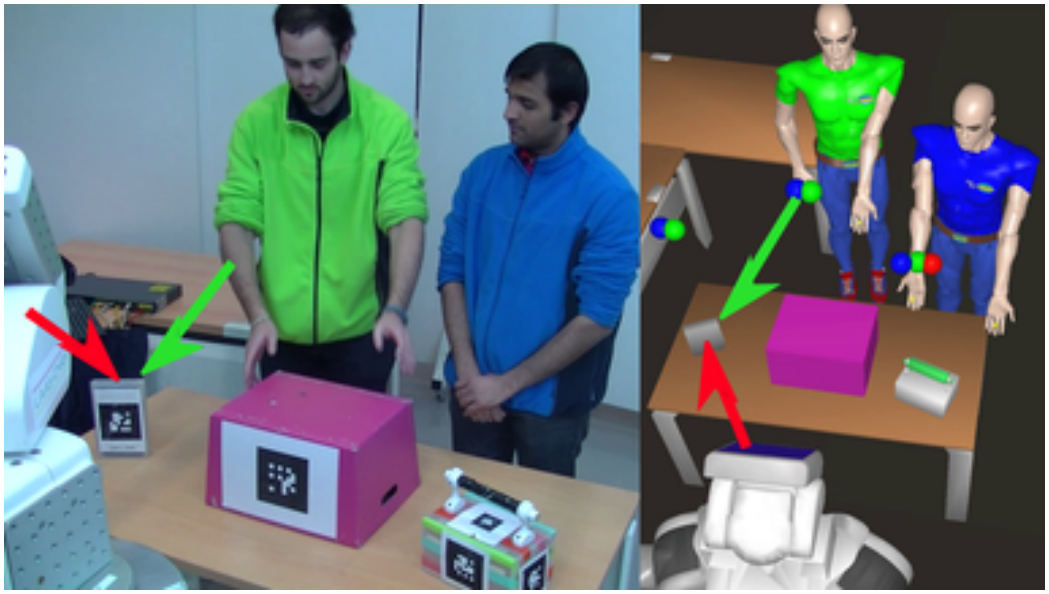
Nous avons introduit ci-dessus comment nous procédons pour suivre les croyances distinctes pour chaque agent. Nous croyons que cette fonctionnalité est utile pour comprendre la verbalisation de l'homme, ses actions et la focalisation de son attention, i.e. pour interagir avec des humains. Comme le robot connaît les croyances des humains, il peut décider des informations qui sont nécessaires à fournir à l'homme et également s'il doit parler ou non en fonction de la situation actuelle ou de l'état de réalisation du plan collaboratif. La fonctionnalité de gestion de croyance présentée dans ce chapitre a été utilisée sur un robot afin de la tester [3].

3.5.2.1 Le Test de Sally et Anne

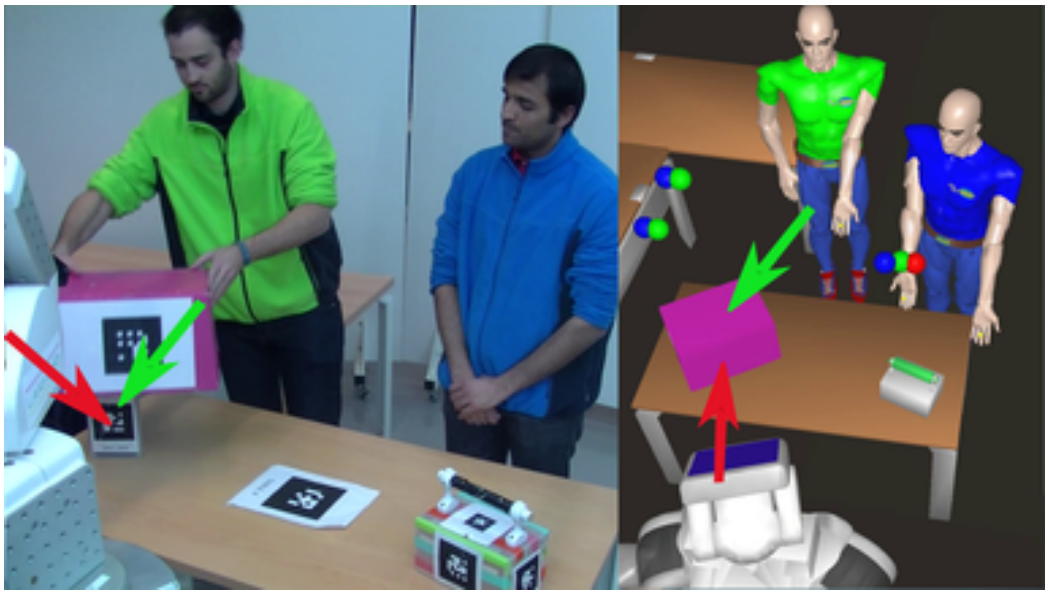
Pour aider à la compréhension des scénarios, la figure 3.8 et la figure 3.9 sont des images de l'expérimentation réelle avec des captures d'écran de la modélisation 3D du module de raisonnement spatial.

2. http ://artoolkit.org/

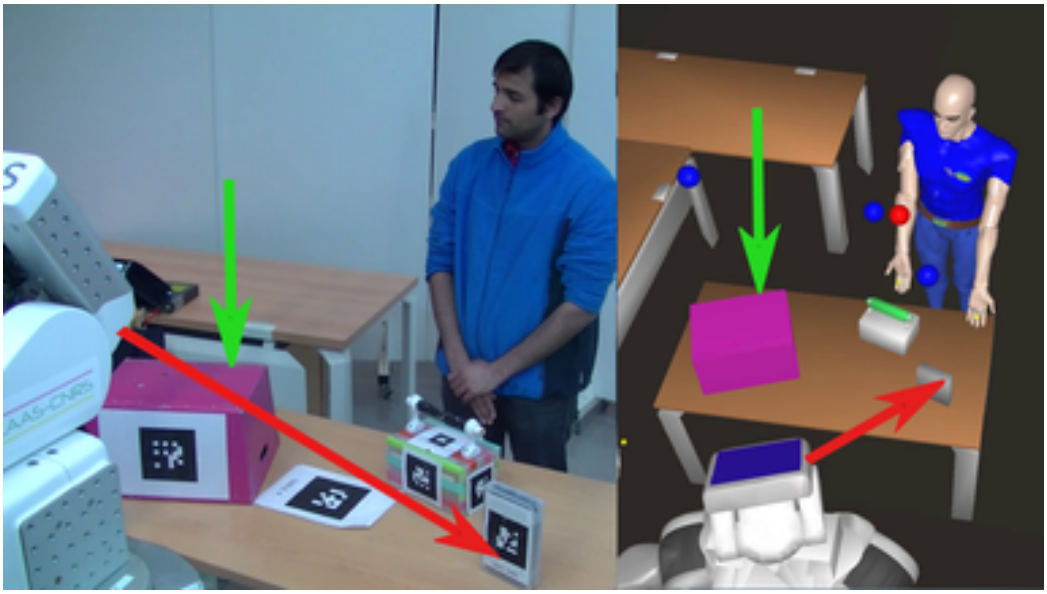
3. Des vidéos des expérimentations sont accessibles à l'url : `http://homepages.laas.fr/gmilliez/roman2014/`



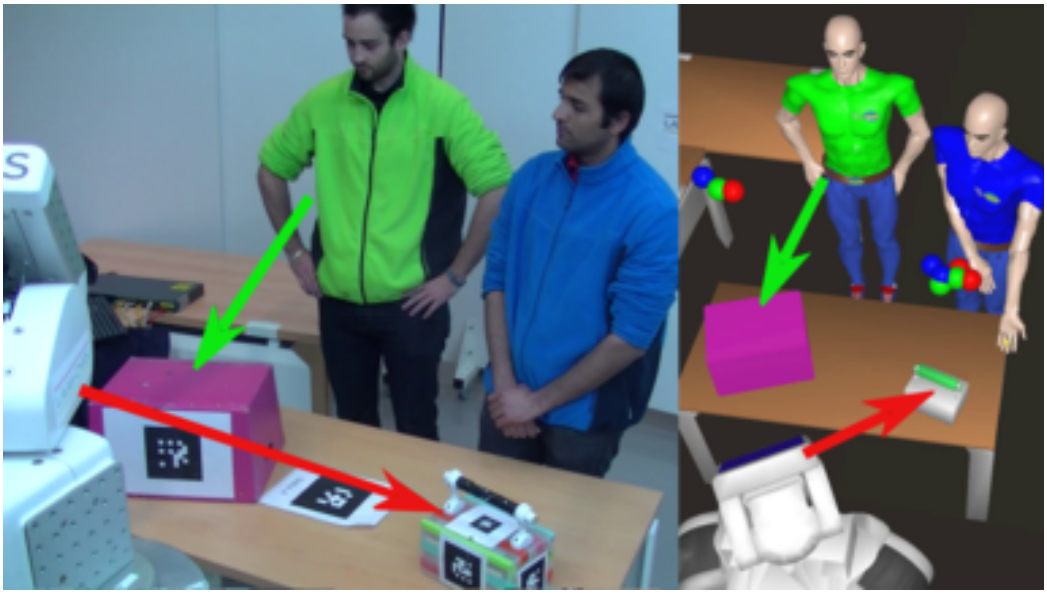
(a) Greg (en vert) et Bob (en bleu) font face au robot. Ils connaissent la position de chaque objet.



(b) Greg met la boîte rose sur le livre blanc.



(c) Greg part et Bob enlève le livre blanc de la boîte rose.



(d) Bob met le livre blanc sous la petite boîte, puis Greg revient. Le robot est capable de comprendre que Greg croit que le livre est sous la boîte rose.

FIGURE 3.8 – Le test de Sally et Anne dans l'environnement réel (partie de gauche) et tel qu'il est perçu par le robot (partie de droite). Pour aider à la compréhension, nous montrons explicitement les croyances concernant la position du livre blanc. Les croyances de Greg sont indiquées avec des flèches vertes et celles du robot avec des flèches rouges. Les petites sphères de couleur dans l'environnement perçu sont utilisées comme des indications sur l'accessibilité d'un objet pour un agent.

Dans le premier scénario, afin de tester notre système et d'illustrer la capacité du robot à détecter les croyances fausses/divergentes sur la position d'un objet nous avons décidé de le soumettre à une adaptation du test de Sally et Anne, déjà présenté par la figure 3.3. Dans notre expérience, deux utilisateurs Greg (en vert) et Bob (en bleu) font face au robot. La scène est composée d'un livre blanc et de deux boîtes se trouvant sur une table 3.8(a)). Greg met le livre blanc sous la boîte rose (3.8(b)). Puis Greg quitte la scène. Pendant que Greg est parti, Bob prend le livre blanc et le met sous la petite boîte. (3.8(c) et 3.8(d)). Puis, quand Greg revient, nous demandons au robot où Greg pense que le livre se trouve.

Nous allons présenter ci-dessous quelques faits symboliques intéressants pour Greg et le robot correspondants à la situation décrite en 3.8(d). Les croyances de Bob restent identiques à celles du robot et ne seront donc pas affichées par simplification.

```
          ROBOT                          GREG
GREG canSee PINK_BOX  TRUE      GREG canSee PINK_BOX  TRUE
GREG canSee SMALL_BOX TRUE      GREG canSee SMALL_BOX TRUE
GREG canSee BOOK      FALSE     GREG canSee BOOK      FALSE
BOOK isIn   SMALL_BOX TRUE      BOOK isIn   PINK_BOX  TRUE
```

Comme le robot sait que le livre n'est pas visible par Greg, il n'a pas mis à jour ses croyances concernant la position de cet objet. Avec l'aide du système d'estimation de la situation maintenant le modèle de croyance des agents, le robot est capable d'observer que, comme Greg ne peut observer ce qui a changé, il a une croyance divergente sur la position de l'objet. Si l'homme et le robot doivent accomplir une tâche impliquant cet objet, le robot sera capable de savoir qu'il doit informer l'homme de la position de l'objet.

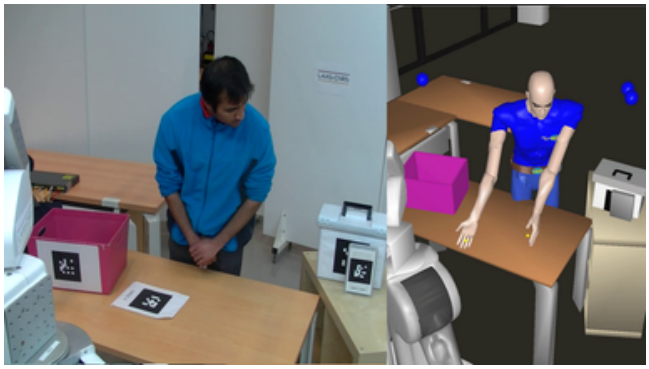
3.5.2.2 Gestion de croyances divergentes

Pour aller plus loin et illustrer les trois différentes situations présentées en section 3.3.2.2, nous avons tourné un autre scénario illustré par la figure 3.9. Dans cette deuxième expérimentation, deux utilisateurs, Greg et Bob, font face au robot. Ils ont un livre blanc et une boîte blanche qui servira pour faire de l'occlusion visuelle (3.9(a)). Une fois que Bob part (3.9(b)), Greg prend le livre blanc et le met derrière la boîte blanche pour qu'il ne soit pas visible du point de vue humain (3.9(b)). Puis Greg part et Bob revient. Le robot est capable d'inférer que, comme le livre blanc est caché, Bob ne sait pas où il se trouve. Toutefois, Bob peut voir que le livre n'est plus là où il pensait être. Cette information manquante est représentée par une sphère transparente à l'emplacement où Bob pensait trouver l'objet (3.9(c)).

(a) Greg (en vert) et Bob (en bleu) font face au robot. Ils connaissent la position de chaque objet.

(b) Bob part, Greg met le livre blanc derrière la boîte blanche.

(c) Greg part et Bob revient. Bob ignore la position du livre blanc. Le robot est au courant de ce manque d'information (une sphère bleue transparente représente la dernière croyance de Bob concernant la position du livre blanc).



(d) Bob regarde derrière la boîte blanche (et donc met à jour ses croyances) et prend le livre blanc.

(e) Bob part avec le livre blanc, puis Greg revient. Greg pense toujours que le livre blanc est derrière la boîte blanche. Le robot est au courant de cette croyance divergente (Une sphère opaque verte représente la croyance actuelle de Greg concernant la position du livre blanc).

(f) Greg regarde derrière la boîte blanche et est alors informé que sa croyance est erronée mais ne connaît pas la nouvelle position de l'objet (la sphère devient transparente).

FIGURE 3.9 – Scénario de croyances divergentes impliquant un robot Pr2 et deux humains. Le robot est capable de modéliser les croyances de chacun. Les images présentent les étapes du scénario dans l'environnement réel (partie de gauche) et tel qu'il est perçu par le robot (partie de droite).

Ci-dessous sont représentés quelques faits symboliques concernant les croyances de Bob et du robot tel que le robot les modélise.

ROBOT				BOB			
ROBOT	canSee	WHITE_BOX	TRUE	ROBOT	canSee	WHITE_BOX	TRUE
BOB	canSee	WHITE_BOX	TRUE	BOB	canSee	WHITE_BOX	TRUE
BOB	canSee	BOOK	FALSE	BOB	canSee	BOOK	FALSE
BOOK	isNextTo	WHITE_BOX	TRUE	BOOK	location		unknown

A l'étape suivante, Bob regarde derrière la boîte blanche et met à jour ses croyances (3.9(d)). Les croyances de Bob et du robot sont alors identiques. Puis il prend le livre blanc et s'en va. Lorsque Greg revient, comme l'objet était caché auparavant et n'est toujours pas visible, le robot est capable de comprendre que Greg n'a pas mis à jour ses croyances et a donc une croyance erronée sur la position du livre blanc. Cette croyance erronée est représentée par une sphère opaque à l'emplacement où Greg pense que le livre se trouve (3.9(e)).

Ci-dessous sont représentés quelques faits symboliques concernant les croyances de Greg et du robot tel que le robot les modélise à ce moment là.

ROBOT				GREG			
ROBOT	canSee	WHITE_BOX	TRUE	ROBOT	canSee	WHITE_BOX	TRUE
GREG	canSee	WHITE_BOX	TRUE	GREG	canSee	WHITE_BOX	TRUE
GREG	canSee	BOOK	FALSE	GREG	canSee	BOOK	FALSE
BOB	hasInHand	BOOK	TRUE	BOOK	isNextTo	WHITE_BOX	

Puis Greg regarde derrière la boîte blanche. Dès qu'il voit que le livre n'y est pas, il n'a plus de croyance divergente mais il ignore toujours où se trouve le livre. Comme le robot a suivi cette action d'observation, il a mis à jour le modèle d'état mental de Greg. La sphère représentant la croyance de Greg sur la position du livre est alors devenue transparente (3.9(f)).

Ci-dessous sont représentés quelques faits symboliques concernant les croyances de Greg et du robot tel que le robot les modélise à ce moment là.

ROBOT				GREG			
ROBOT	canSee	WHITE_BOX	TRUE	ROBOT	canSee	WHITE_BOX	TRUE
GREG	canSee	WHITE_BOX	TRUE	GREG	canSee	WHITE_BOX	TRUE
GREG	canSee	BOOK	FALSE	GREG	canSee	BOOK	FALSE
BOB	hasInHand	BOOK	TRUE	BOOK	location		unknown

Sans notre algorithme, à chaque étape le robot penserait que les humains ont la même croyance que lui concernant la position des objets (à savoir que l'humain connaît la nouvelle position de l'objet).

3.6 Conclusion

Nous avons montré comment en ajoutant des calculs géométriques à notre système d'évaluation de la situation il nous est possible de mettre en place un suivi de la situation spatiale de l'homme et de ses affordances en terme de visibilité et d'accessibilité des éléments.

Basé sur cette visibilité, nous avons montré comment il est possible de modéliser et de maintenir un état de croyance pour chaque agent. Ceci est réalisé en utilisant des règles fondées sur l'observabilité des propriétés et sur la possibilité pour l'agent concerné de voir l'entité liée à la propriété. Nous avons également conduit une étude permettant de valider et mettre en pratique la capacité de prise de perspective conceptuelle du robot, notamment en faisant passer une réplique du test de Sally et Anne.

Ces capacités de prise de perspective et de raisonnement sur l'état des agents en terme de situation spatiale, de capacité perceptive et d'état de croyance sont essentielles pour une bonne compréhension de l'homme. Dans le chapitre suivant, nous montrerons comment la prise de perspective de niveau aide à l'identification de référent pour le dialogue situé homme robot et comment la prise de perspective de niveau deux peut être intégrée à un système de dialogue pour donner un aspect "situé" et ainsi en améliorer l'efficacité et le taux de succès.

Dans le chapitre 5, la capacité de prise de perspective conceptuelle est utilisée pour interpréter convenablement les actions d'un homme afin d'en déduire son intention et de donner un comportement proactif au robot.

Chapitre 4

Vers un Dialogue Situé Homme-Robot

Sommaire

4.1 Contexte

4.1.1 Introduction

L'une des compétences essentielles pour interagir convenablement avec l'homme est de fournir des moyens d'assurer la compréhension mutuelle dans le contexte situé de l'activité jointe. Le robot et l'humain doivent avoir des références aux éléments de l'environnement qui soient communes (common ground), ce qui signifie qu'ils doivent pouvoir identifier, dans leur propre représentation du monde, les actions, les entités (humains, robots ou objets) et les propriétés énoncées par leur interlocuteur.

Les systèmes robotiques reposent sur les capteurs pour reconnaître et localiser les entités afin de construire l'état du monde. Ces capteurs produisent des coordonnées pour positionner les entités par rapport à un repère donné. Par exemple, une caméra stéréo avec un logiciel de reconnaissance peut permettre de savoir qu'une tasse est à une position donnée x, y, z avec une orientation θ, ϕ, ψ. Les humains quand à eux, utilisent les relations spatiales entre les éléments pour décrire leur position. Pour indiquer la position de la tasse, l'humain dirait par exemple qu'elle se trouve sur la table de la cuisine, sans donner les coordonnées précises. Pour comprendre les références de l'homme et générer des déclarations compréhensibles, le robot se doit donc de construire une représentation symbolique du monde, basée sur les données géométriques qu'il a collectées par ses capteurs, comme cela a été fait dans [Lemaignan 2012].

En plus de l'état du monde (qui peut être considéré comme l'état de croyance du robot), pour vraiment comprendre les actes dialogiques de l'humain dans un contexte situé, il est nécessaire au robot de comprendre la situation spatiale et mentale de l'homme. En effet, les références qu'il crée dans ses actes communicatifs avec le robot sont susceptibles de grandement dépendre de cette situation spatiale ou mentale.

Pour établir ces références communes et comprendre les actes communicatifs humains à la lumière de sa situation, nous avons utilisé notre infrastructure logicielle d'évaluation de la situation, présentée dans les chapitres précédents, dans le cadre d'un projet visant à mettre en place un dialogue situé homme-robot

4.1.2 Le projet MaRDi

Ces travaux ont été réalisés dans le cadre du projet MaRDi [1] de l'Agence Nationale pour la Recherche (ANR). Il a été financé par l'appel à projet Contenu et interactions. Les travaux réalisés ont été faits en collaboration avec le Labora-

1. Man-Robot Dialogue - http ://mardi.metz.supelec.fr

toire d'Informatique Fondamentale de Lille (LIFL), l'École supérieure d'électricité (Supélec), le groupe Acapela, le Laboratoire Informatique d'Avignon (LIA) et le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS). Ce projet vise à étudier l'apport d'une approche "située" du dialogue homme-robot. Le terme "située" est ici relatif à l'incarnation physique d'un système de dialogue dans une plateforme robotique qui permet l'intégration d'informations issues de la perception du robot dans le contexte de l'interaction pour compléter ou lever des ambiguïtés introduites par le médium vocal. Ce projet s'inscrit également dans l'utilisation de méthodes d'apprentissage numérique, exploitant les données collectées au travers de la conduite de véritables interactions afin d'améliorer l'efficacité et le naturel du système dans le temps. L'originalité de l'approche est de ne pas considérer les technologies vocales comme disponibles et dissociées de la tâche d'interaction homme-robot, mais bel et bien comme moyen d'en améliorer l'expérience et les performances.

En effet, le système de dialogue utilisé dans le cadre de ce projet est basé sur le système de résolution de la tâche *TownInfo* qui vise à fournir aux utilisateurs un service vocal donnant accès à des informations touristiques concernant une ville virtuelle et portant essentiellement sur la recherche de données pratiques (téléphone, adresse, etc.) sur des établissements (hôtels, restaurants, bars et infrastructures). Il s'agit donc d'un système de dialogue de type recherche d'informations pour s'enquérir des choix de l'utilisateur (e.g : restaurant chinois à cinq minutes) et pouvoir lui proposer les établissements correspondant au mieux à cette demande. Nous utilisons une implémentation du système de dialogue basée sur celle proposée par l'université de Cambridge et qui repose sur le formalisme POMDP-HIS [Young 2010]. L'une des caractéristiques de ces systèmes est qu'ils peuvent également effectuer une relâche de contrainte pour proposer une solution proche des critères de l'utilisateur si aucune solution n'a été trouvée.

La tâche MaRDi ne peut cependant pas être directement assimilable à un problème de recherche d'information standard. Pour donner l'aspect incorporé, le robot doit être capable de maintenir un contexte d'interaction suffisamment riche pour pouvoir être à même de prendre des décisions sur la suite à donner à celle-ci. Ce contexte intègre les entrées fournies par l'humain mais aussi les informations issues des calculs géométriques et des raisonnements effectués à partir de la perception de l'environnement. Sur ces aspects en particulier, le projet s'appuie sur les recherches présentées aux chapitres précédents, pour le raisonnement spatial et la prise de perspective, pour tenter de résoudre des ambiguïtés. Pour prendre des décisions afin de poursuivre l'interaction, la machine devra s'appuyer sur un contexte de l'interaction (historique, état de l'environnement, etc.) et tenir compte de son aspect incertain. En effet, dans le cadre situé, le contexte de l'interaction ne peut être

considéré comme une donnée sûre car de possibles erreurs ont pu être introduites par la chaîne de traitement automatique des entrées vocales et visuelles. Une fois sa décision prise, le système devra également pouvoir la restituer vocalement et physiquement à l'humain. Ainsi il faudra qu'il soit capable de planifier ses mouvements et ses actions physiques de façon précise pour répondre aux besoins de l'utilisateur (déplacer un objet, se rendre dans une pièce, etc.), mais également capable de s'exprimer de façon adéquate pour se faire comprendre par l'utilisateur et lui témoigner de sa compréhension du contexte interactif courant. Pour cela, le robot devra par exemple adopter des attitudes physiques particulières, ou encore utiliser une intonation de voix particulière.

4.1.3 Scénario associé

Afin de positionner le scénario dans un cadre naturel et fonctionnel, le choix a été fait de faire interagir un robot assistant avec une personne handicapée dans son appartement (aide à la personne). La personne pourra ainsi, en interagissant avec le robot, lui faire manipuler divers objets. Les objets en question auront des propriétés associées en terme de couleur, de type d'objet (par exemple un DVD), de type de contenu (par exemple science-fiction) et de position et auront un identifiant unique. Pour se faire comprendre par le robot, l'utilisateur usera principalement de la parole, mais pourra également employer des gestes déictiques (comme le fait de pointer un objet particulier). Un dialogue multimodal sera ici employé pour pouvoir résoudre les possibles ambiguïtés liées à une précision insuffisante de la requête utilisateur. Ce dialogue se poursuivra jusqu'à la fin de l'exécution effective de la tâche ou l'échec de l'interaction. Cette dernière situation peut par exemple être due à un désengagement explicite de l'utilisateur, ou encore à l'exécution d'une commande erronée de la part du robot. Un exemple d'un tel dialogue multimodal est donné dans le tableau 4.1. Dans ce contexte d'étude nous pourrons également jouer sur la définition de divers scénarios (configuration de l'environnement, connaissances initiales du robot, position des agents, etc.) pour rendre l'interaction plus ou moins complexe. Ceci nous permet de pouvoir tester divers cas d'interactions, notamment dans des configurations "limites" comme celles employées lorsque nous étudierons les tâches de fausse croyance (voir 4.3.3.2).

Le robot utilisé est le PR2. L'appartement choisi pour conduire nos expériences est la réplique taille réelle d'un 3 pièces (salon, cuisine, chambre) présent dans les locaux du LAAS-CNRS et dont une modélisation fidèle a été faite sur simulateur 3D. Dans chaque pièce se trouvent des meubles, sur lesquels des objets peuvent être posés. Une version tridimensionnelle de cet environnement est représenté à la figure

Robot1	Comment Puis-je vous aider ? hello() [Greet]
User1	Peux-tu mettre le livre sur ma table de chevet ? inform(cmd.action=move, cmd.location=bedroom_bedsidetable)
Robot2	Désolé, mais savez vous où se trouve le livre dont vous me parlez ? confreq(object.type=book,object.location) [TentRQ]
User2	C'est celui là. inform(idobj= ?) + pointsAt BLUE_BOOK = inform(idobj=BLUE_BOOK)
Robot3	Ok, je vais mettre le livre bleu sur votre table de chevet. execute(cmd.action=move,cmd.location=bedroom_bedsidetable, idobj=BLUE_BOOK,object.location=livingroom_table,object.type=book, object.color=blue) [Execute] move(BLUE_BOOK,livingroom_table,bedroom_bedsidetable)

TABLE 4.1 – Exemple de dialogue multimodal.

4.1.

FIGURE 4.1 – Représentation tridimensionnelle de l'environnement utilisé dans le cadre du projet MaRDi.

4.2 Les trois phases de l'interaction

L'architecture mise en place pour la réalisation de l'interaction contient divers modules qui interviennent à différentes étapes. Pour plus de clarté, nous distinguons trois phases de l'interaction permettant d'aller de la détermination du but utilisateur à son accomplissement.

— La première phase fait intervenir les éléments permettant le bon fonctionnement du dialogue situé afin de communiquer avec l'utilisateur pour pouvoir le satisfaire (en lui procurant une information ou en agissant sur l'environnement).

— La deuxième phase consiste à transmettre un but au planificateur afin que celui-ci puisse, en utilisant les informations sur la configuration actuelle de l'environnement, trouver un plan qui permette d'atteindre le but. Ce but peut être issu d'une demande explicite de l'utilisateur (e.g. *"apporte moi le DVD du Seigneur des Anneaux"*) ou d'un but "intermédiaire" afin que le robot puisse acquérir une information manquante sur l'environnement.

— La dernière phase fait intervenir la supervision qui pilote la planification de trajectoires et de mouvements et contrôle le bon déroulement de l'exécution de la tâche à accomplir.

Ces différentes phases font intervenir divers éléments du système. Durant une interaction les différentes phases sont utilisées à divers moments pour pouvoir faire progresser l'interaction. Ainsi, selon ce que veut l'utilisateur certaines phases peuvent être appelées plusieurs fois ou pas du tout. En effet, dans le cas le plus simple où l'utilisateur demande une information au robot et que celui-ci a déjà cette information, seule la première phase sera utilisée alors que sur des scénarios plus complexes, le système va boucler plusieurs fois en passant par les différentes phases avant de pouvoir satisfaire la demande de l'utilisateur. Nous allons détailler les processus impliqués dans chaque phase et les échanges de données.

4.2.1 Détermination de la demande utilisateur

Durant cette phase, l'humain et le robot dialoguent et le système essaye de caractériser la demande de l'utilisateur. Pour expliquer les différents flux, nous nous appuierons sur la figure 4.2.

Dans cette première phase, l'homme peut en permanence agir sur l'environnement (flux A). L'environnement étant en permanence mis à jour par notre système d'évaluation de la situation (B), la base de connaissance sera mise à jour en conséquence. Lors de cette phase d'estimation de la demande de l'utilisateur, l'homme parle au robot (1) en utilisant potentiellement des signes pour accompagner sa pa-

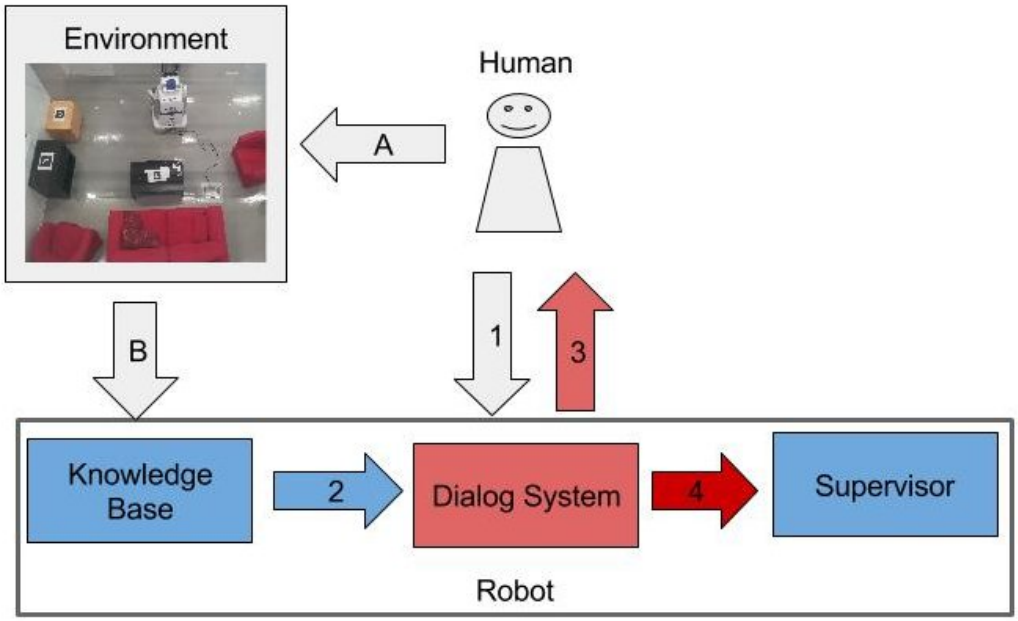


FIGURE 4.2 – Schéma des différents composants et flux intervenant dans la phase de détermination du but utilisateur.

role. Le système de dialogue du robot interprète l'acte communicatif de l'homme en utilisant sa base de connaissances (2) et produit une réponse appropriée à l'homme, par exemple en lui demandant une précision sur la tâche à accomplir (3). L'opération est répétée (flux 1, 2, 3) jusqu'à ce que le système de dialogue ait une compréhension suffisante de ce que veut l'homme. Si le système de dialogue peut répondre à la demande il le fait et attend la prochaine tâche. Cependant, dans le cas où le système robotique doit agir pour acquérir une information ou pour apporter des modifications à l'environnement, à la place de répondre à l'homme (3) le système de dialogue envoie un but au superviseur (4). La première phase est alors interrompue et l'interaction passe dans la phase d'élaboration de plan permettant de résoudre le but (qui peut être directement un but provenant de l'utilisateur ou un but intermédiaire permettant l'acquisition d'informations).

4.2.2 Élaboration de plan

Durant cette phase, le robot cherche à générer un plan permettant de résoudre le but produit lors de la première phase. Pour expliquer les différents flux, nous nous appuierons sur la figure 4.3.

À la fin de la première phase, lorsque un but est produit, le système de dialogue transmet le but au superviseur (flux 4). Le superviseur par la suite fait une requête

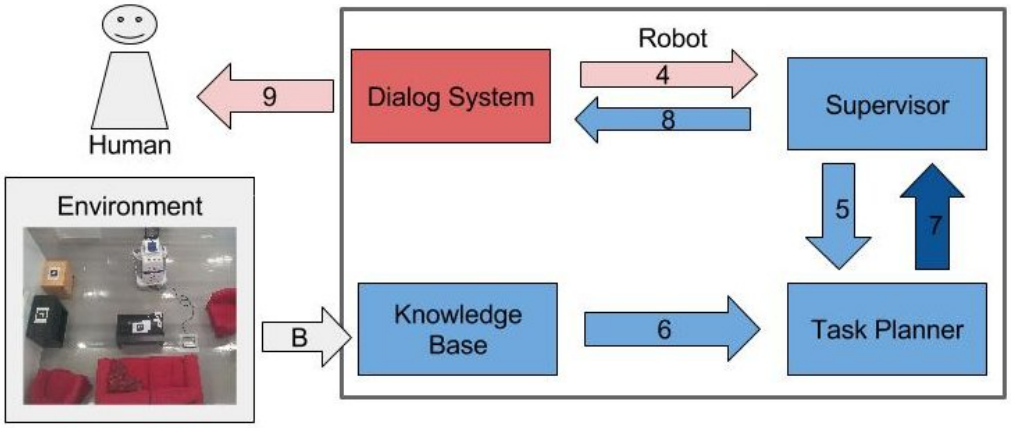


FIGURE 4.3 – Schéma des différents composants et flux intervenant dans la phase d'élaboration de plan pour résoudre un but.

au planificateur de tâches pour résoudre le but (5). Le planificateur de tâche utilise la représentation symbolique de l'environnement, contenu dans la base de connaissance et provenant des calculs et raisonnements faits par le système d'évaluation de la situation, pour avoir connaissance de l'état du monde actuel et générer un plan permettant d'atteindre le but souhaité par l'utilisateur. Le planificateur de tâche renvoie un message au superviseur, l'informant du succès ou de l'échec de la planification (7). En cas d'échec, le superviseur informe le système de dialogue (8) qui à son tour informe l'humain de l'impossibilité d'exécution du plan (9). L'interaction retourne alors à la phase précédente afin de tenter d'établir un but utilisateur qui soit réalisable. Dans le cas où la planification est un succès (un plan a été trouvé), le plan est envoyé au superviseur (7) et l'interaction peut passer à la phase d'exécution du plan.

4.2.3 Exécution du plan

Durant cette phase, le robot cherche à exécuter les tâches du plan permettant d'accomplir le but. Pour expliquer les différents flux, nous nous appuierons sur la figure 4.4.

À la fin de la phase précédente, le planificateur de tâche transmet le plan à la supervision (flux 7). Le composant de supervision gère l'exécution de la tâche grâce à la planification de mouvements (10) qui envoie des commandes aux actionneurs du robot (12). Le superviseur contrôle la bonne exécution du plan à l'aide des éventuels retours d'erreurs des différents composants et en surveillant l'évolution de l'état du monde (14). Durant cette exécution, l'homme peut à tout moment dialoguer avec

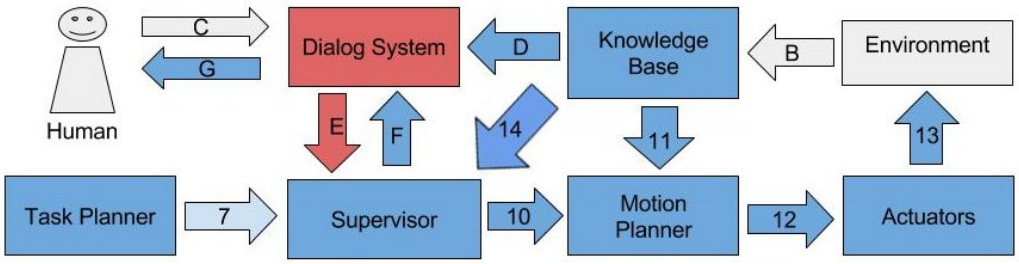


FIGURE 4.4 – Schéma des différents composants et flux intervenant dans la phase d'exécution de plan pour accomplir le but utilisateur.

le système (C) afin de :

- Demander une information sur l'environnement. Dans ce cas le système de dialogue utilise la base de connaissance (D) pour répondre au mieux à l'humain (G).
- S'enquérir de l'état d'avancement de la tâche ou de l'action courante du robot. Dans ce cas le système de dialogue envoie une requête correspondante au superviseur (E) qui transmet cette information à l'homme (G) par le système de dialogue (F).
- Demander un abandon de la tâche ou un changement de plan. Le système de dialogue transmet alors cette requête au superviseur (flux E). Puis en cas d'abandon, l'interaction retourne en phase de détermination de but utilisateur ou en cas de changement de plan, en phase d'élaboration d'un nouveau plan prenant en compte les demandes de l'utilisateur. Cette dernière fonctionnalité est présentée plus en détail en section 6.4.

4.3 Architecture du système de dialogue situé

Dans cette partie, nous allons aller plus loin dans l'explication du système de dialogue situé, intervenant principalement dans la première phase, en détaillant les différents modules qui entrent en jeu lors de la détermination du but utilisateur. Nous allons notamment détailler comment le système de dialogue est enrichi en utilisant le contexte de l'interaction et la base de faits maintenue par le système d'évaluation de la situation.

Nous considérons ici une interaction pouvant être modélisée par une alternance de "tours" entre les deux participants. Cet enchaînement de tours est appelé cycle du dialogue. Un tour représente le laps de temps pendant lequel un des participants peut s'adresser à l'autre pour faire progresser le dialogue.

La plupart des travaux sur les systèmes de dialogue [Levin 1997, Young 2010,

Thomson 2010] utilisent une architecture incorporant :

— Des modules de compréhension pour interpréter la parole générée par l'interlocuteur.

— Un gestionnaire de dialogue lié à une base de données afin de prendre une décision pour continuer l'interaction.

— Des modules de générations permettant de restituer la décision à l'interlocuteur.

Ce type d'architecture est illustré par la figure 4.5.

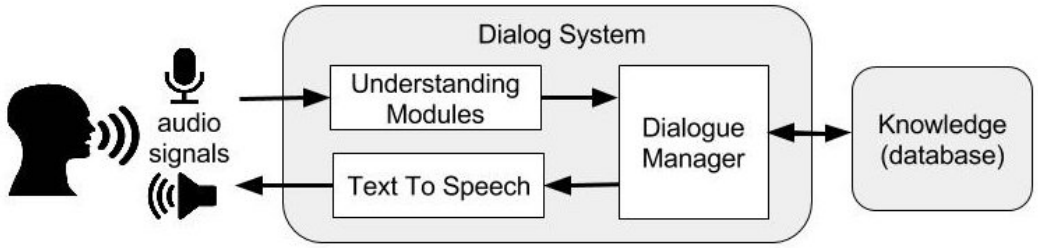


FIGURE 4.5 – Architecture usuelle d'un système de dialogue.

Notre système de dialogue a la particularité d'être un système multimodal, c'est à dire dans lequel la gestuelle et la parole sont utilisés, et situé, c'est à dire que le contexte physique de l'interaction doit être pris en compte. Nous avons donc incorporé dans les entrées du système le contexte produit par TOASTER. Nous présentons à la figure 4.6, l'architecture détaillée mise en place pour permettre la communication avec l'utilisateur. Nous nous baserons sur cette figure pour expliquer les différents flux, ainsi que les différents modules qui permettent le bon fonctionnement du dialogue situé. Cette architecture est issue de la collaboration du LIA et du LAAS-CNRS et est également présentée dans la thèse d'Emmanuel Ferreira dans [Ferreira 2015a].

Il est possible d'identifier plusieurs sous parties dans cette architecture. La première est en charge des entrées multimodales de l'utilisateur. Une deuxième partie est responsable de l'acquisition et du maintien de l'état du monde ainsi que de la génération de données symboliques. Le gestionnaire de dialogue, ou DM (Dialogue Manager) permet de décider, en fonction des entrées interprétées de l'utilisateur et du contexte de l'interaction de la réponse donnée par le robot afin de parvenir à qualifier la demande de l'utilisateur. La dernière partie s'attache à restituer sous forme multimodale la sortie du système de dialogue décidée par le gestionnaire de dialogue. Certains modules relevant d'avantage du dialogue proprement parlé

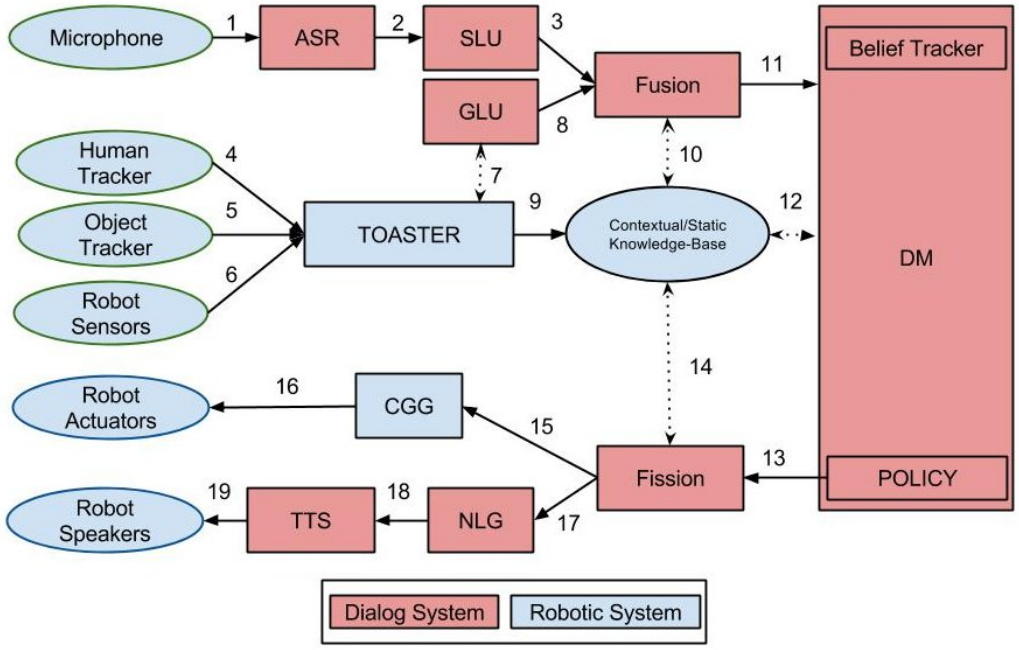


FIGURE 4.6 – Architecture détaillant les différents composants du système de dialogue situé élaborée pour le projet MaRDi.

que de l'usage du système d'évaluation de la situation, une description brève en sera donnée. Pour plus de renseignements sur ces modules et une description plus exhaustive de la partie dialogue, nous invitons le lecteur à consulter les articles [Ferreira 2013b, Ferreira 2013a] ainsi que la thèse [Ferreira 2015a].

4.3.1 Entrées multimodales utilisateur

Cette première partie permet d'acquérir et d'interpréter les entrées multimodales de l'homme en fonction du contexte de l'interaction. L'utilisateur peut faire l'usage de la parole et/ou de gestes déictiques de façon non contrainte tout au long de l'interaction pour s'adresser au robot. Une fois la commande de l'utilisateur interprétée, elle est transmise au module de gestion de dialogue.

Pour acquérir et interpréter les commandes de l'utilisateur, quatre modules sont impliqués.

— **Le module *ASR*** : la reconnaissance de la parole est effectuée grâce au module *ASR* (Automatic Speech Recognition). Il utilise le flux audio (flux 1 sur le schéma) provenant du microphone pour traduire ce flux en mots. Ce module est basé sur la Google Web Speech API 3.

— **Le module *SLU*** : l'*ASR* fournit les mots possiblement prononcés par

l'homme au *SLU* (2). Ces mots sont interprétés par le *SLU* (Spoken Language Understanding) pour permettre de relier les mots aux concepts mis en jeux.

— **Le module *GRU*** : ce module est chargé de la reconnaissance et la compréhension des gestes déictiques émis par l'utilisateur lors de son tour d'interaction (Gesture Recognition and Understanding). Les gestes de pointage sont détectés et interprétés dynamiquement par notre raisonneur spatial (*TOASTER*) décrit au chapitre 2 de cette thèse. Ce dernier exploite à la fois les coordonnées spatiales des objets (5) et les données articulaires de l'utilisateur (4) telles que déterminées grâce aux informations issues des capteurs visuels du robot pour savoir si un objet est pointé par l'utilisateur. Lorsque c'est le cas, un *fait* de la forme *AGENT_ID pointsAt OBJECT_ID* est alors généré et transmis au GRU (7). De plus, ce *fait*, comme tous les faits générés par notre infrastructure de raisonnement géométrique, contient un indicateur temporel. Cela permet de simplifier le mécanisme de fusion avec les entrées vocales. Bien que les composants aient été prévu pour s'interfacer de la façon décrite, dans le cadre du projet MaRDi le lien (7) n'a pas pu être établi, par manque de temps, lors de l'intégration du système.

— **Le module de fusion** : l'objectif du mécanisme de fusion est de combiner les actes de dialogue extraits du signal de parole utilisateur (3) aux événements déictiques capturés grâce au module *GRU* (8). Pour ce faire, il faut tenir compte à la fois du contexte de l'interaction (positions des objets dans l'environnement physique, etc.), du niveau de confiance que l'on porte aux différentes hypothèses unimodales (étant donné qu'elles peuvent être erronées) mais également à leur marqueur temporel. La première étape de ce processus consiste donc à déterminer si les hypothèses en provenance des différentes modalités sont synchrones entre elles et peuvent être fusionnées ou doivent être considérées séparément. Du fait que la parole est considérée dans notre étude comme modalité principale de l'utilisateur, les tours d'interaction seront calés sur celui des entrées vocales. Ainsi, comme dans [Holzapfel 2004], seuls les gestes déictiques détectés dans un segment temporel de 20ms avant et après celui du tour de parole courant seront exploités par le mécanisme de fusion. De ce fait, si des entrées *SLU* apparaissent seules ou que les gestes détectés ne leur sont pas synchrones, elles seront directement considérées comme résultat de la fusion. La méthode de fusion retenue ici repose sur la définition d'un ensemble de règles. Par exemple si l'utilisateur prononce la phrase « prends ça » tout en désignant un objet du doigt la fusion a pour rôle principal d'identifier un candidat valable. Pour ce faire le

Phrase utilisateur	Passe moi le livre bleu qui est sur la table du salon.
Sémantique bas niveau	inform(action=give, type=book, color=blue, spatial.indicator=on, type=table, room=livingroom)
Regroupement	inform(cmd.action=give, object.type=book, object.color=blue, object.location=[spatial.indicator=on, type=table, room=livingroom])
Sémantique haut niveau	inform(cmd.action=give, object.type=book, object.color=blue, object.position=livingroom_table)

TABLE 4.2 – Exemple de phrase utilisateur et de l'interprétation qui en est faite par le système.

mécanisme de fusion s'appuie notamment sur la détection de concepts bas niveau qui témoignent d'un besoin de résolution de référents dans l'énoncé utilisateur (le mot « ça » dans l'exemple précédent).
La dernière étape du processus consiste à convertir les hypothèses ainsi produites dans leur représentation sémantique haut niveau pour pouvoir les transmettre au gestionnaire de dialogue (11). Des heuristiques définies manuellement sont employées pour déterminer les valeurs des concepts de haut niveau identifiés à partir des hypothèses bas niveau.

Pour illustrer les mécanismes de compréhension des entrées utilisateur, un exemple de phrase utilisateur et de son interprétation par le système est donné au tableau 4.2.

4.3.2 Le contexte comme aide à la compréhension

Comme déjà présenté au chapitre 2, la modélisation du contexte de l'interaction joue un rôle déterminant dans une application HRI. Grâce à elle le robot peut modéliser dynamiquement l'environnement géométrique avec lequel il est en train d'interagir et ainsi en avoir une représentation symbolique adaptée au raisonnement logique. Dans notre configuration le système dispose à la fois d'une base statique de connaissances contenant la liste de tous les objets connus du robot (même ceux non encore perçus durant l'interaction) et de leurs propriétés statiques (couleur, identifiant, etc.) mais aussi d'une base dynamique de connaissances dans laquelle sont stockées les informations contextuelles et la représentation symbolique de l'environnement.

Ainsi, si l'homme fait référence à un objet en le décrivant par rapport à sa position relative à un autre objet, en utilisant les faits produits décrits en section 2.4.2.2

tel que *isNextTo*, *isIn* ou *isOn*, il est possible sinon d'identifier l'objet, au moins de réduire la liste des candidats potentiels. De même les descriptions de positionnement relatives (gauche, droite) des objets permettent de discriminer les candidats potentiels parmi les entités présentes pouvant correspondre à une référence prononcée par l'homme. En utilisant une base de donnée SQL tel que décrite en section 2.6, il est possible de faire des jointures et donc d'identifier les objets correspondant aux critères définis dans la requête de l'utilisateur. Ainsi, si l'homme parle d'un objet vert situé sur la table du salon, en une seule requête à la base de données combinant ces deux propriétés, la base de données renvoie l'ensemble des objets correspondant à ces critères. De la même manière, le robot peut choisir de parler d'un objet en utilisant ces mêmes faits pour le décrire. Ceci permet au robot de comprendre et d'identifier les références faites par l'homme et d'utiliser des références naturelles et compréhensibles pour l'homme.

Pour aller plus loin dans la compréhension, le robot doit aussi considérer l'homme comme une entité logique faisant des requêtes logiques. En partant de ce postulat, il est possible d'exploiter les données issues du raisonnement de prise de perspective perceptuelle décrits en section 3.3.1. Prenons l'exemple où l'homme demande au robot d'apporter un objet et que le robot hésite entre deux objets, l'objet A et l'objet B. L'une des différences entre ces deux objets est que l'objet A est devant l'homme. Dans ce cas le robot doit être capable de comprendre que l'homme ne réclamerait pas un objet qui lui est déjà accessible et donc apporter directement l'objet B.

De même, si l'homme demande la position d'un objet et que le robot identifie deux candidats (A et B), et que l'objet A est visible de l'homme, le robot doit être capable de comprendre que l'objet recherché par l'homme est celui qui ne lui est pas perceptible, c'est à dire l'objet B.

Ces données issues de la prise de perspective perceptuelle permettent donc un raisonnement de haut niveau basé sur des règles logiques liées aux requêtes de l'homme et permettent au module de fusion d'identifier plus rapidement l'objet requis par l'homme. Cette capacité réduit donc le nombre de tours et améliore ainsi l'efficacité du système de dialogue.

De même, la capacité de prise de perspective conceptuelle gérée par le système d'évaluation de la situation et décrite en section 3.3.2 est une capacité importante pour améliorer la qualité du dialogue situé. En effet, les requêtes de l'homme doivent être interprétées dans son état mental car l'homme exprime sa requête en fonction de son état de croyance sur la situation de l'environnement.

Nous décrivons dans la partie suivante, en section 4.3.3.2, comment nous utilisons cette représentation de l'état de croyance pour améliorer le module de gestion

du dialogue.

4.3.3 Gestionnaire de dialogue

4.3.3.1 Présentation générale

Le composant responsable de la gestion du dialogue, noté DM pour Dialogue Manager dans la figure 4.6, est basé sur le paradigme POMDP (Processus de Décision Markovien Partiellement Observable) HIS (Hidden Information State) [Young 2010] qui a été adapté ici pour l'interaction multimodale. Dans cette configuration, l'état de croyance du système de dialogue est représenté par un ensemble de partitions. Chaque partition représente une commande utilisateur possible. La prise de décision se fait sur un état résumé pour permettre l'apprentissage par renforcement. À chaque tour, le système choisit une action résumée (par exemple inform, confirm, execute). Pour ce qui est de la politique du gestionnaire de dialogue, l'algorithme KTD-SARSA RL [Daubigney 2012] a été utilisé. Plus de détails sur cette configuration sont disponibles dans les articles [Ferreira 2013b, Ferreira 2013a].

Une des limites du paradigme HIS pour le problème qui nous concerne est qu'il n'offre dans sa version initiale que des mécanismes capables de gérer l'incertitude due aux bruits présents dans le canal de communication (reconnaissance puis compréhension de la parole). Or, nous pensons qu'une autre source possible d'incertitude peut provenir de situations de fausses croyances où la croyance en des faits erronées (par exemple une position antérieure d'un objet) viendrait bruiter les actes communicatifs de l'utilisateur et provoquer une difficulté supplémentaire de compréhension pour le système de dialogue. En effet, si l'état mental de l'utilisateur n'est pas modélisé ni pris en compte, seul des mécanismes de résolution classique de l'incertitude peuvent être appliqués par la politique du gestionnaire de dialogue, par exemple demander à l'utilisateur de confirmer des hypothèses jusqu'à ce que sa demande corresponde à la réalité observée, et ce même dans des situations où il aurait été possible d'interpréter correctement la commande de l'utilisateur en utilisant la modélisation de son état de croyance.

4.3.3.2 Prise en compte de l'état mental

Pour répondre à la problématique soulevée ci-dessus, nous avons collaboré avec le LIA (Laboratoire d'Informatique d'Avignon) afin de proposer d'intégrer l'analyse sur les croyances factuelles du robot et de l'utilisateur directement dans le mécanisme de prise de décision du système HIS, pour améliorer la qualité et l'efficacité du dialogue. Ainsi, nous proposons d'augmenter l'état résumé du dialogue avec un état sur la croyance divergente, nommé le d-status. Ce dernier est utilisé pour

notifier de la présence d'une situation de fausse croyance. Dans notre cadre expérimental actuel, ces croyances sont gérées par le module d'évaluation de la situation, comme expliqué au chapitre 3. Elles sont donc maintenues de façon indépendante aux mécanismes de gestion de l'état interne du système de dialogue. Nous proposons également dans cette extension d'ajouter une nouvelle action dédiée à la résolution des situations de fausses croyances, *InformDivergentBelief*.

Nous prenons comme exemple, un scénario ou deux livres ont étés inter-changés sans que l'homme en soit informé. Cette situation est résumée par la figure 4.7.

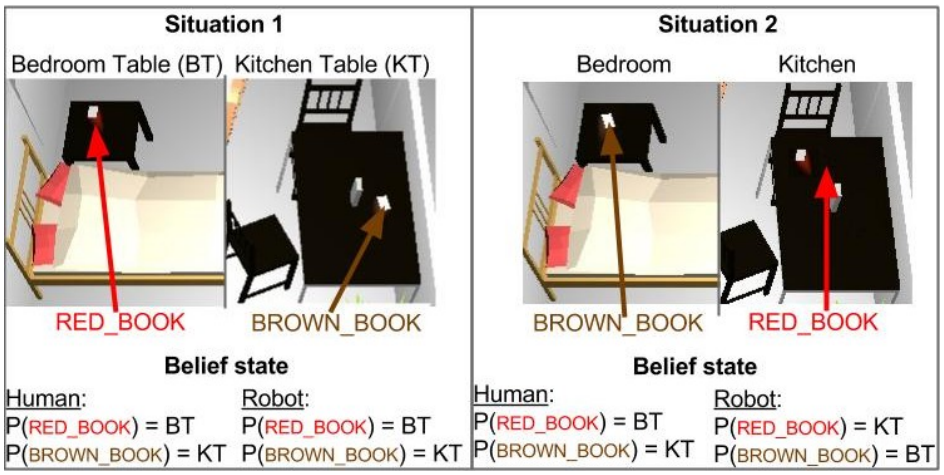


FIGURE 4.7 – Exemple où l'homme a une croyance divergente concernant l'emplacement de deux livres ayant étés interchangés.

Nous illustrons la prise en compte de la croyance divergente dans cet exemple avec la figure 4.8. Les modifications apportées au modèle initial sont illustrées par les éléments en orange.

Soit la situation suivante : l'utilisateur vient de prononcer la phrase « donne moi le livre qui est sur ma table de chevet ». Comme le montre la figure 4.8, logiquement la partition de plus grande probabilité devient celle qui modélise le but utilisateur qui consiste à vouloir lui « apporter un livre situé sur la table de chevet ». Du point de vue du robot (ROBOT FACTS sur la figure) ce livre est identifié de façon unique comme étant l'objet RED_BOOK. Cependant, du point de vue de l'utilisateur (USER FACTS sur la figure), il est également identifié de façon unique comme étant l'objet BROWN_BOOK. Cette situation est considérée comme divergente et le d-status est réglé sur unique parce qu'il n'y a qu'un seul objet possible qui correspond à cette description dans le modèle de l'utilisateur et que ce dernier est différent de celui également identifié dans la base de faits du robot. Dans nos travaux, le d-status ne peut prendre que les deux valeurs *unique* et *other*, nous considérons cependant cet élément comme étant catégoriel et non binaire car nous

comptons étendre à terme le nombre des valeurs ainsi considérées (identification de différentes situations comme par exemple le fait que plusieurs objets candidats présentent une position divergente). En ce qui concerne la nouvelle action résumée *InformDivergentBelief* introduite pour résoudre la situation de divergence, les actes de dialogue qui lui sont associés sont déterminés par l'intermédiaire d'heuristiques expertes. Dans cette première version, lorsqu'un cas de divergence est détecté, idéalement l'action prise par le système doit permettre d'informer l'utilisateur de manière explicite de la présence et de la nature de cette divergence. Pour ce faire, un acte de dialogue est employé pour informer l'utilisateur sur l'existence d'une divergence quant à la valeur de la position de l'objet "sujet" de la commande. Grâce à l'émission de cet acte de dialogue l'utilisateur va pouvoir mettre à jour ses croyances avant de poursuivre son objectif initial. Ainsi, lorsque le système informera l'utilisateur oralement de la véritable position de l'objet en question, le modèle de croyance utilisateur sera mis à jour en fonction. Ce processus est également illustré dans la figure 4.8 lorsque l'action *InformDivergentBelief* est sélectionnée en tant que prochaine action du système (action maître). L'action finalement exécutée dans l'espace maître sera : *deny(object.location=bedside_table, object.location=kitchen_table, object.type=book, object.color=brown)*, qui sera transformée par le module de Génération de Language Naturel *NLG* dans l'énoncé système suivant "le livre marron n'est plus sur la table de chevet mais sur la table de la cuisine".

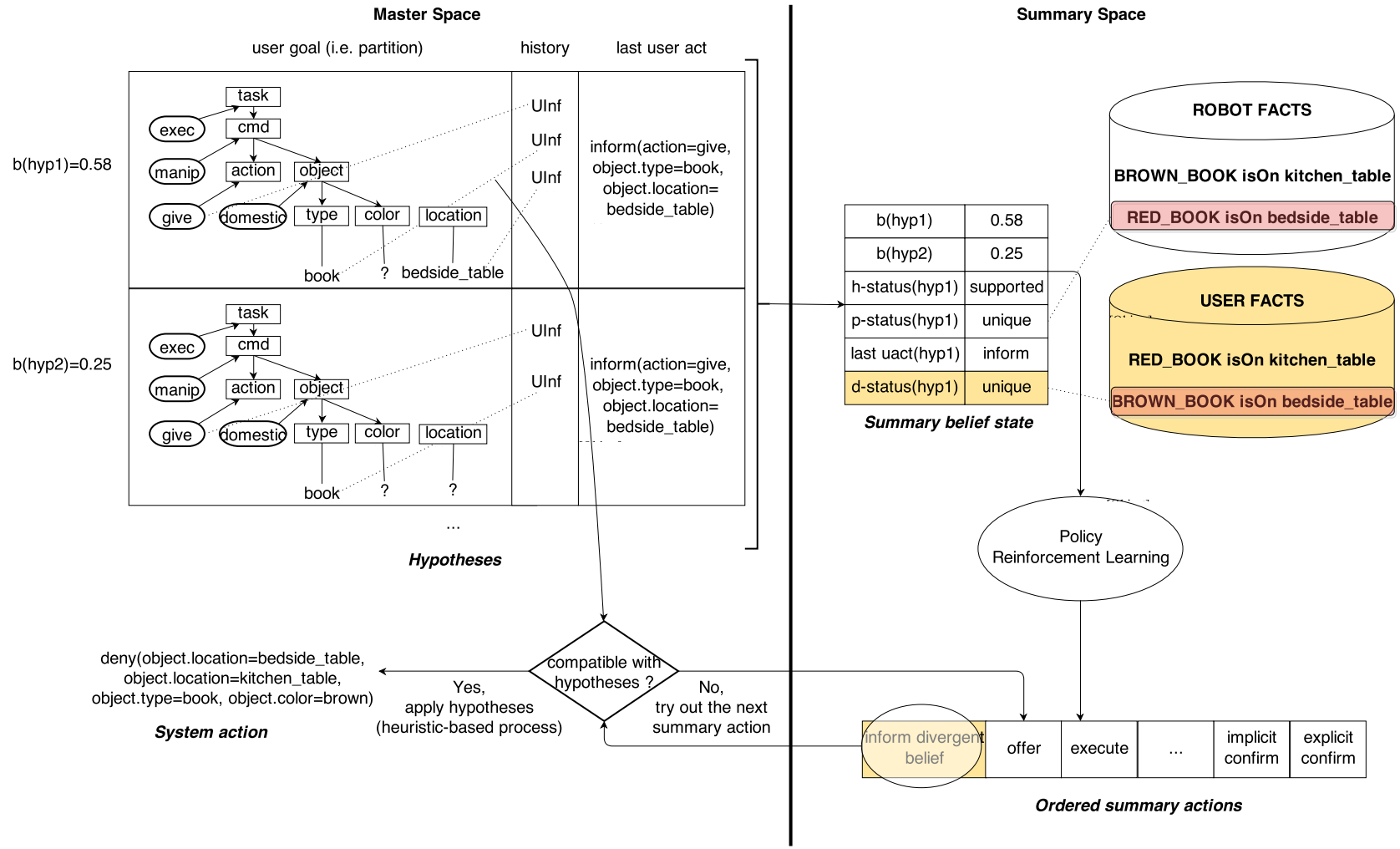


FIGURE 4.8 – Aperçu de l'extension apportée au système HIS pour prendre en compte les croyances divergentes, graphique réalisé par Emmanuel Ferreira dans le cadre du projet MaRDi [Ferreira 2015a].

4.3.4 Restitution multimodale

Le robot utilise quatre modules pour gérer la sortie du système afin de transmettre à l'homme la commande choisie par le gestionnaire de dialogue.

— **Le module de fission** : il a en charge le processus de traduction des décisions abstraites (actes de dialogue haut niveau) du système vers des actions verbales et non-verbales (déplacement, prise de position). Pour l'instant, ce module est basé sur la définition d'un ensemble de règles prenant en compte la nature de la décision du système (flux 13) et le contexte courant (14). La solution retenue considère également les flux de sorties comme parallèles (pas de synchronisation fine entre gestes et paroles).

— **Le module *NLG*** : ce module est responsable de la génération de langage naturel. Il s'appuie sur des patrons lexicaux pour générer un texte qui soit compréhensible de l'homme afin de le transmettre au *TTS* (18).

— **Le module *TTS*** : ce module est responsable de la traduction de texte en parole (Text To Speech), c'est à dire qu'il doit générer un signal sonore correspondant au texte reçu du *NLG* afin qu'il soit émis par les haut-parleurs du robot (19). Ce module a été spécialement implémenté par notre partenaire ACAPELA Group dans le cadre du projet MaRDi. Son originalité réside dans le fait qu'il repose sur des mécanismes d'interpolations de modèles pour élargir la richesse expressive de la voix employée tout en offrant un contrôle continu pour moduler dynamiquement la voix au cours de la synthèse d'un même énoncé [Astrinaki 2012]. Dans sa version actuelle nous pouvons donc jouer sur trois paramètres simultanément, à savoir le style de voix (portée, chuchotée ou normale), l'émotion transmise (ton joyeux, triste ou normal) et la vitesse d'élocution (rapide, lente, normale). Selon la nature de l'acte de dialogue sélectionné par le système et le contexte interactif, le module de fission va donc attribuer une étiquette sur l'acte vocal pour que le module *TTS* puisse faire une synthèse expressive de la phrase générée par le *NLG*. Par exemple, si l'utilisateur et le robot ne sont pas dans la même pièce le module de fission va obtenir cette information du contexte (14) et va attribuer l'étiquette indiquant qu'il va falloir que le robot parle plus fort (avec une voix portée), ou encore si le système informe l'utilisateur qu'il ne peut pas réaliser l'action (par exemple si l'objet est hors de porté pour lui) alors il pourra faire jouer une synthèse vocale employant une voix triste.

— **Le module *CGG*** : ce module est responsable de la génération des gestes communicatifs (Communicative Gestures Generation). Cela permet d'ajouter de l'expressivité aux réponses du robot en ajoutant une séquence de

gestes expressifs pendant que le robot parle. Un ensemble de mouvements correspondants à un panel d'expression a été généré hors ligne. La fission choisit s'il est pertinent d'ajouter un geste expressif à la parole en fonction du contexte (visibilité du robot par l'homme, nécessité d'une gestuelle pour améliorer l'expressivité de la parole) et envoi le cas échéant l'expression à jouer (15). Le CGG se contente d'envoyer de façon séquentielle les différents mouvements correspondant à l'expressivité gestuelle souhaitée. Suite à un manque de temps ce module n'a pas pu être intégré à l'architecture.

4.4 Implémentation dans un simulateur robotique

4.4.1 Motivation

Les systèmes de simulation sont très utilisés en robotique. Ils permettent aux roboticiens d'évaluer et valider leur travaux au niveau d'abstraction souhaité. De cette manière, les projets reposant sur des calculs de haut niveau (interaction, dialogue, supervision) peuvent utiliser un simulateur pour abstraire les niveaux inférieurs (navigation, traitement d'image, manipulation) et éviter que les problèmes qui leur sont liés interfèrent durant l'interaction.

Pour le projet MaRDi, le simulateur est également utile pour partager un même environnement et une même configuration expérimentale entre les différents partenaires. De plus, le système de dialogue reposant sur un gestionnaire de dialogue ayant une politique apprise par renforcement, l'usage de la simulation permet d'effectuer cet apprentissage avec des moyens très réduits [Milliez 2014b].

Il est cependant nécessaire que le simulateur soit adapté à l'interaction homme-robot. En terme de simulation, deux solutions sont possibles pour ajouter l'homme à la boucle : 1) en modélisant et implémentant leurs comportements et actions, et 2) en utilisant la téléopération pour contrôler les avatars humains.

La première solution présente l'avantage de l'automatisation et ne demande aucune manipulation manuelle. Cette solution est donc moins coûteuse en temps et plus facile à mettre en place. Cependant, selon les caractéristiques humaines requises, il est potentiellement extrêmement complexe d'avoir un modèle de comportement humain réaliste. Les humains sont des entités complexes avec des réactions et comportements quasi impossible à synthétiser de manière satisfaisante. Cette solution est en général retenue pour les études qui n'impliquent pas les comportements humains les plus complexes, comme la navigation ou la manipulation.

Dans la seconde solution, un homme télé-opère un avatar virtuel. La simulation est donc plus complexe à établir car elle nécessite de monopoliser un humain.

Cependant, l'avatar du simulateur aura un comportement beaucoup plus réaliste. Pour ce faire, l'environnement doit avoir un rendu visuel réaliste et le contrôle de l'avatar doit être suffisamment intuitif.

D'autres projets de dialogue situé homme-robot reposent sur une étude dans un simulateur. Par exemple pour un scénario de Pick-Place-Carry (Prendre-Placer-Transporter) [Lucignano 2013], de robot barman [Stiefelhagen 2007] ou de navigation dans un environnent virtuel [Byron 2006]. Cependant, peu de travaux considèrent l'environnement de simulation comme moyen pour l'acquisition du corpus de dialogue situé ou comme moyen de tester l'apprentissage de politique en ligne. En effet, la plupart repose sur des expériences en magicien d'Oz [Prommer 2006, Stiefelhagen 2007, Rieser 2008].

4.4.2 Choix du simulateur

Dans le domaine de la robotique, de nombreux simulateurs sont disponibles. Nous pouvons citer la suite Player/Stage/Gazebo [Rusu 2007], la plateforme de simulation intégrée OpenHRP [Nakaoka 2007], l'architecture logicielle multiplateforme OpenRAVE [Diankov 2010] ou même le simulateur commercial V-REP [Freese 2010]. Cependant, seuls quelques-uns d'entre eux sont adaptés à l'interaction homme-robot. Ils limitent généralement les comportements des agents humains à des mouvements et des capacités d'interaction relativement simple, ce qui constitue l'une des raisons pour laquelle les simulations de HRI ont été menées jusqu'à présent dans des configuration de *télé-opération*, où seul le robot et l'environnement (et pas l'agent humain) sont en fait simulés. Les simulateurs robotiques USARSim [Lewis 2007] et MORSE [Echeverria 2012, Lemaignan 2014] sont tous deux utilisés dans des dizaines d'études HRI en raison de leur capacité à contrôler un agent humain. Cependant, MORSE présente plusieurs avantages spécifiques qui ont motivé notre choix.

MORSE est un simulateur open-source, avec une communauté très active, qui a été développé spécifiquement pour la simulation robotique. Il prend en charge une large gamme de middleware (par exemple ROS, YARP, pocolibs) ainsi que des implémentations fiables et réalistes des capteurs et des actionneurs qui facilitent l'intégration sur de véritables plateformes robotiques. En outre, MORSE offre une configuration de simulation adaptable en permettant à des robots virtuels d'interagir avec l'environnement virtuel en utilisant soit des capteurs/actionneurs réalistes ou des capteurs/actionneurs ayant un niveau d'abstraction supérieur. De ce fait, les roboticiens peuvent contrôler le coût de calcul de traitement des données de bas niveau en exploitant les sorties de haut niveau à partir de composants "irréalistes".

Par exemple, Morse fournit à la fois une caméra de vision et un capteur dit "caméra sémantique". Alors que la première caméra fournit une image brute (pixels) en tant que sortie, la seconde donne directement les noms des objets perçus et leurs positions dans la scène. Ce dernier capteur évite pour les utilisateurs à devoir effectuer la reconnaissance et les processus de localisation d'objets lors de la focalisation sur les problématiques de niveau supérieur.

De plus, MORSE repose sur "Blender Game Engine", un moteur d'exécution 3D en temps réel intégré au toolkit de modélisation open-source Blender, à la fois pour la 3D avancé (OpenGL Shader) et pour la simulation de la physique (basée sur le moteur physique BULLET). Cette configuration permet un rendu réaliste d'environnement complexe et fournit une interface utilisateur graphique immersive, qui est une caractéristique nécessaire pour la modélisation HRI.

Dans MORSE, l'avatar humain peut être contrôlé par un opérateur humain ou directement par le biais de scripts externes comme tout autre robot.

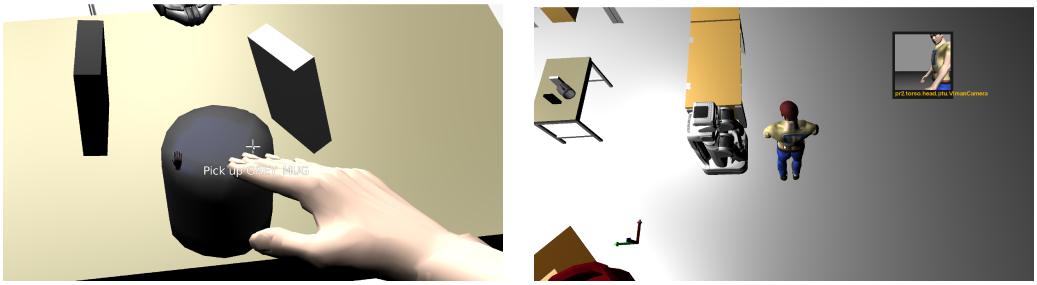


FIGURE 4.9 – Avatar humain attrapant un objet, contrôlé par un opérateur (image de gauche) et l'avatar humain en vue à la troisième personne (image de droite).

Dans le premier cas, l'opérateur contrôle l'humain virtuel de manière immersive (voir la figure 4.9) en termes de déplacement, de regard, et des interactions sur l'environnement, tel que la manipulation d'objets (par exemple, saisir ou déposer un objet).

Dans le second cas, l'avatar est contrôlé par programme en utilisant des actionneurs MORSE standard. A titre d'exemple, il est possible d'utiliser un actionneur "waypoint" (point de cheminement) sur l'humain pour définir un trajet qu'il doit suivre.

4.4.3 Simulation de l'interaction

La simulation de l'interaction nous permet dans notre cas d'obtenir rapidement un moyen d'entraîner le système de dialogue et de tester la validité de notre architecture pour la phase de détermination de but décrite en 4.2.1. Les phases d'élaboration

de but et d'exécution seront donc simplifiées au maximum.

Dans notre scénario, une personne handicapée est dans son appartement de trois pièces et possède un robot pour l'assister dans les tâches quotidiennes. Le but ici est de faire en sorte que le système robotique puisse comprendre, en raisonnant sur les paroles de l'homme, les gestes et l'environnement, la requête de l'homme concernant les objets. Les objets sont limités ici aux objets qui peuvent être saisis comme des livres, des DVDs et des tasses, qui ont divers couleurs, type, position et un identifiant unique.

Nous utilisons le robot Pr2 dans le simulateur car c'est la plateforme que nous utilisons au LAAS-CNRS. Le Pr2 est déjà présent dans les modèles de MORSE, le rendant directement utilisable. Nous avons ajouté une caméra symbolique (MORSE semantic camera) au modèle standard afin qu'il puisse réaliser la reconnaissance d'objet. Nous avons également ajouté un actionneur appelé "teleport" pour bouger le robot à une position donnée en évitant ainsi d'avoir à gérer la navigation et en gagnant le temps du déplacement. Nous utilisons un modèle virtuel de l'environnement physique dans lequel le robot réel sera testé. Enfin, nous ajoutons un avatar humain. La configuration expérimentale est schématisée à la figure 4.10. L'humain utilise le clavier (1) pour contrôler son avatar et a un retour visuel sur l'écran (2). Un casque audio avec micro intégré est utilisé pour dialoguer avec le système (3 et 4). Les flux 5 et 6 seront détaillés en section 4.4.4.

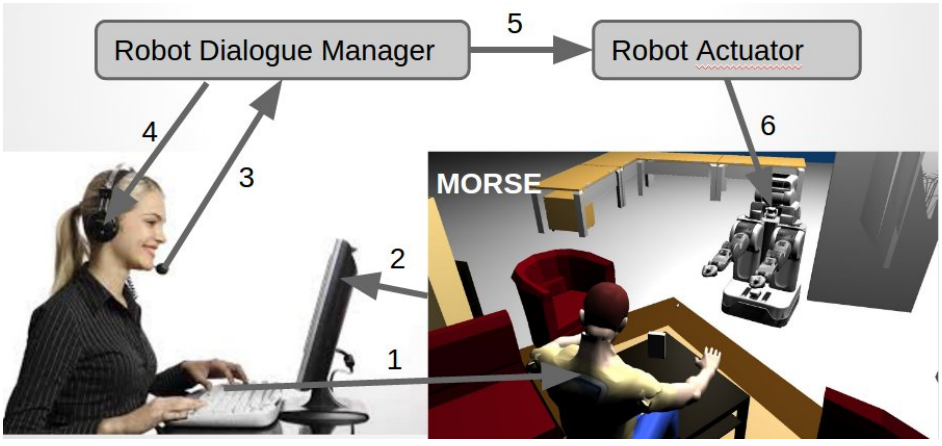


FIGURE 4.10 – Configuration expérimentale pour la simulation dans morse avec un agent humain contrôlant l'avatar du simulateur.

Au début de la simulation, un script positionne des objets au hasard dans des zones prédéfinies (comme la table de la cuisine, celle du salon, l'étagère de la chambre...). Ces zones sont appelées des zones de manipulation. Cela nous permet d'utiliser différentes configurations d'environnement sans changer le fichier d'initia-

lisation (MORSE builder script).

4.4.4 Simulation de l'exécution

Pour permettre aux utilisateurs d'évaluer l'accomplissement de sa requête (par exemple, est-ce que le robot rapporte le bon objet), nous avons développé une librairie abstraite d'actions haut niveau réalisable par le robot. Ces actions haut niveaux seront combinées pour accomplir l'objectif énoncé par l'utilisateur. Leur haut niveau d'abstraction permet ici de pouvoir accomplir le but utilisateur sans avoir recours au planificateur ou à la supervision et donc de contourner les phases deux et trois décrites en section 4.2.2 et 4.2.3.

La liste des actions abstraites est la suivante :

— *Move* : pour explorer l'environnement ou rapporter un objet, le robot a besoin de se déplacer vers les zones de manipulation. Pour ce faire, nous utilisons l'actionneur *teleport* de MORSE. Cet actionneur permet de déplacer instantanément le robot à l'endroit désiré. Nous avons défini une fonction de script pour bouger le robot vers chaque zone de manipulation définie. De cette façon, le robot peut se rendre à chaque position pour prendre un objet ou explorer une zone pour obtenir des informations contextuelles.

— *Explore* : le robot est capable de scanner une zone de manipulation. Pour rendre cette action possible, une caméra symbolique a été ajoutée à la tête du robot. La tête se déplace séquentiellement pour scanner l'environnement.

— *Grab* : le robot peut prendre un objet. Pour réaliser cette action, le service *grasp* lié au Pr2 de MORSE est utilisé. Nous précisons le nom de l'objet qu'il doit attraper et si l'objet est proche de la pince du robot, il sera placé au niveau de celle-ci et lié à son déplacement. De la même façon, nous avons ajouté une fonction *Drop* qui prend comme paramètre la zone dans laquelle l'objet doit être déposé. Le robot relâchera alors l'objet sur le meuble correspondant.

— *Give* : la dernière action est de donner l'objet à l'humain. Pour ce faire, le robot se déplace vers la position de l'homme puis déploie son bras vers l'homme afin de lui remettre l'objet. Nous utilisons simplement l'armature du robot pour contrôler son bras.

Ces actions haut niveau permettent d'effectuer des expérimentations avec un système réalisant la détermination de la demande utilisateur jusqu'à la réalisation et l'exécution d'actions, tout en nous focalisant sur la partie dialogue situé (dans l'univers simulé). Cela rend possible d'entraîner le système de dialogue et de faire des premières évaluations de façon bien plus rapide et moins coûteuse en terme de

ressources requises.

4.4.5 Expérimentation et résultats

Une première étude a été menée pour montrer la capacité du système à apprendre la politique du gestionnaire de dialogue. Cette étude a permis de montrer que le système de simulation permet d'entraîner et d'améliorer significativement et avec un coût réduit le système de dialogue. Les détails et résultats de cette expérimentation disponibles dans [Milliez 2014b]. Cette étude a permis de montrer qu'il est possible de réaliser un apprentissage de la stratégie du gestionnaire de dialogue à partir du simulateur morse. Les détails de cette première étude ne seront pas discutés ici afin de nous focaliser sur l'aspect gestion de croyance.

La seconde expérimentation menée sur le simulateur robotique morse concerne l'évaluation de l'amélioration du système de gestion de dialogue situé multimodal lors de la prise en compte des croyances erronées, comme présenté en section 4.3.3.2. Nous nous focalisons donc sur des scénarios où une divergence de croyance est présente entre le modèle d'état d'esprit de l'homme et celui du robot. Dans une configuration réaliste, ces scénarios nécessitent des interactions d'une certaine durée pour faire apparaître une croyance divergente. Pour contourner ce processus coûteux en temps dans notre configuration d'évaluation, nous proposons directement un but erroné à l'utilisateur au début de l'interaction. Par conséquent, une fausse croyance sur la localisation d'un objet non visible de l'homme est automatiquement ajoutée. Même si cette situation est créée de façon artificielle, le même comportement peut être obtenu en utilisant notre système d'évaluation spatiale, comme présenté en section 3.5.2.

Par conséquent, cette configuration a été utilisée pour évaluer la capacité de notre système à gérer à la fois les tâches de manipulation classique (CLASSIC) et de fausse croyance (FB pour False Belief). Pour ce faire, nous comparons un système ayant connaissance des croyances de l'homme (BA pour Belief Aware) avec un système n'ayant pas de gestion de croyance (BU pour Belief Unaware). Ces deux systèmes ont des politiques apprises par l'interaction avec deux utilisateurs experts ayant réalisé 60 dialogues.

Dans la configuration d'évaluation, 10 dialogues ont été enregistrés pour *BA* et *BU*, ces dialogues provenant de 6 sujets distincts (deux femmes et quatre hommes d'environ 25 ans), soit au total 120 dialogues. 30% de ces dialogues impliquaient des tâches de type *FB*. Les sujets n'avaient pas connaissance de la configuration du système, les tests ayant été présentés dans un ordre aléatoire. À la fin de l'interaction, les utilisateurs évaluent le système en terme de réussite de la tâche avec un

	BU			BA		
TASK	*Avg.R*	*Length*	*SuccR*	*Avg.R*	*Length*	*SuccR*
CLASSIC	17.62	2.95	0.93	17.69	2.88	0.93
FB	12.72	5.94	0.83	13.89	4.78	0.83
ALL	16.15	3.85	0.9	16.55	3.45	0.9

TABLE 4.3 – Performance du système sur des tâches classiques (CLASSIC), de fausse croyance (FB) et globales (ALL) en terme de moyenne sur les récompenses cumulées (Avg.R), de longueur moyenne des dialogues en nombre de tours (Length) et de taux de succès moyen (SuccR).

R_1 : Puis-je vous aider ? U_1 : Apporte moi le livre qui est sur ma table de nuit.

(a)

R_2 :	**Le livre marron n'est plus sur la table de nuit, il a été déplacé vers la cuisine.**
U_2 :	D'accord, apporte le moi
R_3 :	Je vous apporte le livre marron qui est sur la table de la cuisine.

(b)

R_2 :	Voulez-vous le rouge ?
U_2 :	Non, le marron.
R_3 :	Il n'y a pas de livre marron dans votre chambre mais il y en a un dans la cuisine.
U_3 :	Tu es sûr ? Et bien, apporte moi celui-ci.
R_4 :	Je vous apporte le livre marron qui est sur la table de la cuisine.

TABLE 4.4 – Exemple de dialogue avec (a) et sans (b) le raisonnement sur les croyances divergentes dans le cas où l'humain n'est pas informé que les livres rouge et marron ont été inter-changés.

questionnaire en ligne.

Le tableau 4.3 est rempli avec les performances obtenues par les deux configurations discutées ci-dessus en considérant les tâches CLASSIC et FB. Ces résultats sont d'abord donnés en terme de moyenne sur les récompenses cumulées (Avg.R). D'après la définition de la fonction de récompense, cette métrique exprime dans une seule valeur les deux variables d'amélioration, le taux de succès (précision) et le nombre de tours avant la fin du dialogue (efficacité temporelle). Ces deux métriques sont également représentées. Les résultats dans le tableau 4.3 sont une moyenne des 60 dialogues réalisés pour chaque système.

Sur les tâches CLASSIC la performance entre BU et BA doit être considérée comme similaire. Ainsi, le mécanisme de résolution de croyance divergente ne semble pas influer sur la gestion du dialogue dans les situations sans croyance divergente. La politique testée est apprise, par conséquent le processus d'attribution d'action est optimisée avec un degré supplémentaire de complexité (plus grand espace état / action que dans BU), donc une perte aurait pu être observée.

Les performances entre BU et BA sur les tâches FB apparaissent en faveur de BA (taux plus élevé de réussite et un petit gain d'efficacité au niveau du temps du processus de gestion du dialogue : gain moyen d'un tour).

Pour avoir une meilleure estimation des différences entre les deux stratégies des

systèmes de dialogue BU et BA, nous avons également effectué une étude qualitative. Dans cette étude, nous étudions précisément les différences comportementales provenant de l'introduction du mécanisme de prise en compte des tâches FB dans un système ayant une politique apprise.

Dans le tableau 4.4 deux extraits de dialogues provenant des données d'évaluation illustrent les différences entre la gestion de dialogue BU et BA. sur la même tâche FB (ici le livre rouge a été inter-changé avec le marron). Si la divergence de croyance n'est pas prise en compte comme en (a), le gestionnaire de dialogue peut être forcé de gérer un niveau supplémentaire d'incompréhension comme en (b) de R_2 à U_3. Nous pouvons également observer en (b) que le système BU a été capable de réussir la tâche FB (ce qui explique que grâce à la politique apprise, le système BU a un taux de succès relativement élevé pour les tâches FB). En effet, si l'objet est clairement identifié par l'utilisateur, par exemple en utilisant la couleur ou le type, le système peut relâcher la contrainte de la position erronée et est donc capable de faire une offre d'exécution de la commande corrigée avec la positon valide de l'objet. Nous observons également que la politique apprise prenant en compte l'état de croyance fait émerger des mécanismes alternatifs à l'utilisation de l'action *informDivergentBelief*. Par exemple le système peut exécuter directement la tâche corrigée de l'utilisateur, ce qui évite les malentendus, lorsque les informations collectées semblent suffisantes pour identifier l'objet.

4.5 Implémentation sur plateforme robotique

Pour compléter cette première implémentation et ces tests sur simulateurs dans lesquels les phases deux et trois décrites en 4.2 ont été simulées afin de mener à bien l'apprentissage de la politique du système de gestion de dialogue [Milliez 2014b] et tester la prise en compte des croyances divergentes [Ferreira 2015b], nous avons implémenté et testé les éléments nécessaires pour la mise en place des phases deux et trois de la boucle d'interaction sur un Pr2. Cette étude a été réalisée avec l'aide des doctorants du LAAS-CNRS, Sandra Devin et de Michelangelo Fiore, pour la partie supervision.

L'environnement est constitué de trois pièces (*Bedroom*, *Livingroom* et *Kitchen*) correspondant à des zones, dans lesquels ce trouvent des zones de manipulation définies au niveau des différents meubles présents dans l'environnement.

Nous ajoutons le fait *hasScanned* qui prendra la valeur *true* lorsque le sujet du fait aura observé ce qui se trouve sur un meuble (qu'on appelle ici "*zone de manipulation*"). Le fait *hasScanned* sera également à *true* pour une pièce si toutes les zones de manipulation de cette pièce ont un fait *hasScanned* à *true* pour l'agent

concerné. Concernant les faits de localisation, la propriété *isInRoom* sera utilisée pour indiquer qu'une entité est dans une pièce, la propriété *isAt* sera utilisée pour indiquer qu'une entité est dans une zone de manipulation. Le type de ces faits pourra être *perceived* ou *dialogue* pour indiquer respectivement que l'information provient de la perception ou du dialogue.

4.5.1 Élaboration du domaine de planification

Afin de générer un plan permettant de résoudre le but utilisateur, nous utilisons ici HATP [Lallement 2014] (pour Hierarchical Agent-based Task Planner), un planificateur basé sur un Réseau Hiérarchique de Tâches (HTN). Ce planificateur nécessite la définition d'un domaine avec les différentes actions et méthodes et leur hiérarchie.

Nous présentons tout d'abord les cinq actions atomiques, correspondant à des actions directement réalisables par le robot.

— *Goto(A, ZM)* : cette action déplace l'agent *A* vers la zone de manipulation *ZM* si celui-ci ne s'y trouve pas encore.
— *Scan(A, ZM, O)* : cette action permet à l'agent *A* de scanner la zone de manipulation *ZM* afin de trouver un objet *O*. Lorsque une zone a été scannée, le fait *hasScanned* passe alors à *true* pour cette zone.
— *Pick(A, O)* : cette action permet à l'agent *A* de prendre l'objet *O*, à conditions que ceux-ci se trouvent dans la même zone de manipulation.
— *Place(A, O, ZM)* : cette action décrit le fait de poser un objet *O* sur une zone de manipulation *ZM*. Pour ce faire, l'objet *O* doit être tenu par l'agent *A* et ce dernier doit se trouver en *ZM*.
— *Handover(A1, O, A2)* : cette action implique que l'agent *A1* ait l'objet *O* et que *A1* et *A2* se trouvent dans la même zone de manipulation. *A1* donne alors l'objet à *A2*.

Cet ensemble d'actions atomiques est complété par des méthodes (tâches composées par ces actions atomiques) afin de définir le domaine de planification. Les premières méthodes permettent d'ajouter le mouvement aux actions de manipulation.

— *Fetch(A, O)* : cette méthode permet d'aller prendre un objet *O*. Elle est composée de l'action *Goto* et de l'action *Pick*.
— *Put(A, O, ZM)* : cette méthode permet d'aller déposer un objet *O* sur une zone de manipulation *ZM*. Elle est composée de l'action *Goto* et de l'action *Put*.
— *Give(A1, O, A2)* : dans cette méthode, l'agent *A1* se déplace vers la zone

de manipulation où se trouve *A2* pour lui donner l'objet *O*. Cette méthode est composée des actions *Goto* et *Handover*.

— *Check(ZM, O)* : cette méthode permet de savoir si un objet *O* se trouve en *ZM*. Pour ce faire, le robot se déplace en *ZM* puis scanne la zone. Cette méthode est composée des actions *Goto* et *Scan*.

Les méthodes de plus haut niveau combinent ces méthodes pour pouvoir accomplir le but utilisateur.

— *Move(O, ZM)* : cette méthode est composée des méthodes *Fetch* et *Put*. Elle permet de déplacer un objet *O* vers la zone de manipulation *ZM*.

— *Bring(O, A)* : cette méthode est composée des méthodes *Fetch* et *Give*. Elle permet d'aller prendre un objet *O* puis de se diriger vers la zone où se trouve *A* afin de lui donner l'objet *O*.

— *LocaliseInRoom(Room, O)* : cette méthode vise à trouver un objet *O* dans une pièce *Room*. Pour ce faire, le robot se déplace vers chaque zone de manipulation qui est à la fois dans *Room* et qui a le fait *hasScanned* à *false* pour le robot. Cette méthode utilise donc la méthode *Check* puis la fonction *LocaliseInRoom* est rappelée de manière récursive afin de continuer la recherche si l'objet n'a pas été trouvé sur le premier meuble. Lorsque tous les meubles ont été scannés, le fait *hasScanned* passe à *true* pour le robot concernant *Room*.

— *Localise(O)* : cette méthode vise à trouver un objet *O*. Contrairement à la méthode *LocaliseInRoom*, le robot n'a pas d'indication sur la pièce où peut se trouver l'objet. Il va donc effectuer la méthode *LocaliseInRoom* pour chaque pièces dont le fait *hasScanned* est faux puis la fonction *Localise* est rappelée de façon récursive.

La hiérarchie de ces différentes méthodes et actions est illustrée par la figure 4.11.

Ce domaine ainsi définit permet de mettre au point les plans nécessaires à la résolution des buts susceptibles d'être demandés par l'utilisateur dans le cadre de notre scénario d'interaction homme-robot.

4.5.2 Supervision et exécution

Parmi les buts possibles provenant du dialogue, certains correspondent directement à une des méthodes haut niveau présentée en section 4.5.1. Ainsi, si l'homme demande au robot de déplacer un objet *O* vers une zone de manipulation *ZM*, si le robot connaît l'emplacement de *O*, il peut directement planifier pour résoudre la méthode *Move(O, ZM)*. Il en va de même pour les autres méthodes de haut

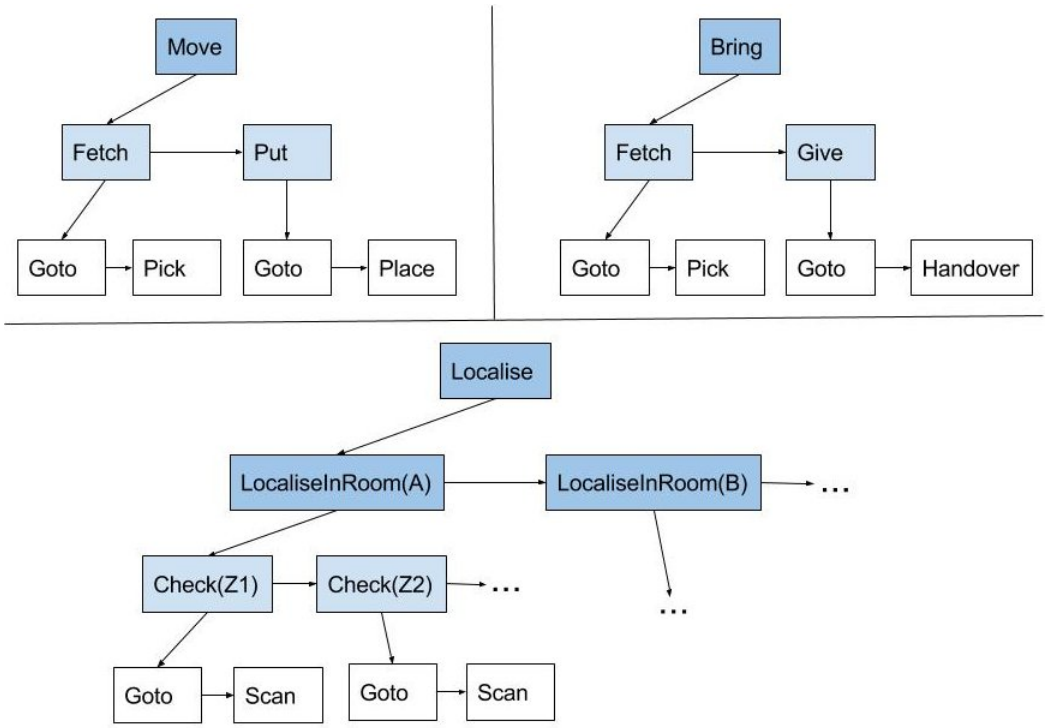


FIGURE 4.11 – Hiérarchie des divers méthodes et actions du domaine de planification définit pour le projet MaRDi.

niveau. Cependant, dans le cas où le robot ignore l'emplacement de l'objet certains raisonnements sont nécessaires pour mener à bien la tâche.

Ainsi, lorsque le robot ne connaît pas l'emplacement de l'objet, une façon simple de résoudre le problème est d'appeler d'abord la méthode *Localise* afin de trouver l'emplacement de l'objet, puis effectuer le déplacement demandé le cas échéant. Cependant, le déplacement du robot et les scans successifs des différentes pièces et zones contenues dans ces pièces est coûteux en temps. Pour réduire au maximum le coût de cette recherche, lorsqu'il doit effectuer un *Localise* le robot va tout d'abord demander si l'humain connaît l'emplacement de l'objet.

Trois cas sont alors possibles.

— **Zone de Manipulation** : l'humain peut indiquer un meuble au robot, dans ce cas la supervision pourra directement faire une requête de planification permettant d'effectuer le déplacement de l'objet initialement demandé.

— **Pièce** : l'humain peut également indiquer une pièce, dans ce cas le superviseur devra faire une requête de planification *LocaliseInRoom* correspondante. Le plan sera achevé dès que l'objet sera trouvé ou que toutes les zones de manipulations auront leur fait *hasScanned* à *true* pour le robot.

— **Emplacement inconnu** : dans le cas où l'homme ignore l'emplacement de

l'objet, le superviseur doit alors faire une requête au planificateur avec la méthode *Localise*. Avant d'exécuter ce plan, le superviseur demande l'autorisation à l'homme de pouvoir chercher dans tout l'appartement.

Nous proposons une règle pour prendre en compte l'état de croyance de l'homme concernant la composition des différentes pièces et zones de manipulation pour améliorer la recherche d'objet. En effet, lorsque l'homme indique une pièce plutôt qu'une zone de manipulation ou lorsqu'il indique qu'il ignore l'emplacement de l'objet, on peut en déduire que l'objet ne se trouve pas dans les pièces ou dans les zones que l'homme a déjà explorées. Lors de l'exécution de la méthode *Localise* le robot va donc explorer en premier lieu les pièces ayant le fait *hasScanned* à *false* pour l'humain et de même pour la méthode *LocaliseInRoom* et les zones de manipulation ayant été explorées par l'homme.

Nous proposons une autre règle pour prendre en compte cette fois-ci l'état de croyance divergente de l'homme et du robot, directement au niveau de la supervision. Dans le cas où l'homme indique une position erronée de l'objet au robot deux situations peuvent apparaître. (1) Le robot ignore l'emplacement de l'objet mais a déjà scanné la position indiquée par l'homme. Dans ce cas le robot informe l'homme et demande confirmation avant de retourner scanner la zone ou la pièce correspondante (il est possible que le robot ait eu une défaillance au niveau de sa perception lors du scan précédent). (2) Si le robot connaît la position de l'objet et si le superviseur est informé de l'état de croyance divergente de l'homme concernant cet objet, le robot exécutera directement la commande en allant chercher l'objet au bon emplacement. Nous considérons ici qu'il n'est pas nécessaire de contredire l'homme car celui-ci mettra à jour la position de l'objet une fois que le robot aura accompli la tâche demandée.

Enfin, nous avons ajouté une dernière règle permettant de gérer les cas d'échec. Imaginons que le robot ait exploré toutes les zones mais qu'il n'ait pas trouvé l'objet demandé. Si le robot a connaissance de l'emplacement d'un objet similaire ayant le même type, c'est à dire qu'il a trouvé un autre livre ou un autre DVD étant classé dans la même catégorie (science-fiction, polar, pour enfant...) le robot pourra indiquer à l'homme qu'il n'a pas trouver l'objet demandé mais lui proposer malgré tout l'objet similaire comme alternative à la commande initiale.

4.5.3 Expérimentation et résultats

4.5.3.1 Configuration expérimentale

Nous avons utilisé la réplique d'appartement du LAAS pour effectuer nos expérimentations. Nous avons simplifié la perception, en utilisant de la capture de

mouvement pour suivre les humains. Pour la détection d'objet nous utilisons un logiciel de reconnaissance de tags nommé Viman (déjà présenté en section 3.5.1).

Pour effectuer les différents mouvements, nous utilisons un planificateur de mouvement[Sisbot 2008] qui commande les actionneurs du robot. Concernant la navigation, nous utilisons *moveBase*[2].

Pour le maintien de l'état du monde perçu par le robot et l'évaluation de la situation nous utilisons l'infrastructure TOASTER et sa capacité de gestion des états mentaux présentée dans le chapitre 3. Dans cette configuration, nous utilisons les modules *PDG* pour l'agrégation des données d'entrées, le module *area_manager* pour gérer les différentes zones et la base de données pour accéder aux faits contenus dans les différents modèles.

Nous configurons l'environnement comme présenté à la figure 4.12. L'environnement contient donc trois pièces qui elles-mêmes contiennent des zones de manipulation.

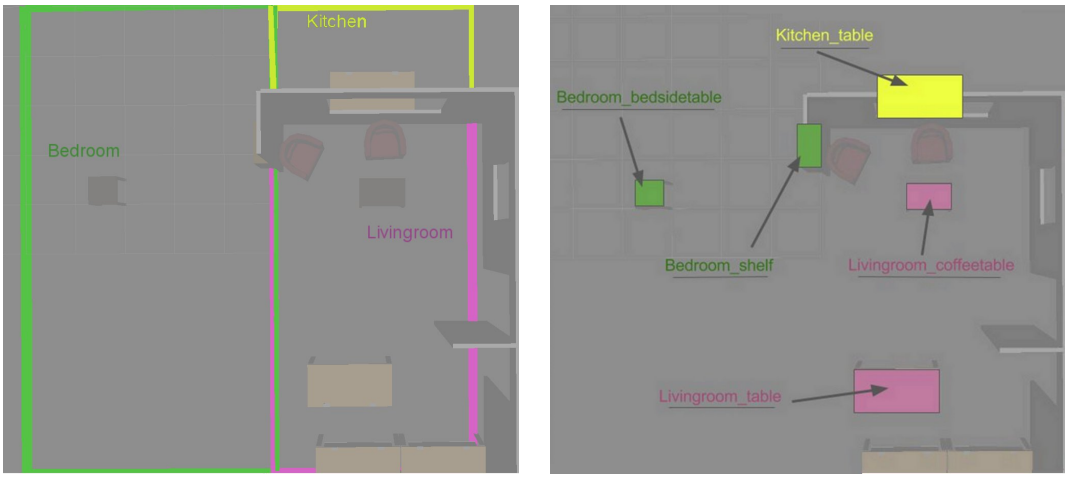


FIGURE 4.12 – Répartition des différentes pièces (image de gauche) et des différentes zones de manipulation correspondants aux meubles contenus dans ces pièces (image de droite).

— **Livingroom** : en rose sur la figure, cette pièce correspond au salon. Elle contient deux zones de manipulation, la table basse (*Livingroom_coffetable*) et une autre table (*Livingroom_table*).
— **Bedroom** : en vert sur la figure, cette pièce correspond à la chambre. Elle contient deux zones de manipulation, la table de nuit (*Bedroom_bedsidetable*) et une étagère (*Bedroom_shelf*).
— **Kitchen** : en jaune sur la figure, cette pièce correspond à la cuisine. Elle

2. http ://wiki.ros.org/move_base

contient une table (*Kitchen_table*).

4.5.3.2 Expérimentation

Plusieurs expérimentations ont été réalisées en utilisant cette configuration du système et avec différentes répartitions d'objets et requêtes utilisateur. Nous reportons ici un scénario présentant plusieurs situations intéressantes.

Au début du scénario, le robot n'a pas encore exploré son environnement et ignore donc la position des objets. Nous démarrons l'expérimentation avec l'homme et le robot se trouvant au niveau de la table du salon *Livingroom_table*. Pour induire une croyance divergente, on informe l'homme que le DVD Bilbo Le Hobit *Bilbo* se trouve sur la table de la cuisine.

Les croyances sur l'état du monde perçu par le robot sont représentées ci-dessous. Tous les faits présents sont vrais. Ils ont été simplifiés pour plus de clarté et le type est indiqué seulement lorsqu'il s'agit d'une information obtenue par le dialogue (les autres ayant été obtenus par perception). Dans le système les faits provenant du dialogue ont un flag pour indiquer la différence avec les autres faits.

```
          Pr2                                    Human
Pr2   isAt       Livingroom_table       Pr2   isAt       Livingroom_table
Human isAt       Livingroom_table       Human isAt       Livingroom_table
Human hasScanned Livingroom_table       Human hasScanned Livingroom_table
                                        Bilbo isInRoom    Kitchen  (dialogue)
```

L'homme demande alors au robot de lui apporter le DVD du Seigneur des Anneaux (*LOTR*). Le robot n'ayant pas de connaissance de la position de cet objet, il demande à l'homme s'il la connaît. L'homme répond alors qu'il se trouve dans la chambre.

Le robot ajoute alors le fait correspondant à son état de croyance et à celui de l'homme :

```
LOTR isInRoom Bedroom  (dialogue)
```

Cette première partie de l'interaction est illustrée par la figure 4.13.

Suite à cela, la supervision va demander la génération d'un premier plan pour obtenir la localisation exacte de l'objet. Pour cela, le planificateur cherche à trouver comment effectuer la tâche *LocaliseInRoom(Bedroom, LOTR)*. Pour exécuter cette première tâche, le robot se rend dans la chambre, au niveau de la table de nuit. Il scanne l'emplacement et y trouve le DVD *Bilbo*.

Les faits suivant sont alors ajoutés à son état de croyance.

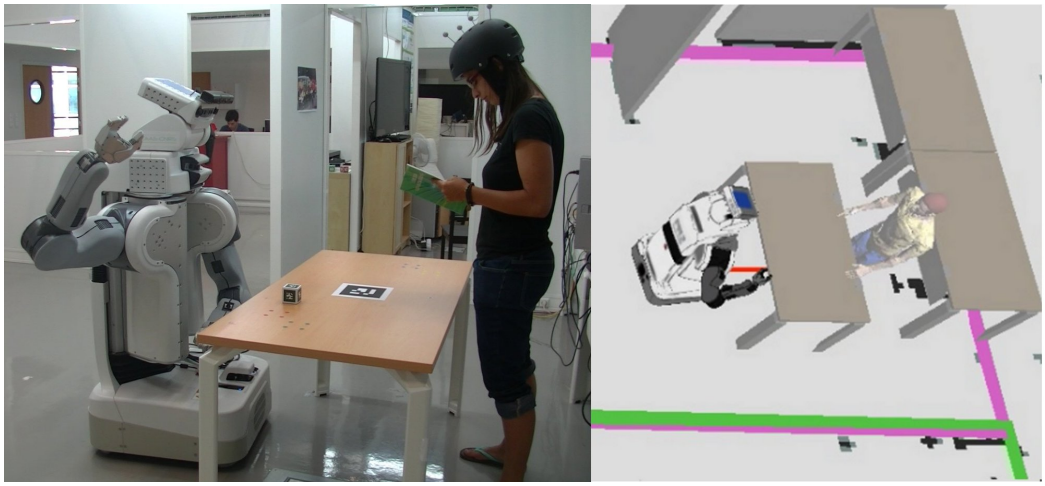


FIGURE 4.13 – L'homme et le robot se trouvent près de la table du salon. À droite la représentation de l'état du monde dans Rviz et à gauche la même scène dans le monde réel.

	Pr2
Bilbo isAt	Bedroom_bedsidetable
Bilbo isInRoom	Bedroom
ROBOT hasScanned	Bedroom_bedsidetable

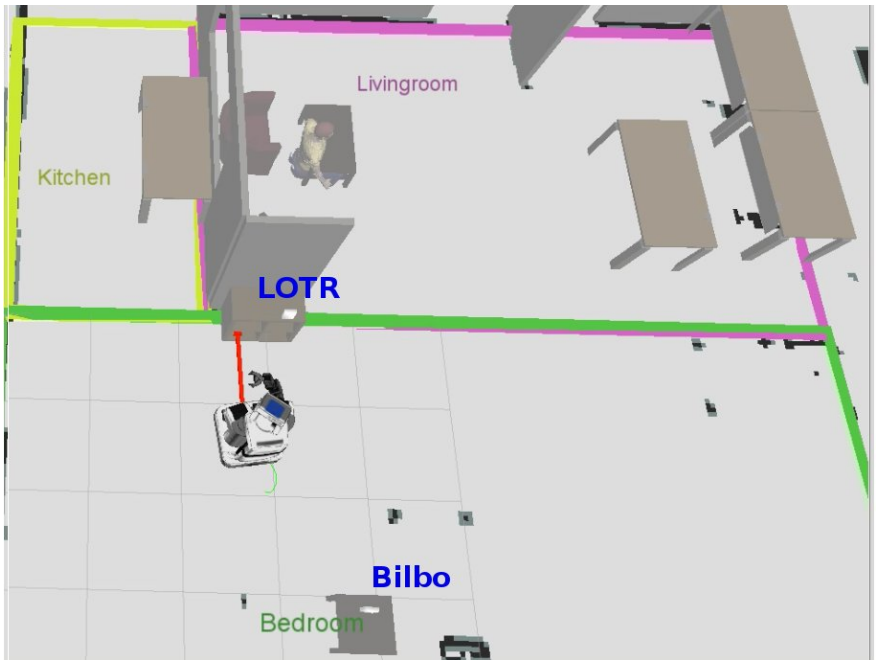
Le robot n'ayant pas trouvé l'objet recherché, le superviseur poursuit l'exécution du plan. Le robot se dirige donc vers l'étagère de la chambre et scanne l'emplacement. Il y découvre le DVD du Seigneur des Anneaux. Le plan de recherche est donc accompli, le robot informe l'homme qu'il a trouvé l'objet en utilisant le *TTS*. Pendant l'accomplissement de ce plan, l'humain s'est rendu à la table basse du salon.

L'évolution de la situation est présentée par la figure 4.14 dans laquelle nous montrons l'état du monde tel qu'il est représenté par TOASTER avant et après l'exploration de la chambre.

Nous présentons les modèles d'état de croyance simplifiés (certains faits isInRoom ont étés enlevés lorsqu'un fait isAt avec le même sujet est présent) :

	Pr2		Human	
Human isAt	Livingroom_coffeetable	Human isAt	Livingroom_coffeetable	
Human hasScanned	Livingroom	Human hasScanned	Livingroom	
Bilbo isAt	Bedroom_bedsidetable	Bilbo isInRoom	Kitchen (dialogue)	
LOTR isAt	Bedroom_shelf	LOTR isAt	Bedroom_shelf (dialogue)	
Pr2 isAt	Bedroom_shelf			

(a) Visualisation dans Rviz de l'état du monde maintenu par TOASTER avant que le robot explore la chambre.



(b) Visualisation dans Rviz de l'état du monde maintenu par TOASTER après que le robot explore la chambre.

FIGURE 4.14 – Représentation de l'évolution de l'état du monde après l'exploration de la chambre.

```
Pr2   hasScanned Bedroom
```

La supervision demande alors au planificateur de produire un plan pour résoudre le but courant, à savoir *Bring(LOTR, Human)*. La supervision va gérer l'exécution pour la génération de mouvement pour attraper l'objet, la navigation vers l'emplacement de l'homme et la génération de mouvement pour transmettre l'objet à l'homme comme illustré par la figure 4.15.

FIGURE 4.15 – Le robot réalise un "Bring" : il va chercher le DVD du Seigneur des Anneaux, le prends puis se déplace vers l'emplacement de l'homme pour lui donner.

Cette première étape permet de montrer comment le robot est capable d'acquérir une information manquante afin d'accomplir une tâche demandée par l'utilisateur. En effet, le robot a demandé à l'homme la position de l'objet, ce qui réduit l'espace de recherche. La réponse de l'homme n'étant pas suffisamment précise pour permettre une exécution directe, la supervision a ordonné au planificateur la production d'un premier plan de recherche d'objet afin d'obtenir l'emplacement précis de l'objet, tout en prenant en compte les indications de l'homme.

Après la réalisation de sa première requête, l'homme fait une autre demande au robot. Il demande cette fois que le robot déplace le livre Cendrillon (*Cinderella*) pour le mettre sur la table basse du salon. Tout comme l'étape précédente, le robot ne connaissant pas l'emplacement de cet objet, il demande à l'homme s'il sait où trouver ce livre. L'homme répond qu'il ignore sa position. Hors, le robot sait que l'homme connaît les éléments qui se trouvent dans le salon (*Human hasScanned Livingroom*), donc si l'homme ignore la position de cet objet c'est qu'il n'est probablement pas dans le salon. De plus, le robot a déjà exploré la chambre (*Pr2 hasScanned Bedroom*). Le superviseur va donc éliminer ces deux zones de sa recherche et directement demander une génération de plan pour résoudre *LocaliseInRoom(Kitchen Cinderella)*. Cette zone n'ayant pas été indiquée par l'homme le robot demande d'abord l'autorisation avant de se rendre à la cuisine. Ceci permet d'éviter que le

robot soit perçu comme intrusif. Le robot se rend donc à la table de la cuisine, scanne l'emplacement et trouve le livre *Peter_Pan* mais ne trouve pas *Cinderella*. Le superviseur n'ayant pas trouvé l'objet et n'ayant plus d'emplacement à scanner, une relâche de contrainte sur l'emplacement est mise en place. Le superviseur exécute alors le plan *LocaliseInRoom(Livingroom Cinderella)*. On considère ici que, si le robot a cherché partout sans trouver l'objet, il doit alors vérifier qu'il ne se trouve pas aux emplacements observés par l'homme, bien que celui-ci ait affirmé ne pas connaître l'emplacement de l'objet. Le robot scanne alors les deux emplacements du salon sans succès. Le robot ayant exploré toutes les zones de l'appartement, il décide alors d'effectuer une relâche de contrainte sur l'objet et propose un objet similaire s'il a connaissance d'un objet similaire présent dans l'environnement. Ici *Peter_Pan* est considéré comme un objet similaire car il est du même type (livre) et du même genre (compte pour enfant). L'humain accepte la proposition, le robot déplace donc *Peter_Pan* de la cuisine à la table basse du salon, cette action est illustrée par la figure 4.16.

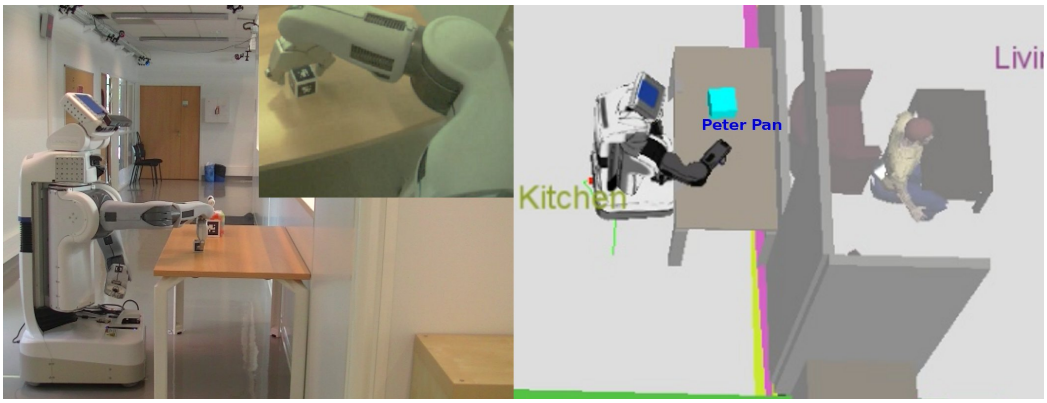


FIGURE 4.16 – Le robot propose de ramener le livre de Peter Pan (ici représenter par un cube pour faciliter la manipulation) car il n'a pas trouver Cendrillon et que ces deux objets sont proches car ils ont le même type (livre) et le même genre (conte pour enfant). L'image de gauche montre le robot attrapant l'objet dans le monde réel et l'image de droite montre la même scène dans Rviz.

Cette deuxième étape illustre la capacité du système à raisonner sur les emplacements possibles en utilisant la prise de perspective perceptuelle pour savoir ce que l'homme a pu observer ou non et pouvoir réduire le champ des possibles concernant l'emplacement de l'objet désiré. Cette expérimentation permet également de montrer comment le robot est capable de relâcher certaines contraintes en cas d'échec de localisation de l'objet désiré, permettant ainsi de proposer une alternative cohérente (proposition d'objet similaire) en cas d'incapacité d'accomplissement du but initial.

Après avoir accompli le déplacement de *Peter_Pan* les états de croyances des agents sont alors :

```
          Pr2                                          Human
Pr2    isAt         Livingroom_coffeetable     Pr2    isAt         Livingroom_coffeetable
Human  isAt         Livingroom_coffeetable     Human  isAt         Livingroom_coffeetable
Human  hasScanned   Livingroom                 Human  hasScanned   Livingroom
Bilbo  isAt         Bedroom_bedsidetable       Bilbo  isInRoom     Kitchen (dialogue)
LOTR   isInHand     Human                      LOTR   isInHand     Human
Peter_Pan isAt      Livingroom_coffeetable     Peter_Pan isAt      Livingroom_coffeetable
Pr2    hasScanned      Bedroom
Pr2    hasScanned      Livingroom
Pr2    hasScanned      Kitchen
```

L'humain demande alors une dernière requête au robot. Il lui demande d'apporter Bilbo qui se trouve dans la cuisine. Le robot étant au courant de la croyance erronée de l'homme, la supervision va directement demander au planificateur de résoudre la tâche *Bring(Bilbo, Human)*. Durant l'exécution, le robot utilise son état de croyance et se dirige donc vers la table de nuit pour prendre Bilbo.

Le robot rapporte Bilbo à l'homme et lui tend l'objet. Le robot est donc parvenu à accomplir la tâche en corrigeant l'énoncé grâce à l'identification de la croyance erronée. De plus, à la fin de l'exécution, comme le robot donne l'objet à l'homme, la croyance erronée a disparu sans que le robot ait eu à en informer l'homme. Cela donne au robot un aspect efficace et discret en évitant d'informer l'homme lorsque ceci n'est pas nécessaire.

Dans le cas où le robot n'aurait pas trouvé Bilbo sur la table de nuit, il aurait alors mis à jour sa croyance et aurait été voir à l'endroit où l'homme pense que Bilbo se trouve au cas où celui-ci aurait été déplacé sans que le robot en soit informé.

4.6 Conclusion

Dans ce chapitre nous avons présenté notre contribution au projet MaRDi qui vise à utiliser les informations contextuelles pour améliorer les performances du dialogue en tenant compte de l'aspect situé de celui-ci durant une interaction homme-robot.

Pour réaliser cela, les données contextuelles générées par notre infrastructure d'évaluation de la situation sont utilisées à différents niveaux de l'interaction et par différents modules impliqués dans l'architecture.

Bien que cela n'ait pas été intégré à la version finale, les données contextuelles de bas niveau (position, configuration), peuvent être utilisées comme données d'entrée.

L'analyse de la gestuelle de l'homme permet une entrée multimodale du système afin de saisir la référence faite à un objet pointé. Ces données contextuelles de bas niveau sont également utilisées au niveau de la fission pour commander au *TTS* de générer une voix adaptée à la situation (voix portée, voix normale) et pour décider d'ajouter ou non de la gestuelle à la parole (si le robot est proche et visible par l'homme).

La représentation symbolique de l'environnement issue de raisonnements géométriques rend également possible l'identification des référents utilisés par l'homme dans sa communication verbale (par exemple, "Le livre qui est sur la table ?" peut être identifié en recherchant tous les livres étant le sujet du fait *? isOn Table*).

Nous avons également montré comment, en partant du principe que l'homme fait des demandes logiques, il est possible d'utiliser les informations disponibles en terme de prise de perspective perceptuelle pour réduire la liste des candidats pour identifier un objet désigné par l'homme. Ainsi, grâce à des règles simples (l'homme ne cherche pas un objet qui lui est visible et ne demande pas de ramener un objet qui lui est atteignable) le mécanisme de fusion est rendu plus efficace.

De même, nous avons montré comment la prise de perspective conceptuelle et notamment l'identification de croyance divergente permet d'améliorer le gestionnaire de dialogue en ajoutant un état et une action possible afin de permettre au système de décider de la politique à adopter en prenant en compte cette croyance divergente lors de l'interprétation de la commande utilisateur. Cette amélioration a pu être testée en utilisant un simulateur robotique afin de faire interagir des utilisateurs avec le système et a confirmé une amélioration tant au niveau de l'efficacité que du taux de réussite concernant la résolution de tâches impliquant une croyance divergente de l'homme.

Enfin, nous avons implémenté un système de supervision basé sur des règles pour permettre au robot d'acquérir une information manquante (la position d'un objet) en utilisant un raisonnement sur sa propre connaissance et sur celle de l'humain. Nous avons également montré comment il est possible au robot de s'adapter en proposant une alternative cohérente dans le cas où la requête utilisateur ne pouvait être satisfaite.

Nous avons donc montré à quel point l'interprétation des données contextuelles, du plus bas niveau (position, distance...) au plus haut niveau (prise de perspective) sont essentielles à tous les niveaux de l'architecture pour permettre un dialogue situé efficace et de qualité.

Cependant, cette architecture et cette implémentation peuvent être améliorées en utilisant par exemple les aspects temporels présents dans la base de données (présentée en section 2.6.2) et également la mémoire des agents (présentée en section

3.4.1). Cela permettrait par exemple de pouvoir répondre à des requête utilisateur du type "Qui a pris le livre qui était sur la table" en interprétant dans la mémoire d'état de croyance de l'interlocuteur afin d'identifier l'objet dont il parle et d'utiliser la table des événements *EVENT_TBL* afin de trouver qui a pris l'objet.

L'interaction considérée prévoit que l'homme est globalement passif durant l'exécution de plan. Un des moyens d'aller plus loin dans cette étude serait de faire collaborer l'homme avec le robot comme pour l'étude qui sera présentée au chapitre 6.

De plus, les études utilisateur effectuées jusque là ont été faites uniquement sur simulateur avec une exécution simulée. Il serait intéressant d'avoir des retours sur la perception du système dans un environnement réel.

Chapitre 5

Prise de Perspective et Reconnaissance d'Intentions

Sommaire

5.1 Contribution

Cette étude a été réalisée en collaboration avec Michelangelo Fiore, doctorant au LAAS-CNRS qui s'est chargé de la modélisation de la reconnaissance d'intention basée sur notre système de modélisation et de gestion des croyances divergentes.

5.2 Contexte

Créer des systèmes capables d'interagir avec l'humain de façon efficace et qui paraissent naturels et intuitifs à l'homme est une des problématiques essentielles

de la robotique d'assistance. Dans leurs activités quotidiennes, les humains collaborent régulièrement sans avoir besoin de requête explicite ou de communication verbale. Une compétence importante pour accomplir cela est la capacité de déduire les croyances et les intentions des autres à partir d'observations et d'indices contextuels. Afin d'être efficace, socialement cohérent et d'aider au mieux l'homme, la connaissance des croyances et la reconnaissance d'intention sont des fonctionnalités importantes à fournir aux robots qui coopèrent ou assistent les humains.

Pour illustrer le problème, nous prenons un scénario. Imaginons une femme, appelée Alice. Alice a fini sa journée de travail et rentre à sa maison. Alice aimerait se détendre en lisant un peu. Elle se met sur le sofa avec un livre et se dirige vers une table à proximité pour prendre ses lunettes. Elle ignore que son mari, pendant la journée, a déplacé ses lunettes dans une autre pièce. Imaginons à présent un robot domestique dans cette situation, qui passe sa journée dans la maison, se déplaçant entre les pièces et essayant d'aider la famille le plus possible. Pour aider les humains, ce robot devrait être doté de plusieurs capacités motrices, perceptuelles, et cognitives. Le robot devrait bien évidemment pouvoir percevoir la présence des humains et des objets de l'environnement, se déplacer librement, et interagir avec les objets.

Pour être un assistant efficace, le robot devrait également être capable de comprendre les problèmes et objectifs des humains. Dans cette situation, il devrait être capable d'analyser qu'Alice, après une journée fatigante, désire se détendre en lisant un livre, qu'elle a besoin pour cela de ses lunettes, et qu'elle ignore la position de celles-ci car son mari les a déplacées. Si le robot avait ces compétences, il pourrait aller chercher les lunettes d'Alice et les lui apporter sans demande spécifique de sa part, faisant de lui un assistant de vie discret, proactif et efficace.

Un des aspects clés dans le scénario présenté est la capacité du robot à savoir qu'Alice ignore que ses lunettes ne sont plus où elle les a laissées. Sans cette connaissance, le robot aurait probablement estimé qu'Alice cherchait à prendre un objet présent sur la table. Il semble que, pour interagir avec les humains, les robots ont besoin d'avoir un ensemble de capacités qui leur permettent de comprendre et modéliser les croyances des utilisateurs, afin d'interpréter leurs actions et d'ainsi permettre la reconnaissance de plan. Une approche pour résoudre ce problème est d'étudier les mécanismes utilisés par les humains lorsqu'ils interagissent entre eux, et essayer de les adapter dans des scénarios de robotique afin de rendre le processus d'interaction homme-robot le plus naturel et intuitif possible pour les humains.

Le premier point qu'il nous faut introduire est le concept d'intention. Il y a beaucoup de définitions différentes de l'intention dans la littérature de psychologie [Bruner 1981] et de philosophie [Bratman 1984]. Ici, nous définissons une intention

comme étant le souhait et la volonté d'atteindre un objectif. L'intention émerge de causes contextuelles (motivations) et reste présente jusqu'à ce que l'objectif soit atteint ou abandonné, poussant l'agent à entreprendre des actions menant à cet objectif.

Comprendre correctement les intentions d'autrui requiert de raisonner sur leurs croyances afin d'interpréter correctement leurs actions. Cette compétence est la théorie de l'esprit et la capacité de prise de perspective associée, présentée au chapitre 3.

Ainsi, Breazeal dans [Breazeal 2009], propose d'utiliser cette prise de perspective pour la reconnaissance de but. Cette capacité de prise de perspective est une prérogative pour la reconnaissance d'intention, car, comme expliqué par [Byom 2013], "en tant qu'humains nous croyons généralement que les autres agissent de façon cohérente avec leurs croyances et leurs objectifs". Il est donc nécessaire que l'interprétation des actions de l'homme soit basée sur leur état de croyance pour déduire correctement leur intention. Ainsi dans [Call J 1998] les sujets sont capables d'interpréter les actions d'autres humains pour distinguer une action "intentionnelle" d'une action "accidentelle".

La reconnaissance des activités humaines est un sujet important dans la recherche en sciences informatiques, et elle peut être étudiée à différents niveaux. L'anticipation des actions humaines et des mouvements permet au robot d'adapter son comportement et d'aider l'humain de façon proactive, comme étudié dans [Koppula 2013]. Dans cette étude la reconnaissance d'actions est basée sur le contexte et sur l'affordance des divers objets présents dans la scène. Une idée intéressante est d'utiliser les modèles internes du robot lui-même pour reconnaître les actions et prédire l'intention de l'utilisateur, comme montré par le système *HAMMER* dans [Demiris 2007].

Les séquences d'actions peuvent être liées à des plans. Ceci constitue une thématique de recherche appelée la reconnaissance de plan. Plusieurs approches ont été étudiées dans ce domaine, en utilisant par exemple la planification classique [Ramırez 2009], probabiliste [Bui 2003] ou des techniques logiques [Singla 2011]. Une infrastructure logicielle intéressante pour la reconnaissance d'intention est la théorie de l'esprit Bayésienne [Baker 2014], utilisée pour représenter les processus de déduction d'un observateur regardant le comportement d'un autre agent, basée sur des POMDPs et des réseaux Bayésiens dynamiques (DBN for Dynamic Bayesian Networks).

Deux approches qui peuvent être utilisées pour l'estimation d'intention sont les processus interactifs de Markov partiellement observés (I-POMDP pour Interactive Partially Observed Markov Decision Processes) et l'apprentissage inverse. Les I-

POMDP [Gmytrasiewicz 2004] offrent un cadre riche qui étend les processus de décision de Markov partiellement observés (POMDP) dans un cadre multi-agent. Les déductions dans ces modèles peuvent être extrêmement complexes, mais il y a eu des tentatives pour résoudre ce problème, comme dans [Doshi 2009, Hoang 2013].

L'apprentissage inverse par renforcement [Ng 2000] formule le problème de calcul d'une fonction de récompense inconnue d'un agent après avoir observé son comportement. Cette stratégie a été appliquée, en utilisant des réseaux bayésiens (BN), dans [Nagai 2015] afin d'apprendre le modèle mental d'un autre agent, et choisir les actions appropriées pour une tâche visant à établir une relation. Une approche liée est la planification inverse, qui a été appliquée dans une infrastructure bayésienne dans [Baker 2009] pour la compréhension de l'action humaine.

Les informations contextuelles peuvent être également utilisées pour mieux résoudre des situations complexes. [Liu 2014] montre un système qui utilise des BNs pour comprendre les intentions des utilisateurs avec une importance tout particulière portée aux données contextuelles.

Les robots assistants ont besoin non seulement de prédire les intentions humaines, mais aussi de produire des plans socialement acceptables afin de les aider à atteindre leur but. Certains systèmes, comme Pike [Karpas 2015] et Chaski [Shah 2011], modélisent explicitement les humains dans leurs plans et permettent au robot d'adapter son comportement aux actions des utilisateurs. D'autres approches, telles que [Levine 2014], intègrent les actions du robot dans le processus de reconnaissance, ce qui permet au système d'adapter avec souplesse son plan pour les humains.

Bien qu'il existe plusieurs travaux intéressants liés au problème de la reconnaissance de l'intention, nous croyons que, pour le moment, peu de systèmes intègrent ces mécanismes dans une architecture complète, capable de représenter et de suivre les modèles mentaux des agents, de reconnaître les intentions, de gérer des plans collaboratifs et d'agir de façon proactive. Dans [Talamadupula 2014], les auteurs utilisent la planification classique, avec une stratégie de replanification efficace, afin d'en déduire les intentions de l'utilisateur. Le système a été mis en œuvre sur un robot PR2 et testé sur un scénario de collaboration. [Breazeal 2009] présente une architecture dans laquelle le robot est capable d'utiliser ses propres schémas et modèles pour déduire les actions et les objectifs humains et d'aider activement à les atteindre. Les plans partagés ne sont pas explicitement représentés dans le système, et le robot aide l'humain en faisant correspondre les informations de but inférées avec ses propres croyances, et en choisissant les actions appropriées.

L'une des contributions principale des travaux présentés ici est l'introduction d'un module de reconnaissance d'intention basé sur le modèle de croyance des

agents maintenu par notre système d'estimation de la situation. Ceci permet au robot d'interpréter correctement les intentions et d'aider de manière proactive et plus pertinente les humains (après avoir observé leurs comportements et en reconnaissant leurs intentions). Nous avons pour cela utilisé notre système d'évaluation de la situation pour fournir les observations des activités humaines nécessaires à la reconnaissance de ces actions et notre système de gestion des croyances. L'algorithme de reconnaissance de l'intention développé par Michelangelo Fiore est basé sur une intégration de BNs et MDPs, qui sont utilisés pour évaluer comment les actions humaines se rapportent à différents objectifs possibles tout en considérant leurs croyances. Nous introduisons le contexte comme information a priori, que nous avons appris de l'homme dans une approche similaire à [Liu 2014]. En utilisant le contexte, nous pourrions représenter, par exemple, que l'homme est plus susceptible de prendre un parapluie au cours d'une journée pluvieuse lorsqu'il sort de chez lui. Nous évaluons ce module dans une étude utilisateur, en comparant les compétences de reconnaissance de l'intention du robot avec celles des humains. Une autre de nos contributions est l'intégration de ce module dans une architecture robotique complète, permettant au robot d'aider de manière proactive les humains après l'estimation de leurs intentions les plus probables.

Notre système est appelé AIR pour Action and Intention Recognition. Il est basé sur plusieurs travaux antérieurs. Dans [Fiore 2014], nous avons présenté un système capable d'exécuter des actions conjointes avec les partenaires humains de manière flexible en adaptant le processus aux préférences de l'utilisateur. L'exécution de la modalité du système peut changer au cours d'un plan, ce qui permet au robot ou à l'humain d'assumer un rôle de leader ou de traiter les participants comme des partenaires égaux. Nous avons également montré dans les chapitres précédents comment nous étions capable de comprendre la situation de l'interaction et notamment de maintenir un modèle d'état mental distinct et cohérent pour chaque agent.

5.3 Estimation de l'intention

Afin de déduire les intentions humaines, nous allons fournir les informations suivantes au système de reconnaissance : une liste de contextes connus, une liste des intentions connues, une liste d'actions connues, un ensemble d'observations de l'action humaine, et un modèle de croyance de l'homme et du robot.

Nous proposons, comme modèle central utilisé pour l'estimation de l'intention, une infrastructure logicielle basée sur des BNs. Un BN est un graphe orienté acyclique avec des variables aléatoires en tant que nœuds. Les connexions entre les nœuds représentent les dépendances conditionnelles entre les variables associées.

Ainsi, on peut associer une fonction de probabilité à chaque nœud, dépendant de ses variables mères, qui produit la distribution de probabilité de la variable correspondante. Lorsque nous acquérons des informations, nous pouvons considérer une partie des nœuds comme *preuves*, en fixant leur valeur, ce qui permet de produire une meilleure estimation des probabilités des autres nœuds. Nous appelons notre implémentation de BN un Graphe d'Intention (IG pour Intention Graph).

Un IG est composé des couches de nœuds suivants :

— Nœuds de Contexte : ces nœuds représentent les données contextuelles, modélisées par des booléens (e.g. HotDay, ColdDay).

— Nœuds d'Intention : ces nœuds booléens représentent l'ensemble des intentions possibles. Chaque intention peut avoir une dépendance conditionnelle à plusieurs contextes.

— Nœuds d'Action. c'est l'ensemble des actions humaines qui sont actuellement surveillées. Chacun de ces nœuds a une dépendance conditionnelle à l'ensemble des nœuds d'intention.

— Nœuds d'observation. Nous associons à chaque action un ensemble de nœuds d'observation particulier, qui ont une dépendance conditionnelle au nœud d'action en question.

Dans une utilisation classique, le robot va créer, pour chaque humain "surveillé" (monitored human), un IG, formé par les nœuds de contexte et d'intention, que nous considérons statiquement connus par le robot, et une liste variable de nœuds d'action et d'observation, qui dépend du modèle de croyance de l'homme. Le système de reconnaissance d'action va créer un nœud d'action pour chaque action connue dont les *preconditions* sont satisfaites dans le modèle de croyance de l'humain, et leurs nœuds d'observation associées. Le modèle de croyance de l'humain pouvant être différent de celui du robot, certaines *preconditions* d'actions peuvent être valides chez l'homme et invalides chez le robot et inversement, ce qui explique que certaines actions soient envisageables et donc présentes dans l'IG d'un agent et pas chez un autre et donc que les IGs de chaque agent peuvent diverger. L'IG de chaque agent devra être mis à jour chaque fois qu'un agent exécute une action A. Pour se faire, le système change dans chaque IG, les nœuds d'action et d'observation de la version précédent A pour les remplacer par de nouveaux, qui correspondent à l'état du monde après que A a été effectuée. En effet, chaque action modifiant l'État du monde, elle modifie potentiellement les actions possibles en validant ou invalidant certaines préconditions.

Lors de la "surveillance" d'un humain par le système, nous utilisons les nœuds de contexte et d'observation comme *preuves*, ce qui revient à les considérer comme observables par le robot. Ces informations nous permettent d'avoir une bonne es-

timation des actions et des intentions les plus probables pour l'homme, comme expliqué dans la section 5.3.4.

Un exemple d'IG, pris d'une expérience, peut être vu dans la figure 5.1. Dans les paragraphes qui suivent, nous allons expliquer le rôle de ces couches de nœuds, et comment les dépendances conditionnelles les reliant sont calculées.

5.3.1 Du contexte à l'intention

Nous introduisons un ensemble de contextes dans notre domaine. Nous considérons comme contexte n'importe quelle information qui peut être utilisée pour caractériser et motiver une intention [Abowd 1999]. Nous modélisons un contexte sous la forme d'une propriété, qui peut prendre différentes valeurs et influencer la probabilité d'un utilisateur d'avoir une intention donnée. Par exemple, on établit qu'un humain a plus de chances de vouloir cuisiner à l'heure du dîner, ou de boire une boisson chaude lorsqu'il fait froid.

Les nœuds de contexte peuvent directement influencer un ou plusieurs nœuds d'intention. Dans notre approche, nous avons choisi d'apprendre ces dépendances conditionnelles directement de l'homme, comme expliqué dans la partie 5.6.2.

5.3.2 De l'intention à l'action

Pour comprendre comment les actions sont liées aux intentions, il faut pouvoir répondre à la question : quelles actions ferait un humain, dans cette situation, étant donné son état de croyance sur l'état du monde, pour accomplir l'objectif lié à son intention ? Notre démarche est basée sur le principe de rationalité [Dennett 1989], qui affirme que les agents ont tendance à, étant donné leur état de croyance, choisir l'action qui leur semble la plus efficace, afin d'atteindre leur but.

Dans [Blakemore 2001], les auteurs expliquent que "l'attribution des intentions aux actions pourrait reposer sur l'identification de l'action observée et sa correspondance dans notre propre représentation de l'intention". Nous suivons cette idée en donnant au robot un ensemble de modèles de planification. Chacun de ces modèles de planification est relié à une intention, et représente l'ensemble des plans associés permettant d'atteindre le but de l'intention. Grâce à cela, nous pouvons quantifier la correspondance de l'action humaine avec un plan donné, qui est lui même associé à une intention. En d'autres termes, cela permet de répondre à la question "Est-ce que l'humain pense que l'action qu'il vient de faire lui permet d'atteindre tel ou tel but lié à telle ou telle intention".

Dans notre implémentation, pour chaque intention connue par le système, nous créons un Processus de Décision Markovien (MDP pour Markov Decision Process) associé, pour représenter tous les plans possibles liés à cette intention. Un MDP

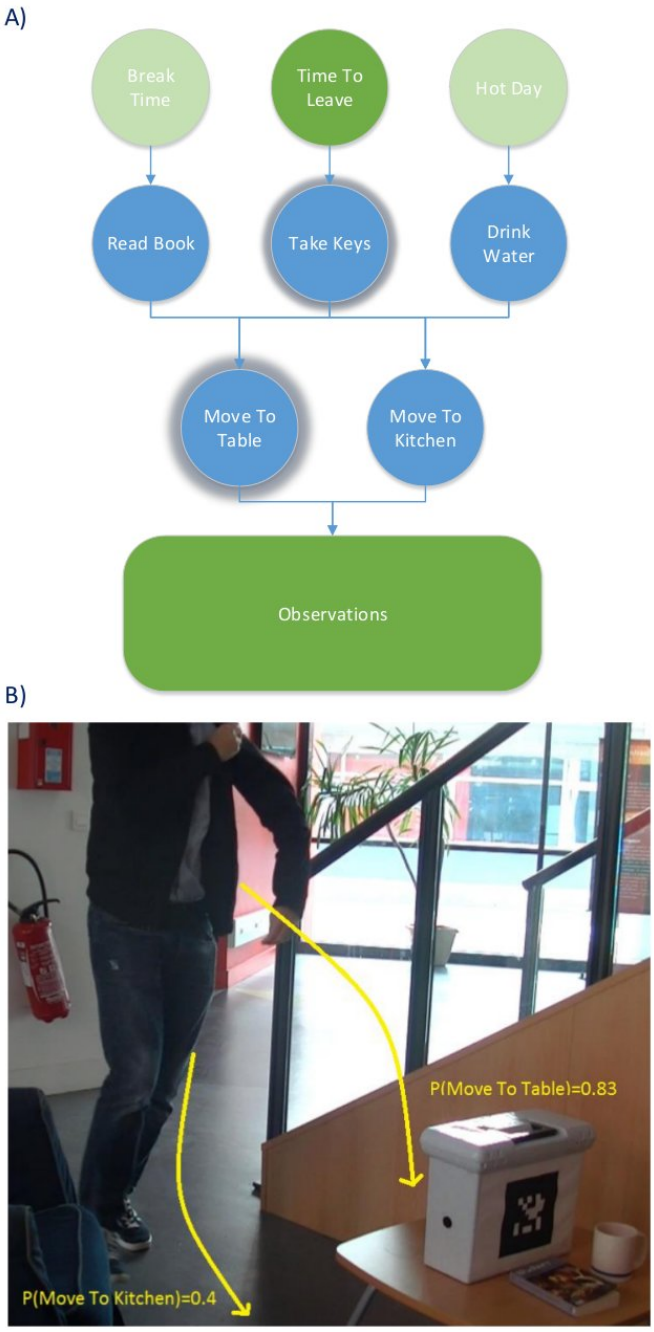


FIGURE 5.1 – Une scène d'expérimentation. Les flèches jaunes montrent les actions possibles et leurs probabilités associées. Le diagramme représente l'IG (Intention Graph) associé. Les cercles verts représentent les nœuds considérés comme preuves et les bleus sont les autres nœuds. Pour les nœuds de contexte, situés en haut du graphe, les nœuds ayant une valeur fausse sont grisés. Parmi les autres nœuds, ceux ayant la probabilité la plus forte sont entourés. Les nœuds d'observation ont étés regroupés en un seul bloc pour simplifier le diagramme.

modélise le processus de décision pour un agent dans des situations où le résultat d'une action est partiellement aléatoire, et peut amener à plusieurs résultats. De façon formelle, un MDP est un tuple (Q, A, T, R, γ), où Q est l'état du système, A est un ensemble d'actions, $T(s, a, s')$ est la probabilité que réaliser l'action a dans l'état s mènera à un état s', $R(s, a)$ est la récompense que l'agent obtiendra après avoir réalisé l'action a dans l'état s et γ est un facteur d'actualisation.

Le souci principal des MDPs est de déterminer la fonction $\pi(s)$ définissant la politique optimale qui associe la meilleure action à chaque état. "Meilleur" signifie ici que nous cherchons l'action qui maximise le gain de la récompense sur un horizon potentiellement infini. Il y a plusieurs algorithmes bien connus pour calculer la politique d'un MDP (voir [Mausam 2012]). En utilisant ces algorithmes nous pouvons également calculer, pendant l'apprentissage, la fonction de valeur d'action $Q(s, a)$ qui associe à un état s et une action a la récompense attendue sur un horizon actualisé infini. Cette fonction est un point essentiel dans notre approche du problème d'estimation de l'intention.

Nous allons à présent expliquer comment nous utilisons la fonction de valeur d'action pour créer des dépendances conditionnelles entre nœuds d'intention et nœuds d'action dans l'IG. Commençons par définir $P(a|I_i = 1)$, la probabilité que l'action a soit réalisée si l'intention I_i est vraie. Nous modélisons cette probabilité comme $P(a|I_i = 1) = \frac{Q_i(s,a)}{\sum_b (Q_i(s,b))}$, où nous normalisons la fonction de valeur $Q_i(s, a)$ pour l'intention i et l'action a dans l'état de croyance de l'homme s, divisé par la fonction de valeur $Q_i(s, b)$ calculée sur toutes les actions surveillées b. Nous pouvons étendre ce calcul au cas où un nombre générique d'intentions sont vraies pour calculer la probabilité des nœuds d'action : $P(a|I_1, I_2, ..., I_m) = \frac{\sum_{i:I_i=1} Q_i(s,a)}{\sum_b \sum_{i:I_i=1} Q_i(s,b)}$.

L'idée principale dans ce problème est d'utiliser les croyances de l'humain comme entrées des fonctions de valeurs des MDPs. De cette façon, nous utilisons la prise de perspective au niveau de la planification. Cela traduit le fait que l'humain a des actions cohérentes avec son intention dans son propre état de croyance. Ainsi, même dans la situation où l'homme a des actions qui semblent incohérentes par rapport à l'état du monde courant (par exemple dans le cas où il a une croyance erronée), l'utilisation de la prise de perspective permet malgré tout de les interpréter et de les relier à une intention.

5.3.3 De l'action aux observations

Les intentions sont déduites des actions humaines, donc le robot a besoin de surveiller leur exécution. Dans notre étude, les actions possibles de l'homme sont des actions de manipulation d'objets. Il nous faut donc pouvoir observer les mouvements

de l'homme par rapport aux objets avec lesquels il peut interagir. Pour chaque nœud d'action nous définissons un ensemble de quatre nœuds d'observation : la distance du corps de l'humain à la *cible* de l'action, la variation de celle-ci, la distance de la main de l'humain à la *cible* de l'action, et la variation de celle-ci. Les dépendances conditionnelles des nœuds d'observation sont pré-calculées.

5.3.4 Déduction d'intention et d'action

Nous supposons ici, qu'à chaque moment, un humain ne peut exécuter qu'une action à la fois et que le robot réagira seulement à l'intention la plus probable. L'action et l'intention la plus probable sont déduites du BN de la façon suivante.

— $P(n)$ est la probabilité inférée d'un nœud n.
— $B(n)$ est l'ensemble des frères de n (c'est à dire les nœuds de la même couche).
— δ_1, δ_2 sont deux seuils.

Le robot déduit qu'une action a été réalisée, ou qu'un humain a une intention en suivant ces règles :

1. $P(n_i) > \delta_1$
2. $\forall b \in B(n_i) : P(n_i) > P(b) + \delta_2$

n_i étant le nœud associé à l'intention ou l'action concernée. La première condition permet de s'assurer que l'action ou l'intention est suffisamment probante et la seconde permet quant à elle de vérifier qu'elle se démarque des autres actions ou intentions potentielles.

Pour illustrer ces règles, on reprend l'exemple présenté en 5.1. Deux actions sont alors possibles : $MoveToTable$ et $MoveToKitchen$. Imaginons dans un premier temps qu'on ait :

— $P(MoveToKitchen) = 0.4$
— $P(MoveToTable) = 0.83$
— $\delta_1 = 0.7$
— $\delta_2 = 0.5$

Dans cette situation, la première règle est valide pour l'action $MoveToTable$ (car $P(MoveToTable) > \delta_1$) et est invalide pour $MoveToKitchen$ (car $P(MoveToTable) < \delta_1$). Cela signifie que l'action $MoveToTable$ est suffisamment probable pour être considérée comme acceptable et non $MoveToKitchen$. Cependant, on a $P(MoveToTable) < P(MoveToKitchen)+\delta_2$ La seconde condition n'est donc pas respectée. Cela signifie que, étant donné les seuils choisis, l'action de se diriger vers la table ne se démarque pas suffisamment, en terme de probabilité, des autres actions possibles (ici en l'occurrence de l'action $MoveToKitchen$). Ces deux règles permettent de réduire les faux positifs lorsqu'une situation est ambiguë.

Lorsque le robot déduit qu'une action a été réalisée, il met à jour l'état du monde avec ses *postconditions*, déclenchant une mise à jour des croyances de tous les agents présents. Lorsque le robot déduit l'intention actuelle de l'homme, il choisit une réaction possible, comme expliqué dans la section 5.4.

5.4 Comportements pro-actifs

Une fois que le robot est parvenu à estimer l'intention la plus probable, il peut essayer d'aider l'homme à atteindre son but. Nous définissons deux types d'attitudes proactives que le robot peut exécuter à tout moment : corriger l'état de croyance de l'homme et accomplir une partie du plan pour atteindre le but.

5.4.0.1 Correction de l'état de croyance

Avoir une croyance erronée ou incomplète sur l'environnement peut amener l'agent à exécuter des actions non optimales, inutiles, voir contre-productives ou dangereuses. Le robot a besoin de détecter ces situations afin de prévenir l'homme pour que celui-ci puisse corriger son état de croyance et accomplir les actions appropriées. Le robot suppose ici qu'il a toujours un état de croyance correct. Notre solution utilise la récompense attendue de l'action, introduite dans la partie 5.3.2. Le principe est de comparer la récompense attendue lors de l'accomplissement d'une action, pour l'humain d'une part (basé sur son modèle de croyance) et pour le robot d'autre part. En d'autres termes, nous regardons si l'homme a une croyance erronée (différente du robot) sur l'influence que son action aura sur l'accomplissement du but. Pour formaliser : nous comparons la valeur d'action $Q_m(s_h, a_h)$ et $Q_m(s_r, a_h)$, où m est l'intention la plus probable, s_h et s_r sont les états de croyance de l'homme et du robot, et a_h est l'action la plus probable. Si ces valeurs ne sont pas égales, cela signifie que l'homme attend un résultat de son action qui est différent de ce qu'il va réellement faire.

Nous proposons une solution simple, où le robot prévient l'homme de la croyance divergente détectée pour cette action. Par exemple, Bob veut boire du thé contenu dans une bouteille opaque. Cependant, le robot sait que cette bouteille est vide (par exemple parce qu'un agent a bu le dernier verre), mais Bob l'ignore. Lorsqu'il s'approche de la bouteille, le robot détecte que l'intention la plus probable (en utilisant le processus décrit en 5.3) est de boire du thé. Le système calcule la récompense attendue par l'action de prendre la bouteille dans le modèle de Bob et du robot afin de modéliser l'avancement du plan attendu par cette action par rapport à l'état de croyance de chacun. On obtient des valeurs différentes. Ici la valeur est plus élevée

pour Bob signifiant qu'il pense que son action aura une conséquence "meilleure" (qui fera avancer l'état du monde vers son but) que celle prévue par le robot (la récompense dans l'état mental du robot étant inférieure). Le système vérifie les valeurs des propriétés associées à la bouteille dans les deux modèles mentaux et en extrait celles qui sont divergentes. En utilisant cette information, le robot corrige la croyance divergente en informant l'humain que la bouteille ne contient plus de thé.

5.4.0.2 Réalisation d'une partie du plan

Il y a des situations dans lesquelles le robot devrait aider l'homme à accomplir son but en agissant. Dans ces cas là, le robot crée un plan partagé pour accomplir le but lié à l'intention reconnue, et exécute les actions qui lui incombent. Le robot est capable de surveiller les actions de l'homme et replanifier si ses actions diffèrent du plan qu'il a établi, permettant ainsi d'être flexible aux choix de l'homme et de s'adapter dynamiquement de la même façon que décrite en [Fiore 2014].

5.5 Architecture

Dans section nous présentons l'architecture permettant de mettre en place les processus détaillés ci-dessus. Pour détailler les différents modules nous nous appuierons sur la figure 5.2

Dans un premier temps, on peut retrouver les différents modules présentés dans les chapitres 2 et 3. Ces modules sont chargés de collecter les données auprès des capteurs et de produire des faits au *Belief Manager* qui se charge de maintenir un modèle d'état du monde pour chaque agent.

Le module appelé *Intention Graph Manager* est chargé de créer l'IG de chaque humain. Pour ce faire, il utilise les données de contexte et les intentions du domaine d'application considérées comme connues par le robot. Puis le module va récupérer, auprès du *Belief Manager*, l'état du monde tel qu'il est perçu par l'humain concerné par l'IG en construction. À partir de cet état du monde, le module peut identifier quelles actions sont réalisables et va ajouter ces actions et les observations associées à l'IG, comme expliqué en section 5.3. Enfin, pour créer les dépendances conditionnelles entre les nœuds d'intention et les nœuds d'action, le module de gestion des IGs fait une demande au module de gestion des MDPs pour connaître la probabilité $P(a|I = 1)$ pour chaque action a et intention I contenues dans l'IG, comme expliqué dans la section 5.3.2.

Par la suite, pour obtenir et mettre à jour les valeurs des nœuds d'observation, le module qui gère les IGs va utiliser les faits exportés par le module *Agent Monitor*.

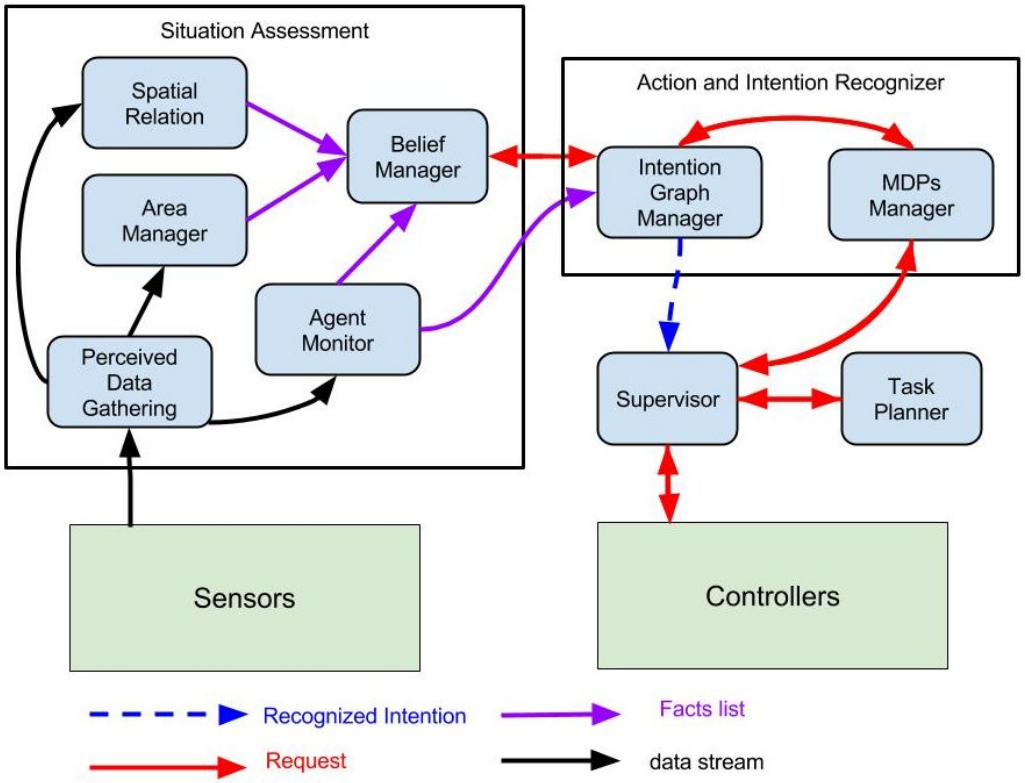


FIGURE 5.2 – Schéma d'architecture des différents modules impliqués dans la reconnaissance et l'utilisation de l'intention.

Enfin, le module décide de valider une intention lorsque les règles présentées en section 5.3.4 sont valides. À ce moment, le module de gestion des IGs envoie l'intention reconnue au superviseur.

Le superviseur vérifie alors que l'action courante de l'humain est "valide" par rapport à l'état du monde réel (comme expliqué en section 5.4.0.1). Le superviseur informe l'humain si nécessaire de sa croyance erronée, puis utilise le module de planification de tâches (Task Planner) pour produire un plan permettant d'accomplir le but lié à l'intention de l'humain. Si dans ce plan le robot peut accomplir certaines actions pour aider l'homme à réaliser son but, le superviseur va alors piloter les différents modules permettant l'exécution de ces actions.

5.6 Étude expérimentale

Pour estimer les performances de notre système en terme de reconnaissance d'intention, nous voulons mener une étude expérimentale. Le problème au niveau de l'évaluation de notre système est que les intentions humaines ne sont pas directe-

ment observables. Une des façons de procéder est présentée dans [Baker 2014]. Nous allons comparer l'estimation de l'intention humaine faite par d'autres humains avec la prédiction de notre système. Afin de pouvoir réaliser cette comparaison nous avons effectué une étude utilisateur où nous avons montré aux participants plusieurs vidéos, leur demandant d'estimer la probabilité d'un ensemble d'intentions pour chaque vidéo, puis nous avons regroupé ces résultats.

Nous avons réalisé un test d'équivalence TOST (Two One-Sided Test) où nous comparons les intentions telles qu'elles ont été estimées par les humains avec les estimations du système AIR. Nous choisissons comme seuil pour l'équivalence la déviation standard σ des réponses des utilisateurs. L'idée derrière ce choix est que, si la réponse du système est plus proche que la déviation standard de la réponse moyenne des humains, ses prédictions sont comparables à la réponse d'un utilisateur moyen présent dans notre groupe d'utilisateurs.

Nous définissons nos hypothèses :

1. $H_0 : \mu_{hi} - \mu_{ri} \leq -\sigma_{hi}$ OU $\mu_{hi} - \mu_{ri} \geq \sigma_{hi}$
2. $H_A : -\sigma_{hi} < \mu_{hi} - \mu_{ri} < \sigma_{hi}$

Où μ_{hi} et μ_{ri} sont la moyenne humaine et la réponse de notre système pour le test i, σ_{hi} est la variance des réponses humaines pour le test i.

Nous avons effectué des tests pour évaluer : a) la prédiction sans indices (No clues), b) la prédiction en présence d'indices contextuels (Contextual clues), c) la prédiction en présence d'indices sur l'état mental (Divergent belief).

Nous établissons un environnement avec un ensemble de meubles : une *Étagère de Cuisine*, une *Table*, un *Canapé*, et une *Chaise*. Dans cet environnement, nous avons créé deux scénarios, dans lesquels se déroulent plusieurs tests, avec deux humains, *Max* et *Bob*, qui réalisent différentes actions. chaque scénario contient une collection d'objets, et un ensemble d'intentions préétablies. Pour les tests relatifs aux états de croyances, nous commençons par montrer aux utilisateurs et au système une séquence d'événements, leur permettant de construire un modèle d'état mental pour les agents. Nous allons détailler les deux scénarios et les tests s'y rapportant.

5.6.0.1 Scénario des cookies

— Objets : une *Boîte de cookie*, un *Mug*, et une *Bouteille d'eau* sont placés sur une *Table*, proches les uns des autres comme exposé dans l'image 5.3. Un paquet de *Cookies* est placé dans l'*Étagère de cuisine*. La *Boîte de cookie* peut contenir ou non des *Cookies*.

— Intentions : manger un cookie (*Eating a cookie*), boire de l'eau (*Drinking water*), et lire un livre (*Reading the book*).

— Tests :
  — *No clues* : *Max* s'approche de la *Table*.
  — *Contextual clues* : *Max* s'approche de la *Table* en faisant un commentaire sur la chaleur ambiante.
  — *Divergent belief Max* : *Max* s'approche de la *Table*.
  — *Divergent belief Bob* : *Bob* s'approche de la *Table*.
— *Événement de croyance divergente* : *Max* et *Bob* discutent sur le *Canapé*. Max mange le dernier *Cookie* de la *Boîte de Cookie* avant de la refermer et de sortir. Pendant que *Max* est parti, *Bob* prend des *Cookies* de l'*Étagère de la cuisine*, remplit la *Boîte de cookies*, et la referme, avant de partir.

FIGURE 5.3 – Image représentant la configuration du scénario des cookies issu des vidéos présentées aux humains chargés d'estimer les intentions.

L'*Événement de croyance divergente* a été montré aux utilisateurs et au système entre le test *Contextual clues* et le test *Divergent belief Max*.

Nous avons volontairement inclus une intention, *Reading the book*, sans mettre de livre dans l'environnement visible, afin d'introduire un élément incertain dans le scénario.

5.6.0.2 Scénario des clés

— Objets : Une *Boîte* est placée sur une *Table*. La *Boîte* cache partiellement la vue des personnes qui approchent. Un *Livre* et un *Mug* sont placés derrière la *Boîte*, afin qu'ils puissent être vus depuis le *Canapé* mais pas des gens qui

s'approchent. Pour illustrer la configuration, une image extraite de la vidéo est présentée à la figure 5.4.

— Intentions : prendre le *Mug Taking the mug*, prendre les *clés Taking the keys*, lire un *Livre Reading the book*.

FIGURE 5.4 – Image représentant la configuration du scénario des clés issue des vidéos présentées aux utilisateurs chargés d'estimer les intentions.

— Tests et événements :
  — *No clues* : *Max* s'approche de la *Table*.
  — *Contextual clues* : *Max* s'approche de la *Table* en hâte, tout en enfilant un manteau.
  — *Divergent belief Max* : *Max* s'approche de la *Table* en hâte, tout en enfilant un manteau.
— *Événements de croyance divergente* : *Max* est assis sur le *Canapé*, buvant dans le *Mug*, et ayant les *Clés* en mains. Son téléphone sonne, il pose les *Clés* et le *Mug* sur la *Table*, derrière la *Boîte*, et quitte la salle. Pendant que *Max* est absent, *Bob* arrive et prend place sur le *Canapé*, lisant un *Livre*. Lorsqu'il voit les *Clés*, il pose le *Livre* sur la *Table* et prend les *Clés* pour les amener aux objets trouvés.

L'*Événement de croyance divergente* est montré aux utilisateurs et au système AIR entre les événements de *Contextual clues* et *Divegent belief Max*.

5.6.1 Étude utilisateurs

Pour collecter les estimations d'intentions de la part d'humains, nous avons créé une étude en ligne, où nous présentons des vidéos en relation avec les tests et événements des deux scénarios. Les utilisateurs ont estimé la probabilité de chaque

intention disponible sur une échelle de Likert à 5 niveaux. L'étude a été réalisée en trois langues, avec des utilisateurs résidents dans différents pays [1]. Nous avons recueilli les réponses de 78 adultes, calculé la moyenne de chaque réponse puis nous avons converti ces moyennes en pourcentages, afin de les comparer aux réponses de notre système.

Lorsqu'on observe les réponses des utilisateurs (voir figure 5.5), nous observons que, dans l'absence d'indices, les individus ont donné la même note aux différentes intentions liées aux objets visibles. Les indices de contexte ont eu la plus grosse influence sur les réponses des utilisateurs. Cela est particulièrement visible dans le test *Contextual Clues* du scénario des clés (*Keys Scenario*), où les utilisateurs ont noté comme intention la plus probable *Take Keys*, même si aucune clé n'était visible dans la vidéo. Les croyances divergentes ont également influencé l'estimation des humains, mais dans une proportion moins importante. Globalement, les réponses les plus fortes, ont été données par le test *Divergent Belief Max* sur le scénario des clés, qui utilise à la fois des indices de contexte et la connaissance de l'état mental divergent.

5.6.2 Implémentation

Nous avons répliqué les deux scénarios présentés précédemment. Afin d'évaluer notre système dans des conditions optimales, nous avons simulé les observations faites par le robot en fournissant directement au système les données correspondantes. [2].

Au début d'un scénario, le robot construit un modèle de l'état du monde. Nous considérons au démarrage d'un test que la boîte de cookie est pleine et ses valeurs sont mises à jour en utilisant les *postconditions* (effets) des actions humaines simulées. Nous considérons que la boîte est vide lorsqu'un humain prend un cookie, et comme pleine lorsqu'un humain y met un cookie.

Nous avons construit différents IGs pour les scénarios. Chaque test a un graphe différent, lié à l'agent principal dont on veut reconnaître l'intention. Nous considérons trois différents nœuds de contexte pour ces IGs : *HotDay*, qui est vrai lorsque la journée est particulièrement chaude ; BreakTime, qui est vrai lorsque les agents prennent une pause ; TimeToLeave, qui est vrai lorsqu'il est tard dans la journée, et que les humains partent en général de leur lieu de travail pour rentrer chez eux.

Comme indiqué précédemment, nous avons choisi de suivre [Liu 2014] afin d'apprendre le lien entre contexte et intention. Nous avons fait une petite étude utilisa-

1. Une version de cette étude est fournit à l'adresse http ://goo.gl/forms/gWwdIutbOCUk3vdx2
2. Des vidéos d'illustration des expériences associées peuvent être vue sur http ://homepages.laas.fr/mfiore/roman2016.html

teur avec 15 personnes, dans laquelle 5 scénarios ont été présentés. Chaque scénario est lié à une des intentions utilisées dans nos tests. Pour chaque scénario, nous avons demandé aux utilisateurs de noter le lien perçu entre l'intention et les trois contextes. Pour ce faire, les individus ont noté la "force" de ce lien sur une échelle de Likert à cinq niveaux. Nous avons utilisé la moyenne des réponses pour calculer la probabilité de la dépendance conditionnelle entre les nœuds de contexte et les nœuds d'intention.

Dans le scénario du cookie (*Cookie Scenario*) le graphe pour les tests est construit à partir des nœuds suivants :

— Nœuds de contexte : *Hot Day*, *Break Time*, *Time to Leave*
— Nœuds d'intention : *Fill Cookie Box*, *Eat Cookie*, *Drink Water*, *Read Book*.
— Nœuds d'action : *Move to Table*, *Move to Kitchen*.
— Nœuds d'observation : distance du corps de l'agent et de sa main à chaque cible d'action.

Nous introduisons l'intention de remplir la boîte de cookie (*Fill Cookie Box*), qui n'est pas présente dans les tests soumis aux humains, afin que le système détecte lorsque Bob remplit la boîte de cookies pendant l'événement de croyance divergente.

Notre système, dans cette expérimentation, n'est pas équipé de capacité pour comprendre la parole de l'homme, et assigne directement les nœuds de contexte à des valeurs plausibles et qui pourraient être acquises en regardant les vidéos. Pour le test de *Contextual Clues*, nous assignons la valeur de *Hot Day* à vrai (dans la vidéo Max commente la chaleur du jour), et *Break Time*, et *Time to Leave* à faux (car aucun élément dans la vidéo n'indique que ces contextes sont vrais).

L'événement de croyance divergente (*Divergent Belief Event*), le test *Divergent Belief Max*, et le test *Divergent Belief Bob* ont été présentés dans cet ordre à notre système, qui a pu ainsi suivre et mettre à jour les états de croyance des agents et créer les nouveaux IGs qui conviennent à cette situation. Pendant l'événement de croyance divergente, plusieurs IGs ont étés créés avec différents nœuds d'action et d'observation, pour suivre la séquence d'action réalisée par les deux agents. Par exemple, quand *Max* quitte la pièce, *Bob* a la possibilité d'exécuter les actions *Take Mug*, *Take Water Bottle*, *Open Cookie Box*, *Move to Kitchen Shelf* ou *Leave Room*. Les nœuds d'intention et de contexte restent quand à eux inchangés dans tous les IGs du scénario.

Le scénario des clés (*Keys Scenario*) a un IG, avec les différences suivantes :

— Nœuds de contexte : *Hot Day*, *Break Time* et *Time to Leave*.
— Nœuds d'intention : *Drink Water*, *Take Keys*, *Read Book*.

Les nœuds d'action et d'observation sont les mêmes que le scénario précédent et suivent les mêmes idées durant l'événement de croyance divergente. Un exemple

d'IG utilisé dans les tests est présenté à la figure 5.1. Pour les tests *Contextual Clues* et *Divergent Belief*, nous mettons la valeur contextuelle *Time to Leave* à vrai (Max met un manteau et semble pressé), et les autres valeurs contextuelles à faux. En utilisant notre composant et ces IGs, le système a été capable d'obtenir des prédictions à partir des actions des utilisateurs.

5.6.3 Discussion

Nous réalisons des tests TOST pour chaque intention contenu dans les scénarios, en comparant les réponses des humains avec celles du robot pour un total de 21 tests.

Nous calculons les p-valeurs et réalisons nos tests en utilisant la valeur de signification $\alpha = 0.05$.

En analysant les résultats de nos tests d'équivalence, présentés dans la figure 5.5, des informations intéressantes peuvent en être extraites. 1) Le comportement de notre système est généralement proche des capacités humaines. 19 tests sur les 21 valident notre hypothèse avec une p-valeur inférieure à la valeur de signification. 2) Le contexte et l'état de croyance de l'homme sont à prendre en compte. Un système qui ignorerait ces deux facteurs aurait pu reconnaître correctement que l'intention du test *No Clues*. 3) Certains raisonnements présents chez l'homme sont encore manquants dans notre système. Nous avons échoué à rejeter l'hypothèse nulle pour deux cas. Dans *Divergent Belief Bob* les humains ont donné une note plus importante à l'intention *Eat Cookie* que l'intention *Drink Water*. Une explication est qu'ils ont pensé que, étant donné que *Bob* a rempli la *Boîte de cookies*, il y a plus de chance qu'il souhaite manger un *Cookie*. Cela permet de penser que l'homme utilise des raisonnements temporels complexes pour évaluer l'intention, en considérant tout l'historique des actions des agents pour deviner leur intention.

5.7 Conclusion

Dans ce chapitre, nous avons présenté un système capable d'évaluer les intentions humaines en utilisant la gestion de la croyance, les données contextuelles et l'observation d'actions. Nous avons effectué une étude sur les utilisateurs où nous avons comparé les prédictions de notre système avec ceux des humains. Dans cette étude, nous avons montré que notre système est en mesure, dans plusieurs situations, d'approcher les capacités humaines. Notre système utilise la prédiction de l'intention afin d'aider de manière proactive les humains à atteindre leurs objectifs en effectuant une partie du plan ou en corrigeant verbalement leur état de croyance.

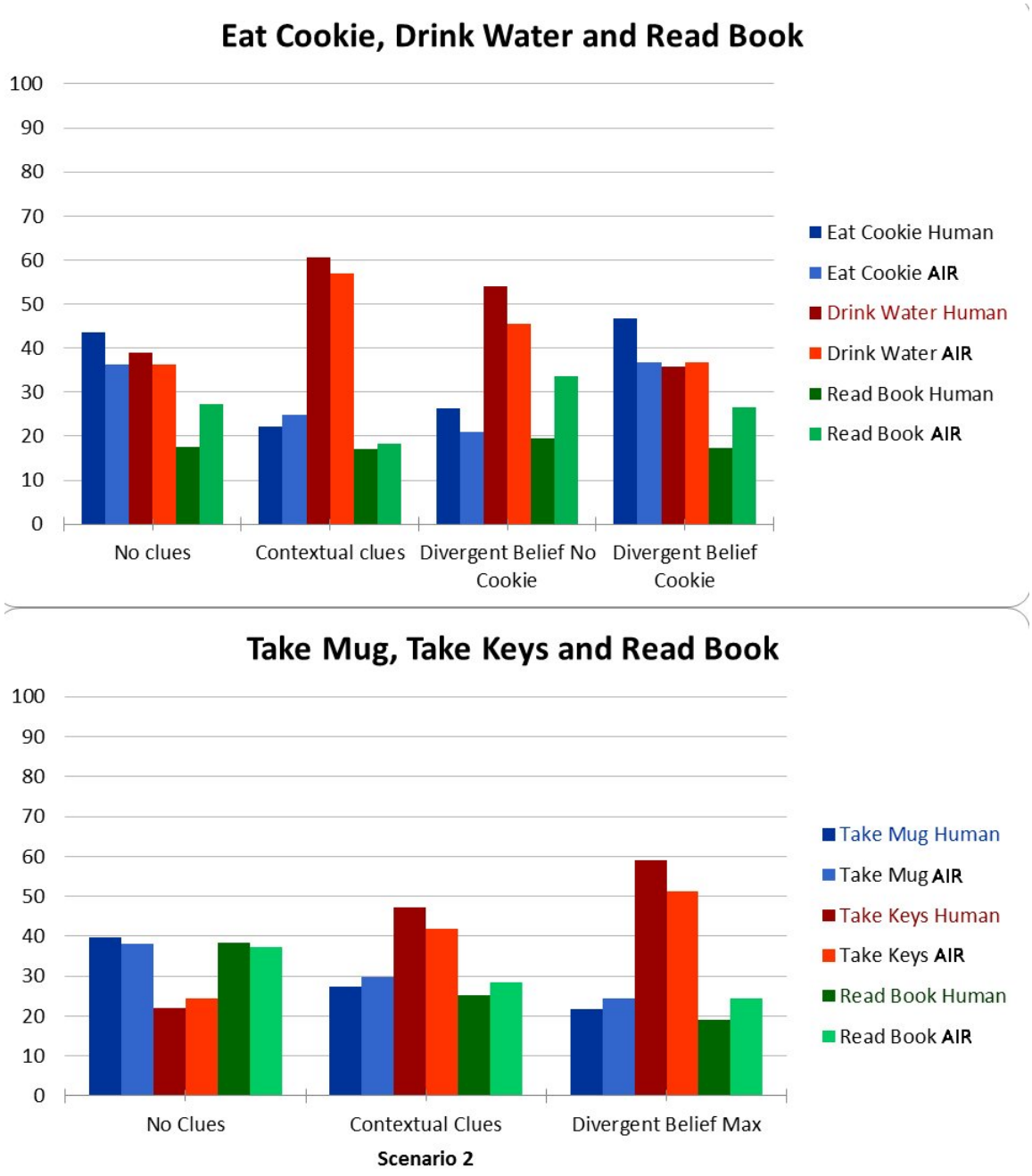


FIGURE 5.5 – Résultats expérimentaux. Les résultats pour les deux scénarios sont représentés sous forme de graphes. Les intentions, estimées par les hommes et le système de reconnaissance d'intention (AIR), sont représentées sous différentes couleurs (voir légende). L'estimation des mêmes intentions par le système et par les hommes sont placées côte à côte. Chaque colonne représente la probabilité d'une intention exprimée en pour-cent. Les P-valeurs du test d'équivalence sont ajoutées au graphe.

Les résultats obtenus mettent également en valeur que les humains utilisent probablement des raisonnements temporels pour désambiguïser les intentions (comme indiqué dans la section 5.6.3). Il serait intéressant d'étudier d'avantage ces mécanismes et d'utiliser notre base de données temporelles présentée en 2.6.2 pour ajouter ce type de mécanisme au robot.

Nous supposons, dans ce travail, que le robot a toujours un état de croyance correcte et l'utilise comme une référence pour l'état de l'environnement. Il serait intéressant d'aborder le problème où le robot a un état de croyance erronée, par exemple en analysant le degré de confiance que l'homme semble avoir dans ses comportements. Enfin, ces expériences ont été réalisées en simulant les observations. Cette première simulation a permis d'isoler le système AIR pour mieux tester ses performances. Nous projetons à l'avenir de tester les performances en conditions réel afin de valider notre approche.

Chapitre 6

Prise en compte de l'Expertise Humaine et Adaptation de la Collaboration

Sommaire

6.1 Contexte et motivation

L'un des défis de la robotique est de faire collaborer convenablement les robots avec des partenaires humains durant l'accomplissement de tâches. Collaborer avec un humain nécessite de la part du robot de planifier ses propres actions, d'anticiper les contributions de son partenaire au plan global, de surveiller les actions potentiellement complexes de l'homme et de maintenir un état du monde à jour. La collaboration doit, en plus de cela, avoir une dimension sociale suffisamment importante pour garantir le confort de l'homme ainsi qu'une lisibilité des actions et décisions du robot afin que le système soit intuitif. Le comportement social a également une influence importante sur l'acceptation du robot par l'homme.

Nous définissons un plan collaboratif, ou plan partagé, comme un ensemble d'actions ordonnées, impliquant plusieurs agents qui coopèrent afin d'avancer vers un objectif commun. Pour produire un plan partagé, le robot devrait prendre en compte non seulement la configuration de son environnement, mais également son/ses partenaire(s) humain(s). Un moyen pragmatique de prendre en compte l'humain consiste à calculer les affordances de celui-ci, lui permettant de créer des plans plus pertinents et pour lesquels l'homme est dans la capacité "physique" d'accomplir les tâches qui lui sont confiées. La plupart des travaux menés jusque là prennent au mieux ces limitations liées aux capacités ou à l'effort humain en compte lors de la planification de tâches, supposant ainsi que l'humain sait comment réaliser les actions qui lui sont assignées. Nous pensons que pour améliorer l'interaction d'un point de vue social, le robot devrait également pouvoir adapter le plan aux préférences et à l'expertise de l'utilisateur. Idéalement, cette adaptation devrait pouvoir se faire au niveau de chaque tâche du plan.

Une fois que le robot a produit un plan pour atteindre l'objectif des agents (hommes et robots) impliqués, il faut pouvoir le partager avec le partenaire humain afin de s'assurer qu'il soit informé des tâches qui lui incombent, et qu'il accepte de les mener à bien. Lorsque le plan et l'objectif sont suffisamment simples, les jeunes enfants sont capables de collaborer sans utiliser le langage, simplement en estimant l'intention et les actions de l'autre. Dans les situations nécessitant des plans plus complexes, le langage est la modalité préférée [Warneken 2006, Warneken 2007]. Cependant, expliquer la totalité du plan en détaillant chaque action unitaire (par exemple "tendre le bras puis fermer la main" pour attraper un objet) d'un seul coup serait inefficace et ennuyeux pour le partenaire humain, tout particulièrement s'il a déjà la connaissance de la façon de réaliser certaines actions. Se pose donc la question du niveau de détail à donner comme indication au partenaire humain. Pour reprendre l'exemple d'attraper un objet, cette tâche fait partie des connais-

sances "universelles" chez l'homme, à savoir que le robot devrait estimer connue peu importe l'identité de l'humain avec lequel il collabore. Il serait donc inutile de décomposer la tâche d'attraper un objet pour détailler les actions (ou sous-tâches) liées à cette tâche.

Les recherches menées en psychologie et en philosophie ont conduit à une meilleure compréhension du comportement humain durant l'action jointe et la collaboration, afin de savoir comment [Tomasello 2005] et pourquoi [Tomasello 2011] l'homme collabore et ce qui est partagé durant ce processus [Butterfill 2011]. Ces recherches sur l'action jointe ont été utilisées en robotique pour accomplir des tâches faisant intervenir un partenaire humain. Alors qu'un nombre substantiel de travaux dans le domaine cherchent à résoudre le problème de génération de plan pour les buts collaboratifs impliquant un partenaire humain [Lallement 2014], seulement quelques uns étudient comment adapter efficacement la génération de plan et son exécution à l'expertise de l'homme. Jusqu'alors nos études portant sur des tâches collaboratives supposaient que les hommes avaient la connaissance suffisante pour exécuter les tâches du plan. Ici nous considérons que ces connaissances peuvent être insuffisantes, voir inexistantes et nous cherchons à représenter et actualiser correctement les informations sur les connaissances de l'homme, qui évoluent au cours des interactions. Nous affirmons qu'une telle adaptation à l'utilisateur améliore considérablement les capacités sociales du robot au cours de l'interaction collaborative avec un partenaire humain.

Nous allons tout d'abord expliquer comment nous estimons et suivons l'évolution de la connaissance de l'homme sur chaque nœud présent dans le plan collaboratif (dans notre recherche nous utilisons des plans de type HTN, pour Hierarchical Task Network), depuis le haut niveau d'abstraction de tâches aux actions atomiques. Nous allons par la suite présenter comment nous sommes capables de prendre en compte l'homme durant la génération du plan pour la collaboration. Puis nous donnerons des détails sur la façon dont nous tirons avantage de la structure d'arbre du HTN produit pour 1) présenter et négocier le plan partagé, et 2) expliquer et surveiller les tâches en fonction du niveau de connaissance de l'utilisateur sur chaque tâche, afin de guider ou enseigner l'homme lorsque nécessaire et d'adapter la surveillance de ses actions. Enfin, nous présenterons une implémentation du système et une étude comparative impliquant deux groupes distincts d'utilisateurs et dont nous discuterons les résultats obtenus [1].

1. Une version en anglais présentant ces travaux est disponible dans l'article [Milliez 2016]

6.2 Travaux associés

Des recherches précédentes tel que [Grosz 1996, Grosz 1999, Bratman 2013], ont permis de montrer la pertinence d'utiliser un plan partagé pour la collaboration, et notamment la collaboration homme-robot [Lallee 2013]. Ainsi, l'utilisation de plan permet au robot de guider la prise de tour pour accomplir l'exécution. Les auteurs décrivent également des expériences réalisées avec des sujets naïfs et suggèrent que le plan partagé devrait être totalement communiqué afin de soutenir une collaboration effective. Dans [Petit 2013], le dialogue est utilisé pour apprendre de nouveaux plans au robot et pour modifier ces plans. Une des façons pour le robot d'apprendre est d'utiliser "la programmation par le langage". Pour ce faire, un humain explique verbalement les tâches au robot, à savoir quelle suite d'actions permet d'accomplir la tâche en question. Cependant, ce travail ne traite pas de la situation où c'est au robot d'expliquer une tâche à l'humain. Dans [Sorce 2015], le système est capable d'apprendre un plan (sous forme de décomposition de tâches de différents niveaux d'abstraction) d'un utilisateur afin de pouvoir le retransmettre à un autre utilisateur. L'article illustre cette transmission de connaissance de tâches collaboratives homme-robot par un scénario sur une station spatiale où des cosmonautes se succèdent et sont amenés à collaborer avec un robot. Dans cette étude, le robot a deux modalités différentes pour adapter son comportement à l'utilisateur : un mode débutant et un mode expert. Notre contribution cherche à créer un système plus adaptatif en ayant un suivi du niveau de connaissances de chaque agent pour chaque tâche et sous-tâche avec une génération en ligne du plan collaboratif.

Comme présenté dans les chapitres précédents, raisonner sur les capacités de l'homme et son état mental (ce qui revient à le doter de la capacité de prise de perspective décrite dans le chapitre 3) est essentiel pour avoir un robot qui soit capable d'interagir socialement, qui soit accepté par l'homme et qui soit un partenaire efficace et pertinent. Dans les travaux présentés ici, nous incorporons un modèle de connaissance du partenaire (humain) concernant les tâches présentes dans le plan collaboratif à effectuer, et ce afin d'obtenir un système qui s'adapte à l'homme pour les actions jointes. Les recherches sur les systèmes de tuteurs intelligents (ITS pour Intelligent Tutoring System) [Brusilovskiy 1994] ainsi que celles sur le e-learning [Brusilovsky 2005], ont prouvé la nécessité de garder et mettre à jour un modèle de connaissances de l'apprenti afin d'enseigner correctement une tâche. Dans nos travaux, nous maintenons et actualisons le niveau de connaissance de l'utilisateur pour chaque tâche et couplons cette information avec l'utilisation de plans hiérarchiques pour gérer l'interaction. L'idée n'est pas d'enseigner un plan à l'utilisateur mais de mettre à profit son modèle de connaissance pour adapter la génération de plans à la

politique de l'interaction (est-ce que l'efficacité est recherchée, ou le fait d'enseigner de nouvelles tâches ?), et au cours de l'exécution, adapter le niveau d'explication de tâches et la surveillance de celles-ci au collaborateur.

Certains systèmes ont une modélisation explicite du plan partagé durant l'exécution de tâche, ce qui permet au robot d'adapter ses plan aux actions de l'homme, comme le robot Pike [Levine 2014, Karpas 2015], et Chaski [Shah 2011]. Dans [St. Clair 2015] le dialogue est utilisé durant l'exécution du plan collaboratif pour améliorer les performances de l'équipe. Leur approche est basée sur les processus de décisions de Markov (MDP) et donne de l'importance au concept du rôle d'un agent dans une tâche, qui peut être estimé et influencé en utilisant le dialogue.

Des études en psychologie montrent que les humains forment une représentation jointe de la tâche, ce qui inclut les actions que chaque partenaire devrait accomplir [Sebanz 2006]. La surveillance de l'exécution doit alors être liée à cette représentation partagée des tâches afin de mieux suivre le niveau d'engagement de chaque membre dans l'action jointe, et si il y a des erreurs qui ont besoin d'être prises en charge.

6.3 Suivi des connaissances de l'homme

6.3.1 Estimation de la situation et état mental

Pour évaluer l'état de connaissances de son collaborateur, le robot a besoin de comprendre la situation et d'en extraire des informations sur les agents. Pour ce faire, et pour maintenir un état du monde cohérent, nous utilisons l'infrastructure logicielle TOASTER décrite dans le premier chapitre de ce manuscrit. L'état du monde produit par ce module de raisonnement spatio-temporel sera utilisé par notre générateur de plan pour calculer un plan adapté à la situation.

L'utilisation du système d'évaluation de la situation, a permis dans nos autres travaux de gérer avec succès un état de croyance pour chaque agent, robot et humain. Le modèle de l'état de croyance de chaque agent est indépendant et logiquement cohérent. La croyance du robot en ce qui concerne l'état mental de ses homologues sur l'environnement est représenté dans ces modèles (Comme expliqué dans les autres chapitres de ce manuscrit).

Dans ces travaux, nous nous concentrons sur la connaissance de l'homme concernant diverses tâches. Cette connaissance est représentée par un vecteur $<$ HUMAN, TASK, PARAMETERS, VALUE $>$. *HUMAN* représente l'homme ayant cette connaissance, *TASK* est le nom de la tâche, *PARAMETERS* la liste des paramètres pertinents pour décrire la connaissance se rapportant à la tâche (voir

ci-dessous) et *VALUE* est la valeur (ou le niveau) de connaissance de l'homme concernant la tâche.

Par exemple, le fait que *Bob* a une connaissance d'*expert* concernant la tâche d'assemblage d'un morceau de meuble A avec un morceau B serait représenté par : <*Bob, assemble, [A,B], EXPERT*>. Les valeurs possibles de connaissance sont, dans l'ordre croissant, *NEW*, *BEGINNER*, *INTERMEDIATE* et *EXPERT*.

Certaines tâches peuvent être considérées comme des connaissances universelles. Par exemple, mettre des ingrédients dans un bol est considéré comme assez simple pour être une action connue pour tout être humain. Ce genre de tâches sera alors étiqueté comme connaissance universelle et considéré comme connu par les utilisateurs, peu importe les paramètres. Certaines autres connaissances liées aux tâches peuvent différer en fonction des paramètres. Pour ces tâches, il se peut que la connaissance soit liée à un "type" de paramètre au lieu d'une instance de cette classe. A titre d'exemple, on peut considérer que si un homme sait comment peindre la salle de séjour, il saura comment peindre une pièce (peu importe quelle pièce). Dans ce cas, nous allons mettre dans sa connaissance le type "pièce" pour la tâche de peindre au lieu de l'instance "salle de séjour". Pour résumer, certaines tâches peuvent être étiquetées comme connaissance universelle alors que d'autres tâches peuvent être décrites par certains de leurs paramètres ou type de paramètres. Ce formalisme de représentation de la tâche exige un expert du domaine pour indiquer comment représenter la connaissance liée à chaque tâche potentiellement présente dans le plan.

6.3.2 Niveau de connaissance de tâche

Dans ce contexte où le robot génère le plan partagé, nous supposons qu'il connaît toutes les tâches nécessaires dans le plan. En ce qui concerne le collaborateur, nous définissons quatre niveaux de connaissance de tâche qui mèneront à des comportements différents de la part du robot.

— *NEW* : cette valeur sera utilisée pour les tâches qui n'ont jamais été effectuées par l'utilisateur. Si l'utilisateur observe l'exécution de la tâche avec des explications ou s'il l'effectue lui-même, la valeur sera modifiée à *BEGINNER*. Toutefois, si l'utilisateur observe l'exécution de la tâche, sans aucune explication, nous gardons le niveau *NEW*. Le choix a été fait de considérer que dans cette situation l'utilisateur n'a pas reçu suffisamment d'informations pour relier l'observation à la tâche.

— *BEGINNER* : cette valeur sera utilisée pour les utilisateurs qui ont déjà accompli la tâche, mais qui ont potentiellement encore besoin d'explications

afin de l'exécuter à nouveau. Si l'utilisateur exécute à nouveau la tâche avec succès, sans demander d'explications, la valeur de connaissance est mise à *INTERMEDIATE* et dans tous les autres cas la valeur sera dégradée à *NEW*.

— *INTERMEDIATE* : cette valeur sera utilisée pour les utilisateurs qui sont en mesure d'accomplir la tâche sans directives. Si l'utilisateur exécute avec succès la tâche à nouveau, la valeur est mise à *EXPERT*. En cas d'échec, elle est rétrogradée à *BEGINNER*.

— *EXPERT* : ce niveau de connaissance sera utilisé pour les utilisateurs qui sont en mesure d'accomplir la tâche sans directives et sont assez expérimentés pour l'expliquer à un tiers. Si l'utilisateur ne parvient pas à effectuer la tâche, sa valeur de connaissance associée est rétrogradée à *INTERMEDIATE*.

Ces niveaux de connaissance de tâche permettent l'adaptation de la génération de plans collaboratifs, du niveau d'explication des tâches et également du niveau de suivi des interventions de l'homme.

6.4 Planificateur HTN s'adaptant à l'homme

Une fois que le système est capable de suivre et modéliser les connaissances de l'homme sur diverses tâches, il doit également pouvoir générer un plan collaboratif.

Dans cette étude, le choix a été fait d'utiliser HATP (Hierarchical Agent-based Task Planner). Ce planificateur est développé au LAAS-CNRS par Raphaël Lallement qui a contribué à l'étude présentée dans ce chapitre.

Nous utilisons une approche hiérarchique de planification car cette méthode offre une meilleure compréhension du contexte dans lequel un agent est invité à effectuer une action. Ce contexte est bénéfique pour l'explication du plan. En effet, il met en œuvre un processus de raffinement contextuel itératif. Par conséquent, le système peut guider les utilisateurs en leur disant pourquoi ils devraient effectuer une action et la façon dont elle est liée à des parties précédentes et suivantes du plan. Nous utilisons un HTN, qui assure la représentation de domaine et une planification efficace. L'HTN est beaucoup plus efficace que la planification classique car l'expert du domaine peut guider le processus de recherche vers l'objectif en fournissant une représentation correcte. Une très célèbre application de HTN est SHOP [Nau 1999].

Le planificateur utilisé, HATP, est un HTN modifié spécialement conçu pour la robotique [Lallement 2014]. Il est livré avec des fonctions spécifiques concernant la production du plan, telles que :

— Agent based : il calcule les plans multi-agents faisant intervenir les humains et les robots.

— Cost driven : le meilleur (ou un "suffisamment" bon) plan est trouvé plus tôt

(en utilisant l'élagage de plan).

— social rules : il affine les plans selon un ensemble de règles destinées à promouvoir l'acceptabilité sociale des plans (par exemple l'équilibre des efforts en fonction des préférences humaines et du contexte, les conventions sociales ...).

Chaque action dans le domaine dispose d'une fonction qui permet d'estimer son coût si elle est ajoutée au plan. Donc, à tout moment il est possible de calculer le coût du plan partiel alors qu'il est en cours de construction. En outre, le score du meilleur plan actuel est stocké ; si, à un moment, le coût du plan partiel courant dépasse ce score, le plan est rejeté et la recherche se poursuit. Cet élagage de plan permet d'accélérer la recherche du meilleur plan. Après que chaque plan ait été calculé, un ensemble de règles de filtrage sont appliquées pour sanctionner les plans qui ne présentent pas certains comportements sociaux. Une fois que le meilleur plan (notez que l'on peut limiter la recherche à un niveau de coût "suffisamment" bon) est récupéré, il est envoyé au superviseur sous la forme d'un arbre de décomposition. En outre, un ensemble de flux d'actions est élaboré ; chaque flux représente les actions qu'un agent (humain ou robot) doit effectuer. Pour assurer le séquençage de l'action appropriée et la synchronisation entre les agents, les liens de causalité sont intégrés. Par ailleurs, le plan peut inclure des actions conjointes attribuées simultanément à deux ou plusieurs agents parce qu'ils ont besoin d'être en étroite collaboration (par exemple dans le cas d'un "handover", ou transfert d'objet). La figure 6.1 dépeint la décomposition de l'arbre d'un plan solution pour cuisiner une tarte à la banane. La figure 6.2 présente le flux d'actions associé au plan pour cuisiner la tarte.

FIGURE 6.1 – Arbre de décomposition d'un plan pour faire une tarte à la banane.

FIGURE 6.2 – Flux d'actions lié au plan pour faire une tarte à la banane.

Pour prendre l'expertise en compte lors de la planification, une nouvelle règle sociale est nécessaire. L'objectif est de sélectionner le meilleur plan adapté à la politique choisie. Nous proposons deux politiques : la préférence de l'enseignement (sous forme de démonstration ou de réalisation accompagnée d'explications) où l'être humain peut apprendre du robot, ou le choix de l'efficacité. Avec la politique de l'enseignement, le planificateur tente de produire des plans afin de maximiser le nombre de tâches humaines où l'humain a l'occasion d'apprendre, alors que l'efficacité pousse le planificateur à sélectionner des plans avec le moins de tâches inconnues pour l'être humain, afin de veiller à ce qu'il puisse être plus efficace (le robot peut cependant s'assigner certaines tâches inconnues à l'homme). Dans le cas d'une politique visant à favoriser l'efficacité, la règle est tout simplement d'appliquer une pénalité à chaque fois qu'un humain doit effectuer une action qu'il ignore. Cette pénalité serait inversée pour la règle de l'enseignement.

Pour illustrer notre planification et la nouvelle règle sociale nous prenons l'exemple où un être humain et un robot doivent cuisiner une tarte aux pommes. Une partie du plan solution est représenté sur la figure **??**. Dans ce contexte, on peut considérer que l'homme sait comment mener à bien toutes les actions (prendre, poser, couper, et ainsi de suite), mais il se peut qu'il ignore l'ordre exact des différentes étapes de préparation (tâche de niveau supérieur). Si nous sommes favorables à l'enseignement, le plan devrait contenir un moyen de réaliser la recette avec un niveau de connaissance minimal sur chaque tâche et, autant que possible, l'homme sera en charge de ces étapes. D'autre part, si nous sommes favorables à l'efficacité, le plan devrait contenir la plus petite quantité de tâches inconnues à effectuer par l'utilisateur. En utilisant cette règle, le robot est capable d'adapter sa génération du plan à la connaissance de l'utilisateur concernant les tâches contenues dans le plan partagé. Pour calculer correctement le coût d'un plan, le planificateur considère également la connaissance d'une tâche comme améliorée après qu'elle ait été ajoutée au plan. Cela permet, lors de l'utilisation de la politique d'efficacité, de préférer les plans qui réutilisent de nombreuses fois la même tâche en l'affectant à un même utilisateur en réduisant le coût pour les prochaines occurrences. Par opposition, si différentes tâches sont utilisées ou si l'agent change en cours de route, le coût sera plus élevé. Ceci permet de prendre en compte la connaissance acquise au cours de l'exécution lors du calcul de coût.

Le planificateur est intégré dans un système de dialogue qui permet de négocier des plans (voir la section 6.5.3). Plus précisément, il permet de poser des questions sur les préférences des utilisateurs et leurs capacités à accomplir une tâche. Si l'utilisateur indique au système qu'il ne peut pas effectuer une tâche donnée, elle ne sera pas ajoutée au plan ou tout du moins ne lui sera pas attribuée (invalidation de la

condition préalable de la tâche correspondante). En ce qui concerne les préférences des utilisateurs, l'étape de la négociation mettra à jour une base de données avec les préférences exprimées par l'utilisateur. Si l'utilisateur spécifie qu'il veut (ou non) effectuer certaines tâches, ces tâches, si elles sont ajoutées au plan et lui sont assignées, auront une récompense importante (respectivement pénalité) au niveau des coûts. Par conséquent, les plans qui contiennent de telles tâches seront considérés comme indésirables. Cependant, s'ils sont les seules solutions possibles (en raison de l'incapacité, etc.), ils seront conservés et le planificateur donnera comme solution l'un de ces plans avec le moins de tâches indésirables et le maximum de tâches voulues attribuées à l'homme.

6.5 Présentation et négociation du plan partagé

Une fois que le système robotique a produit un plan collaboratif adapté à la politique choisie (prenant en compte l'apprentissage, les capacités et les préférences), le plan doit être transmis au partenaire humain. La parole est une "modalité puissante pour l'entretien continu de l'interaction coopérative" [Lallee 2013]. En effet, Tomasello suggère même que la principale fonctionnalité du langage est d'établir et de négocier des plans coopératifs [Tomasello 2005]. En considérant cela, nous décidons d'utiliser la parole pour présenter le plan au collaborateur.

6.5.1 Prétraitement du plan

L'arbre HTN créé représente une solution pour accomplir le but. Cependant, il se peut qu'en l'état il ne convienne pas à la présentation ou à l'explication du plan au collaborateur, car il peut contenir des étapes de raffinement qui allongeraient l'explication ou même pourrait la rendre confuse.

Par exemple, lors de la planification, certaines méthodes peuvent être récursives. En effet, si on reprend le domaine de planification décrit dans le chapitre 4 en section 4.5.1, le robot devait explorer les zones (*Localtion*) d'une pièce (*Room*) n'ayant pas encore étaient explorées. Une telle méthode peut être décrite de façon récursive en commençant par explorer une première zone L non explorée puis en rappelant la méthode d'exploration. Le second appel à la même méthode explorera alors une autre zone L2 car la zone L est déjà explorée, et ainsi de suite jusqu'à ce que toutes les salles soient explorées. Pour illustrer, nous présentons le code du domaine HATP associé à cette méthode. La condition *isIn* symbolise que la zone est dans la pièce et la condition *scanned* permet de savoir si la zone a déjà été explorée.

```
method LocaliseInRoom(Room R, Object O){
```

```
  goal{
    OR{
      A.scanned==true;
      FORALL(Location L, {L.isIn==R;}, {L.scanned==true;});
    }{
      FORALL(Location L2, {}, {L2.isIn!=R;});
    }
  };
  {
    preconditions{R.scanned==false;};
    subtasks{
      L=SELECT(Location, {L.isIn==R; L.scanned==false;});

      1: Check(L, O);
      2: LocaliseInArea(A, O);
    };
  }
}
```

Dans cet exemple, la récursion est donc utile pour planifier efficacement l'exploration de toutes les zones de la pièce qui n'ont pas encore été explorées. Ici, pour expliquer ce type de méthode récursives, nous avons fait le choix de retirer l'aspect récursif qui est un aspect intéressant pour la planification mais qui peu complexifier les explications à procurer. Le robot dirait donc simplement à l'homme d'explorer la première salle puis la seconde puis la suivante...

D'autres méthodes peuvent n'avoir qu'une seule fille. Par exemple la méthode consistant à donner un objet au robot peut contenir un déplacement vers l'objet, une préhension puis un autre déplacement vers le robot pour enfin tendre l'objet au robot. Cependant si l'agent en charge de cette tâche est déjà en face du robot et a l'objet dans la main, il n'aura plus qu'à tendre l'objet au robot. Dans ce second cas, la méthode "donner l'objet au robot" aura une seule méthode fille qui est "tendre l'objet au robot". Nous faisons le choix ici de ne pas verbaliser les deux nœuds mais simplement le plus bas pour garder la partie la plus explicite.

Pour adapter le plan à l'explication nous utilisons deux règles. (1) Nous retirons les tâches récursives. Si un nœud n de l'arbre HTN contient la même méthode (en utilisant la fonction *compare*) que son père $parent(n)$, il sera remplacé dans l'arbre par ses fils $children(n)$. (2) Nous remplaçons également les nœuds avec un fils unique par ce fils.

1. **if** $(compare(n, parent(n)))$ **then** $n \leftarrow children(n)$
2. **if** $(children(n).size() = 1)$ **then** $n \leftarrow children(n)$

Ces règles permettent de réduire la taille de l'arbre à traiter et de le rendre plus adapté à la verbalisation et à la surveillance de l'activité de l'homme. Ces traitements sont illustrés par la figure 6.3.

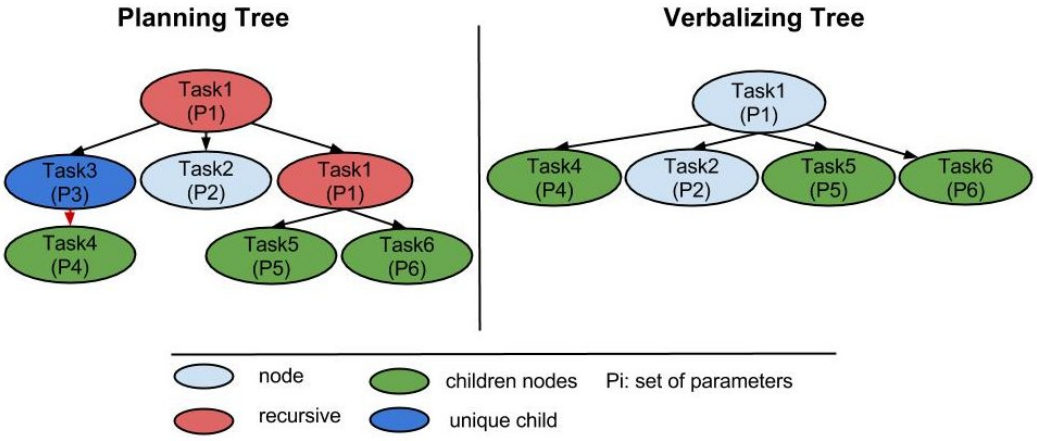


FIGURE 6.3 – Schéma expliquant le prétraitement de l'arbre de planification pour l'adapter à la verbalisation.

6.5.2 Présentation de plan

Avant de pouvoir commencer l'exécution du plan, le robot présente le plan à l'homme et les allocations des tâches de haut niveau pour donner un aperçu global du plan. Une génération de langage naturel (NL pour Natural Language) est utilisée comme présenté dans le tableau 6.1. Pour assurer le bon fonctionnement du système avec des plans de différentes tailles (scalability), lors de la présentation du plan, le robot verbalise seulement les N premières tâches de haut niveau. Par simplicité nous avons choisi N=3. Ce chiffre a été choisi de façon empirique, en effectuant quelques tests lors du développement. Nous pensons qu'établir ce nombre de façon optimale nécessiterait de mener des investigations plus poussées, dépendant du domaine ou de l'utilisateur et de son aisance à accomplir les tâches. Le robot présente les N premières étapes du plan, puis les exécute avec son partenaire. Une fois que cette exécution est achevée, le processus de présenter/négocier/exécuter se répète jusqu'à ce que le plan soit achevé ou abandonné.

agents(root) + have_to + root	"We have to cook an apple pie."
introduce_presentation	"I will tell you the first steps."
agents(child[0]) + first + child[0]	"You will first prepare the dough,"
then + agents(child[1]) + child[1]	"Then I will prepare the mixture,"
finally + agents(child[2]) + child[2]	"Finally, you will prepare the banana."

TABLE 6.1 – Présentation d'un plan pour cuisiner une tarte à la banane comme celui présenter à la figure 6.1. *root* désigne la racine de l'arbre et *child* est la liste de ses fils. Une vidéo avec la présentation d'un plan similaire peut être consultée sur `http://homepages.laas.fr/gmilliez/hri2016/`

6.5.3 Négociation de plan

Une fois que le robot a présenté les tâches principales et la répartition, il lui faut s'assurer que l'homme accepte ce plan partagé. Le robot va simplement demander l'approbation de l'homme et, en cas de désaccord, lui demander ce qui ne lui convient pas. Dans la version actuelle de notre système, deux types de requêtes provenant de l'homme sont prises en charge. Premièrement, l'homme peut exprimer ses préférences. Cela peut être la volonté de faire une tâche qui est assignée au robot ou le refus d'accomplir une tâche qui lui est assignée. La seconde possibilité est d'informer le robot que l'utilisateur ne peut réaliser une action. Dans les deux cas, ces informations seront ajoutées au modèle de l'utilisateur et enregistrées dans la base de données. Le robot va ensuite essayer de trouver un nouveau plan qui évitera à l'humain d'effectuer une tâche qu'il ne souhaite pas ou qu'il n'est pas capable de réaliser. Ce nouveau plan sera alors présenté et le robot demandera à nouveau l'approbation de l'homme. Dans notre système, les préférences des utilisateurs ont un coût plus élevé que la politique employée (apprentissage ou efficacité) car nous considérons que l'utilisateur devrait avoir la décision finale.

6.6 Exécution de plan adaptative

6.6.1 Algorithme de gestion de plan

Une fois que le plan a été présenté et accepté par le collaborateur, l'exécution peut commencer. Nous donnons l'algorithme pour l'exécution adaptative de plan, puis nous donnons les explications associées.

— *execute_tree*(n) est la fonction principale pour la gestion de l'exécution. Celle-ci est appelée après le processus de négociation. Elle a pour paramètre *nodes*, une liste de nœuds initialement remplie avec les fils du nœud racine du HTN.

— *teachPolicy* est un booléen qui permet de définir si nous voulons utiliser la

Algorithm 1 *execute_tree*(*n*)

```
1: for n:=nodes.start to n:=nodes.end do
2:     if agents(n) = {robot} then
3:         if children(n) ≠ ∅ ∧ user_kn(n) = NEW
            ∧ teachPolicy then
4:             execute_tree(children(n))
5:             user_kn(n) := BEGINNER
6:         else
7:             execute(n)
8:         end if
9:     else if user_kn(n) = NEW then
10:        explain(n)
11:        if children(n) ≠ ∅ then
12:            execute_tree(children(n))
13:            user_kn(n) := BEGINNER
14:        else
15:            monitor(n)
16:        end if
17:    else if user_kn(n) = BEGINNER then
18:        if propose_explain(n) then
19:            user_kn(n) := NEW
20:            (...)                          ▷ Same process as NEW
21:        else
22:            monitor(n)
23:        end if
24:    else if user_kn(n) = INTERMEDIATE
            ∨ user_kn(n) = EXPERT then
25:        monitor(n)
26:    end if
27: end for
```

politique d'enseignement ou d'efficacité.

- *agents*(n) cette fonction renvoie la liste d'agents impliqués dans le nœud n (plus précisément impliqués dans la tâche liée au nœud).
- *verbalize*(n) cette fonction permet de verbaliser la tâche courante, en utilisant le contexte du nœud pour la présenter (e.g. en utilisant des relations séquentielles tel que "first" (premièrement), "then" (puis) ou "finally" (enfin) en fonction de la position du nœud dans la liste).
- *user_kn*(n) cette fonction donne le niveau de connaissance de l'utilisateur concernant la tâche n.
- *propose_explain*(n) cette fonction va conduire le robot à proposer une explication pour la tâche courante. Si l'utilisateur accepte l'explication, la fonction renvoie "true" et "false" sinon.
- *explain*(n) cette fonction lance une procédure pour expliquer la tâche actuelle à l'utilisateur. Cette procédure peut être implémentée comme un script pour lancer une vidéo, une explication verbale, ou même de demander à un expert d'expliquer la tâche.
- *monitor*(n) cette fonction envoie une requête au système de supervision pour qu'il suive l'exécution du nœud courant afin de s'assurer qu'elle se déroule correctement. Si la requête renvoie un succès, cela signifie que la tâche correspondant au nœud a été effectuée correctement. Par conséquent la fonction améliorera la connaissance de l'humain puis la fonction *execute_tree* continue de s'exécuter. Dans le cas d'un échec, la fonction dégradera le niveau de connaissance de l'utilisateur, la fonction *execute_tree* sera stoppée, et va renvoyer une erreur qui va entraîner une requête de replanification au superviseur et une nouvelle exécution si un plan est trouvé.
- *execute*(n) cette fonction procède de façon similaire à la fonction de suivi (monitor(n)). La différence est qu'elle envoie une requête pour que le robot exécute le nœud.

6.6.2 Explication de l'algorithme de gestion de plan

Pendant l'exécution, nous utilisons l'arbre HTN pré-traité pour réaliser le plan, donner des explications et surveiller la réalisation des tâches en fonction du niveau de connaissance de la tâche contenue dans la modèle de l'humain. Nous utilisons une procédure d'exploration du plan en profondeur, ou "depth first search" pour procéder durant l'exécution. Cela permet de donner le contexte à la tâche à accomplir. Lorsque le processus atteint un nœud, plusieurs situations peuvent se présenter.

6.6.2.1 Le robot est le seul impliqué (lignes 2- 8)

Si le robot est le seul agent en charge du nœud courant, si le collaborateur a un niveau de connaissance égal à *NEW* pour la tâche actuelle et si la politique choisie pour l'interaction est l'apprentissage, le robot va exécuter les sous-tâches en mode "démonstration", ce qui signifie qu'il va verbaliser chaque tâche-fille avant de la réaliser. Une fois la tâche accomplie, le robot met à jour la connaissance de l'homme sur le nœud courant en lui donnant la valeur *BEGINNER*. La même procédure sera appliquée aux fils, afin que le robot verbalise chaque tâche qui doit être apprise par le collaborateur (et uniquement celles-ci). Si le robot est en charge du nœud, mais que le collaborateur humain a déjà les connaissances suffisantes sur le nœud, ou que la politique d'interaction est l'efficacité, le robot ne verbalisera que la tâche de plus haut niveau qu'il a à accomplir.

Dans le cas où l'humain est impliqué dans le nœud courant, le comportement du robot va dépendre des connaissances de l'humain sur la tâche, car il se peut que le robot doive expliquer celle-ci ou adapter la surveillance des tâches à accomplir par l'homme. L'explication pourrait se faire de différentes manières : en montrant une vidéo, en demandant à un expert d'expliquer la tâche ou simplement en guidant verbalement l'utilisateur, étape par étape. Nous donnerons des détails sur l'explication verbale du robot à l'utilisateur car c'est la méthode qui implique réellement le robot.

6.6.2.2 Le collaborateur est *NEW* (lignes 9- 16)

Si l'humain a un niveau *NEW* pour la tâche actuelle, le robot l'explique. Lorsque le robot guide verbalement l'utilisateur, si le nœud courant a des fils, nous allons plus profondément dans l'arbre et appliquons à nouveau le comportement approprié en fonction du niveau de connaissance aux nœuds fils. Si le nœud actuel n'a pas de fils (est une feuille), le superviseur attend que l'utilisateur accomplisse l'action courante. En cas de succès, le niveau de connaissance pour la tâche est mise au niveau *BEGINNER* (dans l'algorithme ci-dessus, cette étape est faite dans la fonction *monitor*).

6.6.2.3 Le collaborateur est *BEGINNER* (lignes 17- 23)

Si l'humain a un niveau de connaissance à *BEGINNER* pour la tâche courante, nous demandons s'il a besoin d'explications. Si c'est le cas, nous dégradons son niveau de connaissance, pour la tâche actuelle, à *NEW* et appliquons le même procédé que le niveau *NEW*. Si l'utilisateur refuse les explications, le robot va simplement surveiller l'exécution du nœud courant, sans aller plus profondément dans

l'arbre. En cas de succès, le niveau de connaissance pour la tâche courante est amélioré pour être mis à *INTERMEDIATE*. Ce niveau de connaissance (*BEGINNER*) sera également utilisé comme niveau par défaut. De cette façon, lorsque le niveau de connaissance de l'utilisateur est inconnu sur la tâche, nous demanderons tout simplement s'il a besoin d'explication et le robot adaptera son comportement en fonction de sa réponse.

6.6.2.4 Le collaborateur est *INTERMEDIATE* (lignes 24- 26)

Si l'humain a un niveau *INTERMEDIATE* pour la tâche courante, le robot verbalise la tâche et ne propose pas de l'expliquer, car l'homme a réussi à faire au moins une fois la tâche sans explications. D'autre part, nous n'allons pas plus loin dans l'arbre et surveillons directement la tâche courante. Si l'utilisateur échoue, le robot dégrade son niveau de connaissance à *BEGINNER*, sinon il l'améliore en *EXPERT*.

6.6.2.5 Le collaborateur est *EXPERT* (lignes 24- 26)

Dans le cas du niveau *EXPERT*, sur la tâche courante, nous procédons comme le niveau de connaissance précédent, en dégradant à *INTERMEDIATE* si l'utilisateur fait une erreur et en gardant le niveau *EXPERT* s'il l'accomplit comme prévu.

6.6.3 Exécution et surveillance

Une fois que la tâche actuelle à effectuer a été expliquée, le robot l'effectue si elle lui est allouée, ou il surveille l'accomplissement de la tâche par son partenaire humain. Par conséquent, la surveillance peut être faite à un haut niveau d'abstraction si l'humain a suffisamment de connaissance. Nous avons choisi cette adaptabilité au niveau de la surveillance car nous pensons que le robot sera plus efficace, préservant ses ressources en concentrant son attention plus souvent sur les parties du plan qui n'ont jamais été exécutées par le partenaire humain, et moins souvent lorsqu'il a une certaine forme d'expertise, donnant également plus de liberté à l'homme sur la façon d'effectuer la tâche qu'il connaît (par exemple en changeant l'ordre des sous-tâches).

Surveiller les actions humaines est un problème complexe, particulièrement avec des tâches de haut-niveau, où nous ne suivons pas une suite d'actions atomiques. Le système devrait avoir des modèles de raisonnement pour permettre au robot de comprendre si l'état du monde est cohérent avec l'action que l'humain doit accomplir. Il devrait également être capable de mesurer le niveau d'engagement de

l'humain à la tâche, afin de mieux estimer si l'humain est en train d'accomplir ou non sa partie du plan partagé, et réagir de manière appropriée.

6.6.4 Échec et replanification

Le but de suivre les actions humaines est d'être capable de gérer les comportements imprévus de l'homme et rétablir l'interaction. Lorsque cela arrive, le robot doit informer l'homme de son comportement considéré comme faux et dégrader son niveau de connaissance. La prochaine fois que l'humain réalisera cette tâche, le robot va le guider et surveiller son exécution à un niveau plus détaillé (les tâches filles). Le planificateur de tâche reçoit une requête pour calculer un nouveau plan pour atteindre le but avec le nouvel état du monde. Un des bénéfices de cette mise à jour dynamique des connaissances de l'homme est que ce nouveau plan peut comprendre des tâches que le robot a déjà expliqué ou que l'humain a effectué avant que l'échec ne se produise. Dans ce cas, guider l'humain à travers l'exécution du nouveau plan sera plus rapide car le robot n'aura pas à ré-expliquer ces tâches. Ce comportement de replanification permet de donner de la robustesse au système robotique et permet une procédure de recouvrement socialement acceptable où nous informons l'humain de l'erreur et ré-expliquons le plan seulement avec le niveau de détails nécessaires.

6.7 Implémentation

6.7.1 Architecture

Nous avons implémenté les mécanismes proposés dans notre architecture [2]. La figure 6.4 illustre l'architecture et les interactions entre les modules principaux. Le module de gestion de l'exécution du HTN (*htn execution manager*) contient l'algorithme présenté en section 6.6.1. Il est contrôlé par le superviseur et envoie des requêtes au superviseur pour surveiller ou exécuter une tâche.

Pour nos tests, nous utilisons le PR2 de Willow Garage [3]. Comme la perception n'est pas l'aspect principal de ces travaux, nous utilisons simplement la Motion Capture pour suivre les humains, et un système de reconnaissance de tag pour le suivi d'objets. De même, due à la complexité des tâches de manipulation contenues dans le scénario, l'exécution de ces tâches est simulée pour le robot. Au commencement du scénario, le robot scanne l'environnement, ce qui lui permet de construire un modèle de l'état du monde. Nous avons choisi pour cette première implémentation

2. Open-Source modules : http ://homepages.laas.fr/gmilliez/hri2016/
3. https ://www.willowgarage.com/pages/pr2/overview

une stratégie simple de surveillance. Le robot observe l'environnement, met à jour son modèle d'état du monde en conséquence, et surveille les conséquences attendues des actions. Les actions réalisées sont inférées en utilisant la distance entre le bras de l'homme et un point d'intérêt (par exemple une boîte). Nous considérons une action comme "échouée" si elle n'a pas été réalisée avant une durée prédéfinie.

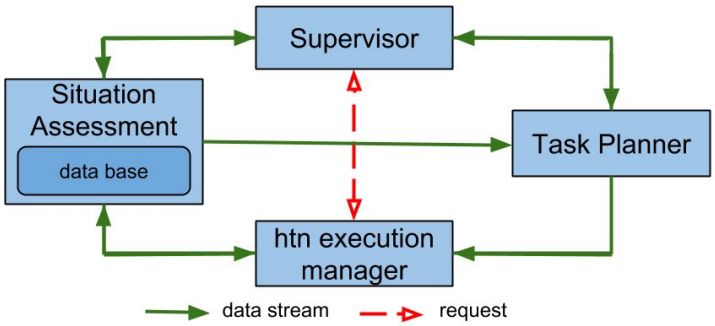


FIGURE 6.4 – Architecture du système permettant d'adapter la collaboration aux connaissances du partenaire humain.

6.7.2 Expérimentation

Pour tester notre système, nous avons choisi un scénario où un humain revient du travail et doit préparer deux desserts qu'il compte apporter à un dîner. Il décide de cuisiner une tourte aux pommes et une tarte à la banane, mais il n'a pas de connaissances préalable sur la façon de les cuisiner. Il demande à son robot domestique des indications et son aide comme illustré par la figure 6.5.

Comme dans ce scénario l'utilisateur est pressé, nous allons utiliser la politique d'efficacité. Nous avons créé un domaine représentant la connaissance nécessaire sur les tâches à utiliser par le planificateur de tâches (*task planner*) et pour le gestionnaire d'exécution d'htn (*htn execution manager*). Pour cuisiner la tourte aux pommes, cinq tâches principales sont nécessaires. Nous imaginons dans ce scénario, qu'à cause de limitations au niveau de l'accessibilité, le robot ne peut faire la pâte ou préparer les fruits. De ce fait, la répartition de tâches sera la suivante.

— L'humain va préparer la pâte, ce qui signifie qu'il va la faire et la mettre dans le moule.
— Le robot va préparer la mixture, ce qui signifie qu'il va la faire et la mettre dans le moule.
— L'humain va alors préparer les fruits, en les coupant et en les mettant dans le moule.
— Puis il va prendre en charge la pâte pour le haut de la tourte.

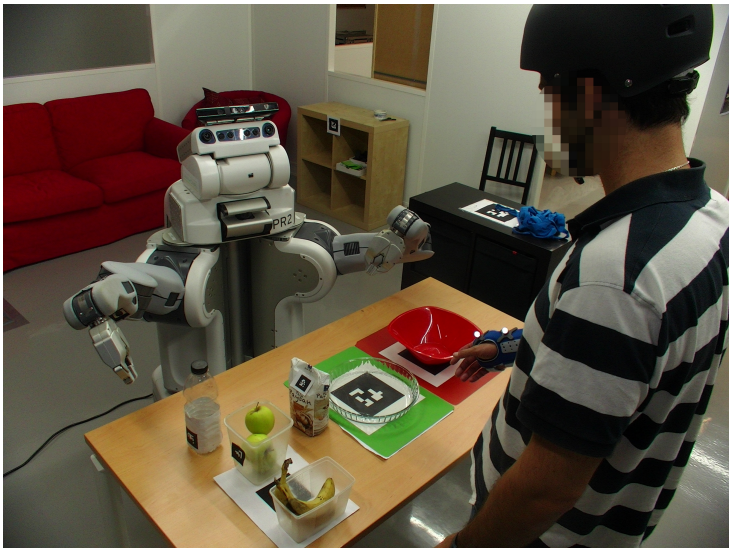


FIGURE 6.5 – Illustration du scénario de préparation de la tarte aux pommes.

— Enfin, le robot s'occupera de la cuisson en mettant la tourte dans le four et en réglant la minuterie.

Après la première étape, l'humain aura acquis la connaissance pour faire la pâte. Par conséquent, pendant l'exécution, lors de l'étape de la seconde pâte (pour le haut de la tourte), le robot demande à l'utilisateur s'il a besoin d'aide. Nous supposons ici qu'il dit "non". Le robot n'explique donc pas cette étape et l'humain atteindra le niveau *INTERMEDIATE* après l'avoir accomplie. Concernant la deuxième tâche humaine, elle sera représentée dans le modèle de connaissance comme <*human1, PrepareFruits, [fruit], VALUE*>. En effet nous considérons que le processus est le même pour n'importe quel fruit (les couper, puis les mettre dans le moule) donc nous utilisons la classe fruit plutôt que l'instance pomme ou banane.

Après avoir cuisiné le premier dessert, à savoir la tourte aux pommes, le robot génère un plan pour cuisiner la tarte à la banane. Cette fois-ci, nous considérons que les deux agents peuvent accomplir toutes les tâches (tout est atteignable pour les deux parties). La tarte à la banane requiert les même tâches avec différents paramètres. La mixture est différente, ainsi que les fruits mais la méthode de préparation de ceux-ci est la même (couper et répartir dans le moule). De plus, la tarte à la banane n'aura pas de pâte sur le dessus et aura un temps de cuisson différent. Le plan produit est présenté dans la figure 6.6. Nous pouvons observer que la politique favorise l'efficacité en donnant les tâches connues à l'utilisateur (*PrepareDough* et *PrepareFruits*). Pendant l'exécution, comme l'utilisateur a un niveau de connaissance à *INTERMEDIATE* sur comment préparer la pâte, le robot ne va pas l'expliquer. Concernant la tâche de préparation de fruits (*prepareFruits*), le robot va

proposer de fournir des explications à l'humain car celui-ci ne l'a accomplie qu'une fois avec des explications.

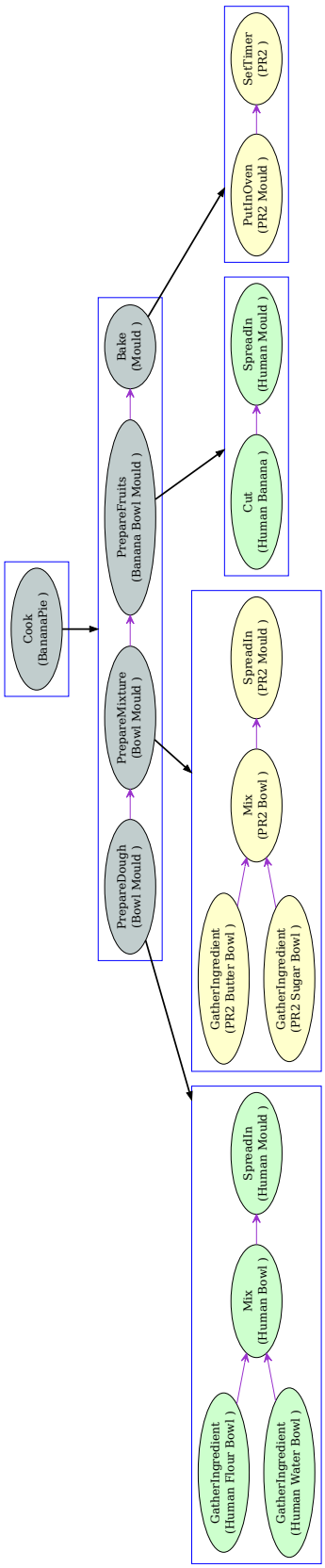


FIGURE 6.6 – L'arbre HTN associé au plan partagé créé pour réaliser la tarte à la banane de façon collaborative.

6.8 Étude utilisateur et discussion

6.8.1 Étude utilisateur

Nous avons conduit une étude comparative afin d'avoir une première évaluation de la façon dont l'adaptabilité du système est perçue par les utilisateurs. Deux groupes d'utilisateurs ont été constitués, auxquels il a été demandé de suivre le scénario des deux desserts. Le premier groupe a interagir avec un robot simulé équipé d'un système basique (BS). BS a le même comportement que notre système, à ceci prêt qu'il n'a pas le mécanisme permettant de connaître le niveau de connaissance de l'homme. Le deuxième groupe a interagi avec un second système, appelé système de connaissance (KS pour Knowledge System), exhibant un comportement similaire à celui fourni par notre système. Tous les participants ont par la suite évalué l'interaction sur plusieurs critères. La répartition de tâche présentée dans la section 6.7.2 est utilisée dans les deux systèmes pour accomplir l'exemple de la tourte aux pommes. Une fois qu'ils ont cuisiné le premier dessert, le robot génère un plan pour cuisiner la tarte aux pommes. Dans KS, le robot a le même comportement que notre système et favorise une répartition de tâches pour la tarte à la banane où l'humain doit accomplir les tâches qu'il a déjà accomplies lors de la préparation de la tourte aux pommes (préparant la pâte et les fruits). Dans BS, nous imaginons que le robot pourrait demander à l'humain de faire la mixture plutôt que la pâte.

Deux groupes composés chacun de 19 participants, de 18 à 60 ans, ont été constitués. Chaque groupe a interagi avec un des systèmes. Cela a été réalisé grâce à une étude utilisateur en ligne [4], où nous avons présenté des images de l'état de la tâche et des enregistrements des paroles du robot en Français pour chaque étape de l'interaction (comme présenté dans la figure 6.7). Pour certaines étapes, l'utilisateur pouvait choisir l'action à réaliser, rendant possible d'effectuer une action fausse conduisant à une replanification du robot. Par souci de simplicité, la replanification était limitée à une annulation de l'action erronée avant de reprendre le plan précédent. De plus, pour se concentrer sur l'adaptabilité de notre solution grâce à la prise en compte des connaissances de l'homme dans les différentes étapes du plan partagé, nous n'avons pas autorisé la négociation avec l'utilisateur et le robot a directement imposé son plan. À la fin de l'interaction simulée, nous avons posé les mêmes questions aux deux groupes, concernant l'adaptabilité du système et le partenaire robotique. Les utilisateurs ont donné des notes en suivant une échelle de Likert de un (pas d'accord) à 5 (d'accord) afin d'exprimer leur ressenti sur plusieurs affirmations (comme indiqué dans la figure 6.7). Par exemple, une des questions était "Avez-vous eu l'im-

4. Étude utilisateur KS http ://goo.gl/forms/qvbtu4vcFW, et BS http ://-goo.gl/forms/ZSvGcCi5le

pression que le robot adaptait ses explications au niveau de connaissance que vous avez acquis au cours de l'interaction ?".

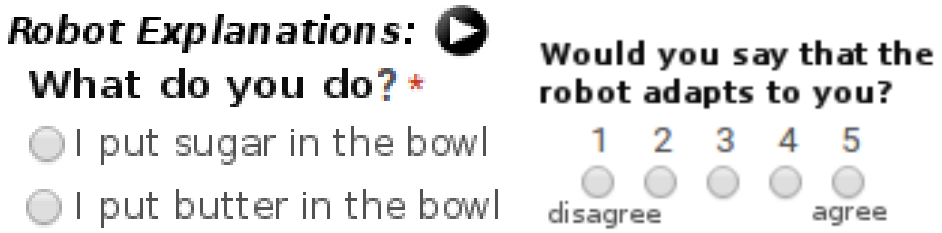


FIGURE 6.7 – L'utilisateur écoute un enregistrement de la voix du robot et choisit l'action à faire (image de *Gauche*). À la fin, l'utilisateur évalue l'interaction en utilisant une échelle de Likert (*image de droite*).

6.8.2 Résultats

Nous avons recueilli les résultats pour chaque groupe d'utilisateurs, et calculé la moyenne ainsi que l'écart type et la p-valeur pour évaluer la fiabilité. La p-valeur a été calculée en utilisant une t-distribution avec 18 degrés de liberté et en considérant que la moyenne des notes pour KS serait plus haute que pour BS comme énoncé de notre hypothèse et que la moyenne des notes pour les deux groupes serait identique comme hypothèse nulle. La figure 6.8 résume les résultats obtenus en affichant la moyenne des notes, l'écart type sous forme de barres d'erreurs ainsi que la p-valeur. En comparant les réponses des utilisateurs, nous pouvons voir que l'adaptation du robot au niveau de connaissance de l'homme concernant les explications fournies est bien perçue avec une moyenne de 3.74 pour KS contre 2.05 pour BS. Les utilisateurs interagissant avec KS ont globalement remarqué que la répartition des tâches prenait en compte leur connaissance en donnant une notation moyenne de 3.42 pour KS et 2.58 pour BS. La dernière question concerne la liberté de choisir la manière d'accomplir une tâche, liée au suivi de plus haut niveau en cas de connaissance suffisante de la tâche. La moyenne est de 2.58 pour KS contre 1.89 pour BS, et la p-valeur est inférieur à 0.05. On peut donc conclure que même sur cet aspect l'adaptabilité a été perçue. Avec le système KS, les utilisateurs ont attribué une moyenne de 3.11 pour l'adaptabilité globale du système contre 1.89 pour le système basique. Nous avons également demandé comment le partenaire robotique était perçu. Bien que en BS le robot ne soit pas perçu comme plus verbeux (2.53 pour KS contre 2.47 pour BS), il semble que les utilisateurs ont trouvé l'interaction légèrement plus naturelle (2.74 contre 2.42) et le robot est apparu comme plus intelligent (2.79 contre 2.26). Même si ces résultats renforcent notre idée, comme la p-valeur est supérieure à 0.1 pour l'aspect naturel, cela ne permet pas de prouver

qu'il y a une différence notable entre les deux systèmes (cela ne permet pas de rejeter l'hypothèse nulle). Nous pensons que d'autres paramètres ont été pris en compte par les utilisateurs, comme la parole elle-même, ce qui a conduit les utilisateurs des deux expériences à évaluer l'interaction comme étant peu naturelle. Améliorer la verbalisation avec l'usage d'un dictionnaire de synonymes pourrait aider à avoir des résultats plus significatifs.

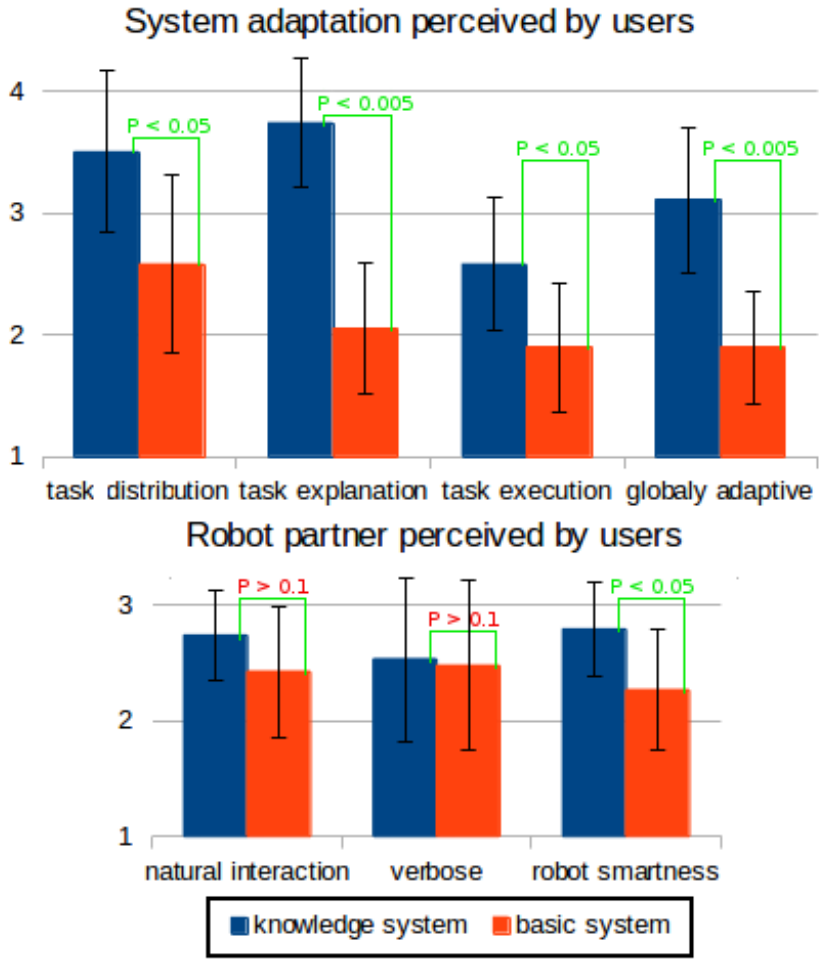


FIGURE 6.8 – Notes moyennes des utilisateurs sur plusieurs critères concernant la perception de l'adaptation du système par rapport à leur connaissance (en haut) et concernant le partenaire robotique (en bas). Les résultats en bleu représentent l'évaluation de notre système (KS) et en rouge un système basique (BS.

Cette étude permet de montrer comment les utilisateurs ont pu percevoir l'adaptation du robot à leur niveau de connaissance concernant la répartition de tâches, les explications fournies, et la surveillance de leur actions. De plus, le robot est apparu comme plus intelligent. Cependant ces premiers résultats doivent être confirmés par une étude sur une population plus grande. De même, comme travaux futurs nous

envisageons de conduire une étude utilisateur avec un vrai robot car nous pensons que cela permettra d'avoir des avis plus réalistes de la part des utilisateurs. Dans les deux études, nous avons demandé aux participants comment l'interaction pourrait être améliorée selon eux. Plusieurs utilisateurs ont suggéré qu'ils aimeraient choisir quelles actions faire ou non, mettant en valeur l'importance de la négociation (nous avions retiré la négociation de l'étude utilisateur pour nous concentrer sur l'adaptation au niveau des connaissances). L'un des utilisateurs a suggéré qu'il aimerait pouvoir être informé de l'évolution du plan de temps en temps. Cela peut facilement être ajouté car à chaque moment le robot connaît le nombre de nœuds qu'il reste dans le plan partagé pour accomplir le but. D'autres utilisateurs ont fait des suggestions concernant la parole du robot, la voix, l'intonation et le choix des mots. Ces aspects ne sont pas l'objet de l'étude, cependant ces commentaires montrent leur importance pour l'interaction avec l'homme.

6.9 Conclusion

Dans ce chapitre nous avons présenté une solution pour permettre à un robot de gérer une activité de collaboration homme-robot, et ce de la génération de plan à l'exécution, en guidant son partenaire humain de manière efficace et socialement acceptable. Notre approche est basée sur un suivi dynamique et une mise à jour en ligne des connaissances du partenaire humain. Nous avons également mis en place deux politiques d'interaction, l'apprentissage ou l'efficacité, qui procurent différents niveaux d'interactions. Notre méthode incorpore le suivi des actions humaines de haut niveau, en se focalisant sur la complétion de la tâche plutôt que sur la façon dont la tâche est accomplie. Cela permet de donner une certaine flexibilité à l'humain lorsqu'il réalise les tâches qui lui sont assignées.

Nous avons mené une étude comparative en ligne. Les résultats sont encourageants, les utilisateurs ont été capables de percevoir l'adaptabilité du robot sur les aspects de génération, d'accompagnement et de surveillance de tâches. À l'heure actuelle, cette approche nécessite que le robot connaisse toutes les méthodes à réaliser (incluant la décomposition de chaque tâche). Ajouter la possibilité d'enseigner une décomposition de tâches au robot comme dans [Mohseni-Kabir 2015] permettrait de surmonter cette restriction. Pour la partie de négociation, nous avons fourni un premier mécanisme, en donnant un coût différent à une tâche lorsque l'utilisateur exprime son souhait de la faire ou non. Cependant, cela pourrait être amélioré en utilisant des travaux précédents tel que [Chu-Carroll 2000] qui expose une infrastructure de Proposition/Évaluation/Modification pour prendre en charge la négociation. Il serait par exemple intéressant de prendre en compte, une fois de plus,

la connaissance de l'utilisateur pour accepter ou refuser une proposition de modification. Le robot pourrait ainsi informer l'utilisateur de la différence de coût pour effectuer une tâche entre l'homme et le robot (une tâche pourrait être plus ou moins simple ou dangereuse selon qu'elle est effectuée par un agent humain ou robotique). En ce qui concerne la surveillance de tâches humaines, l'implémentation actuelle utilise une stratégie simple basée sur l'estimation du résultat d'une tâche après un certain temps. Nous souhaiterions améliorer cela en concevant des mécanismes plus élaborés pour estimer l'engagement de l'homme dans une tâche et mieux reconnaître lorsqu'une tâche surveillée a échoué. L'humain pourrait aussi informer le robot de la raison pour laquelle il n'effectue pas la tâche attendue. Enfin, pour améliorer l'interactivité, il serait pertinent d'autoriser l'humain à demander directement des explications sur les actions du robot comme dans [Lomas 2012].

Chapitre 7

Conclusion et Perspectives

7.1 Conclusion

Ce manuscrit de thèse rapporte à travers cinq chapitres comment il est possible d'acquérir des données contextuelles et de les utiliser comme base de raisonnement afin d'obtenir une estimation de la situation à différents niveaux d'abstraction. Ces données contextuelles combinées avec des processus de raisonnements fournissent les indices nécessaires pour estimer également la situation de l'homme tant au niveau de sa situation spatiale que sa situation mentale. Ces raisonnements sur le contexte et sur la situation de l'homme sont essentiels pour donner des comportements appropriés au robot durant l'interaction homme-robot sur différentes aspects de l'interaction tel que le dialogue situé, l'assistance proactive, la génération de plan, l'exécution de tâches collaboratives...

Dans le chapitre 2 nous avons montré comment il est possible de mettre en place une architecture modulaire basée sur l'agrégation de données capteurs et sur certains raisonnements pour acquérir et maintenir un état du monde tridimensionnel correspondant à la représentation du monde réel tel qu'il est perçu et inféré (grâce à une gestion d'hypothèses) par le robot. Basé sur cette représentation tridimensionnelle, nous montrons comment les différents modules de l'architecture permettent de générer des *faits* constituant une représentation symbolique du monde. Ces faits sont centralisés par une base de donnée temporelle qui possède une table permettant de garder en mémoire les faits passés et les transitions survenues. Nous présentons également dans ce premier chapitre TOASTER, une infrastructure logicielle modulaire Open-Source implémentant les principes et modèles énoncés. Deux exemples d'expérimentations utilisant l'infrastructure logicielle TOASTER comme module d'estimation de la situation ont été présentés comme exemple d'utilisation possible.

Dans le chapitre 3, nous avons montré et expliqué brièvement les concepts de prise de perspective perceptuelle et conceptuelle. Nous montrons également que ces capacités sont importantes dans les interactions sociales humaines. Nous présentons ensuite comment, à partir de calculs géométriques il nous est possible de doter le système robotique de prise de perspective perceptuelle afin que le robot puisse savoir ce qui est perceptible ou atteignable par l'homme. Basé sur cette prise de perspective

perceptuelle et sur un raisonnement permettant de gérer la mise à jour des croyances des agents, nous avons montré qu'il est possible de maintenir un état de croyance distinct pour chaque agent présent dans la scène. Cette gestion des croyances permet de doter le robot de la capacité de prise de perspective conceptuelle. Pour montrer la capacité de notre robot à se mettre à la place d'un autre agent et de raisonner sur ses croyances, nous avons fait passer deux tests à notre robot. Le premier étant le test connu dans la littérature de la philosophie du développement sous le nom de test de Sally et Anne. Le second est un scénario d'interaction où deux hommes manipulent des objets en présence du robot.

Le chapitre 4 décrit comment les données contextuelles et la capacité de prise de perspective permettent de mettre en place un dialogue situé de qualité. Nous avons montré comment les données contextuelles de différents niveaux d'abstraction (position, distance, représentation symbolique, prise de perspective) sont utilisées tout au long du processus de dialogue (compréhension de la parole, interprétation des gestes de l'homme, identification du référent, choix de la politique par la couche décisionnelle, choix de la modalité de sortie et modulation de la voix, exploration et exécution de la tâche). Nous présentons notamment une étude menée sur simulateur qui a permis de montrer l'utilité de la prise de perspective conceptuelle pour rendre le dialogue plus efficace et plus précis. Nous présentons également une expérimentation menée sur plateforme robotique qui illustre diverses stratégies prenant en compte le contexte et l'état mental de l'homme pour adapter l'exécution de la tâche.

Le chapitre 5 présente comment, en utilisant le contexte et l'état mental de l'homme, il est possible d'interpréter les actions de l'homme afin d'en déduire son intention. Nous avons montré comment cette information permet au robot d'agir de façon proactive pour prévenir l'homme dans le cas où celui-ci effectue une action non optimale par rapport à son objectif à cause de croyances erronées. Il est également possible pour le robot d'aider l'homme à accomplir son but sans que celui-ci ait à exprimer explicitement un besoin ou une requête. Cela donne au robot un aspect proactif. Nous avons également présenté une étude en ligne où les performances de notre système ont été comparées aux performances d'humains pour reconnaître l'intention d'un tiers. Cette étude a permis de montrer que dans la plupart des situations, notre système avait une estimation comparable à l'homme.

Enfin, dans le chapitre 6, nous avons présenté nos travaux pour l'enrichissement du modèle de l'homme en terme de ce qu'il peut, veut et sait faire et adapter le comportement du robot à l'expertise humaine concernant les diverses tâches contenues dans un plan collaboratif. Nous avons mis en place une modélisation de l'état de connaissance de l'homme concernant les tâches qui peuvent être présentes dans

un plan collaboratif. En utilisant cette modélisation, le robot est capable d'adapter la génération de plan à l'expertise humaine. Lors de l'exécution du plan, le robot est également capable d'adapter le niveau d'explication donné à l'homme en s'appuyant sur la structure hiérarchique du plan pour décrire plus ou moins en détails les tâches à accomplir. De même, cette structure hiérarchique et le niveau de connaissance de l'homme sont également utilisés pour adapter le niveau de surveillance du robot sur les tâches accomplies par l'homme. Nous avons effectué une étude utilisateur en ligne afin de comparer l'adaptabilité de notre système aux connaissances de l'homme avec un système standard. Les résultats ont permis de montrer que l'adaptation du système était bien perçue par les utilisateurs.

7.2 Améliorations et travaux à venir

Les travaux présentés dans cette thèse peuvent être étendus sur plusieurs aspects. Tout d'abord, l'infrastructure logicielle présentée implémentant les concepts présentés dans les chapitres 2 et 3 a été créée de façon générique et modulaire et est Open-Source. Le but étant de la rendre utilisable par d'autres équipes de recherche. Pour rendre l'infrastructure facilement utilisable, une documentation est en cours de réalisation. Celle-ci contiendra des tutoriels pour permettre la prise en main rapide de TOASTER. L'infrastructure est toujours en cours de développement pour améliorer la généricité, étendre le nombre de capteurs gérés, améliorer les calculs de faits...

De même, la modélisation de la prise de perspective pourrait être améliorée en l'adaptant à l'utilisateur. Ainsi certains utilisateurs peuvent avoir une déficience au niveau de l'un des sens. Pour effectuer une prise de perspective perceptuelle convenable, le robot devrait pouvoir prendre en compte ce fait. De même, selon l'individu (par exemple si le robot interagit avec des enfants) le robot devrait adapter son comportement. Cela nécessite de pouvoir modéliser non seulement l'état mental des agents mais également d'adapter la mise à jour de cet état mental en fonction des capacités de cet agent à émettre des hypothèses sur l'environnement.

Concernant les aspects temporels, un premier développement a permis de mettre en place la gestion des transitions et de la mémoire de chaque agent. À l'heure actuelle cette capacité n'a pas encore été pleinement exploitée et il serait intéressant de baser certains raisonnements pour améliorer le maintien d'état du monde ou pour pouvoir répondre aux demandes de l'homme s'interrogeant sur les événements ayant eu lieu pendant son absence (quel événement est-il intéressant de rapporter ?) ou utiliser la mémoire pour l'apprentissage du comportement humain. De même, pour permettre une interaction de longue durée, il serait intéressant de mettre en place

des mécanismes permettant au robot d'oublier certains faits considérés comme peu importants.

Pour ce qui est de la gestion de plan en fonction de l'expertise humaine, une étude plus approfondie sur la gestion de la négociation, par exemple en utilisant l'état de connaissance de l'homme, le coût relatif d'une tâche selon l'agent ou l'insistance de celui-ci.

Bibliographie

[Abowd 1999] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith et Pete Steggles. *Towards a better understanding of context and context-awareness.* In Handheld and ubiquitous computing, pages 304–307. Springer, 1999. (Cité en page 125.)

[Alami 1998a] R. Alami, R. Chatila, S. Fleury, M. Ghallab et F. Ingrand. *An architecture for autonomy.* International Journal of Robotic Research, 1998. (Cité en page 23.)

[Alami 1998b] Rachid Alami, Raja Chatila, Sara Fleury, Malik Ghallab et Félix Ingrand. *An architecture for autonomy.* The International Journal of Robotics Research, vol. 17, no. 4, pages 315–337, 1998. (Cité en page 9.)

[Alami 2011] Rachid Alami, Mathieu Warnier, Julien Guitton, Severin Lemaignan et Emrah Akin Sisbot. *When the robot considers the human...* In Proceedings of the 15th international symposium on robotics research, 2011. (Cité en page 9.)

[Alami 2013] Rachid Alami. *On human models for collaborative robots.* In Collaboration Technologies and Systems (CTS), 2013 International Conference on, pages 191–194. IEEE, 2013. (Cité en page 9.)

[Astrinaki 2012] Maria Astrinaki, Nicolas d'Alessandro, Benjamin Picart, Thomas Drugman et Thierry Dutoit. *Reactive and continuous control of HMM-based speech synthesis.* In Spoken Language Technology Workshop (SLT), 2012 IEEE, pages 252–257. IEEE, 2012. (Cité en page 97.)

[Baker 2009] Chris L Baker, Rebecca Saxe et Joshua B Tenenbaum. *Action understanding as inverse planning.* Cognition, vol. 113, no. 3, pages 329–349, 2009. (Cité en page 122.)

[Baker 2014] Chris L Baker et Joshua B Tenenbaum. *Modeling human plan recognition using Bayesian theory of mind.* Plan, activity, and intent recognition : Theory and practice, pages 177–204, 2014. (Cité en pages 121 et 132.)

[Baron-Cohen S 1985] Frith U Baron-Cohen S Leslie AM. *Does the autistic child have a 'theory of mind' ?* Cognition, vol. 21, no. 1, pages 37 – 46, 1985. (Cité en page 57.)

[Baron 1980] S Baron, R Muralidharan, R Lancraft et G Zacharias. *PROCRU : A model for analyzing crew procedures in approach to landing.* 1980. (Cité en page 19.)

[Beck 2011] Anders Billesø Beck, Claus Risager, Nils A Andersen et Ole Ravn. *Spacio-temporal situation assessment for mobile robots.* In Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on, pages 1–8. IEEE, 2011. (Cité en page 23.)

[Bedny 1999] Gregory Bedny et David Meister. *Theory of activity and situation awareness.* International Journal of cognitive ergonomics, vol. 3, no. 1, pages 63–72, 1999. (Cité en page 17.)

[Beetz 2015] Michael Beetz, Moritz Tenorth et Jan Winkler. *Open-EASE.* In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 1983–1990. IEEE, 2015. (Cité en page 44.)

[Besl 1985] Paul J. Besl et Ramesh C. Jain. *Three-dimensional Object Recognition.* ACM Comput. Surv., vol. 17, no. 1, pages 75–145, Mars 1985. (Cité en page 27.)

[Bladon 2002] Peter Bladon, Richard J Hall et W Andy Wright. *Situation assessment using graphical models.* In Information Fusion, 2002. Proceedings of the Fifth International Conference on, volume 2, pages 886–893. IEEE, 2002. (Cité en page 20.)

[Blakemore 2001] Sarah-Jayne Blakemore et Jean Decety. *From the perception of action to the understanding of intention.* Nature Reviews Neuroscience 2, 561-567 (August 2001, vol. 2, pages 561 – 567, 2001. (Cité en page 125.)

[Bonasso 1995] R Peter Bonasso, David Kortenkamp, David P Miller et Marc Slack. *Experiences with an architecture for intelligent, reactive agents.* In International Workshop on Agent Theories, Architectures, and Languages, pages 187–202. Springer, 1995. (Cité en page 22.)

[Bratman 1984] Michael Bratman. *Two Faces of Intention.* Philosophical Review, vol. 93, pages 375 – 405, 1984. (Cité en page 120.)

[Bratman 2013] Michael E Bratman. Shared agency : A planning theory of acting together. Oxford University Press, 2013. (Cité en page 144.)

[Breazeal 2006] C. Breazeal, M. Berlin, A. Brooks, J. Gray et A. Thomaz. *Using Perspective Taking to Learn from Ambiguous Demonstrations.* Robotics and Autonomous Systems, 2006. (Cité en page 58.)

[Breazeal 2009] Cynthia Breazeal, Jesse Gray et Matt Berlin. *An Embodied Cognition Approach to Mindreading Skills for Socially Intelligent Robots.* I. J. Robotic Res., 2009. (Cité en pages 58, 121 et 122.)

[Brewka 1993] Gerhard Brewka et Joachim Hertzberg. *How to do things with worlds : On formalizing actions and plans.* Journal of Logic and Computation, vol. 3, no. 5, pages 517–532, 1993. (Cité en page 42.)

[Brooks 1986] Rodney Brooks. *A robust layered control system for a mobile robot.* IEEE journal on robotics and automation, vol. 2, no. 1, pages 14–23, 1986. (Cité en page 22.)

[Bruner 1981] Jerome S. Bruner. *Intention in the structure of action and interaction.* Advances in Infancy Research, vol. 1, pages 41 – 56, 1981. (Cité en page 120.)

[Brusilovskiy 1994] PL Brusilovskiy. *The construction and application of student models in intelligent tutoring systems.* Journal of computer and systems sciences international, vol. 32, no. 1, pages 70–89, 1994. (Cité en page 144.)

[Brusilovsky 2005] P. Brusilovsky, S. Sosnovsky et O. Shcherbinina. *User Modeling in a Distributed E-Learning Architecture.* In User Modeling 2005, volume 3538 of *Lecture Notes in Computer Science*, pages 387–391. Springer Berlin Heidelberg, 2005. (Cité en page 144.)

[Bui 2003] Hung Hai Bui. *A general model for online probabilistic plan recognition.* In IJCAI, volume 3, pages 1309–1315. Citeseer, 2003. (Cité en page 121.)

[Butterfill 2011] S. A. Butterfill et N. Sebanz. *Editorial : Joint Action : What Is Shared ?* Review of Philosophy and Psychology, vol. 2, no. 2, pages 137–146, 2011. (Cité en page 143.)

[Byom 2013] Lindsey J Byom et Bilge Mutlu. *Theory of mind : mechanisms, methods, and new directions.* Frontiers in human neuroscience, vol. 7, 2013. (Cité en page 121.)

[Byron 2006] Donna K. Byron et Eric Fosler-Lussier. *The OSU Quake 2004 corpus of two-party situated problem-solving dialogs.* In Proceedings of the 15th Language Resources and Evaluation Conference (LREC'06), 2006. (Cité en page 99.)

[Call J 1998] Tomasello M Call J. *Distinguishing intentional from accidental actions in orangutans (Pongo pygmaeus), chimpanzees (Pan troglodytes) and human children (Homo sapiens).* Journal of Comparative Psychology, vol. 112, no. 2, pages 192 – 206, 1998. (Cité en page 121.)

[Chu-Carroll 2000] J. Chu-Carroll et S. Carberry. *Conflict resolution in collaborative planning dialogs.* International Journal of Human-Computer Studies, vol. 53, no. 6, pages 969–1015, 2000. (Cité en page 167.)

[Coradeschi 2013] Silvia Coradeschi, Amy Loutfi et Britta Wrede. *A Short Review of Symbol Grounding in Robotic and Intelligent Systems.* KI - Kunstliche Intelligenz, vol. 27, no. 2, 2013. (Cité en page 24.)

[Daoutis 2009] M. Daoutis, S Coradeshi et A Loutfi. *Grounding commonsense knowledge in intelligent systems.* Journal of Ambient Intelligence and Smart Environments, 2009. (Cité en page 24.)

[Das 2002] Subrata Das, Rachel Grey et Paul Gonsalves. *Situation assessment via Bayesian belief networks.* In Information Fusion, 2002. Proceedings of the Fifth International Conference on, volume 1, pages 664–671. IEEE, 2002. (Cité en page 20.)

[Daubigney 2012] L. Daubigney, M. Geist, S. Chandramohan et O. Pietquin. *A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimization.* Journal on Selected Topics in Signal Processing, vol. 6, no. 8, pages 891–902, 2012. (Cité en page 93.)

[Demiris 2007] Yiannis Demiris. *Prediction of intent in robotics and multi-agent systems.* Cognitive processing, vol. 8, no. 3, pages 151–158, 2007. (Cité en page 121.)

[Dennett 1989] Daniel Clement Dennett. The intentional stance. MIT press, 1989. (Cité en page 125.)

[Devin 2016] Sandra Devin et Rachid Alami. *An Implemented Theory of Mind to Improve Human-Robot Shared Plans Execution.* In ACM/IEEE International Conference on Human-Robot Interaction, HRI'16,, 2016. (Cité en page 9.)

[Diankov 2010] Rosen Diankov. *Automated Construction of Robotic Manipulation Programs.* PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010. (Cité en page 99.)

[Doshi 2009] Prashant Doshi et Piotr J Gmytrasiewicz. *Monte Carlo sampling methods for approximating interactive POMDPs.* Journal of Artificial Intelligence Research, pages 297–337, 2009. (Cité en page 122.)

[Echeverria 2011] Gilberto Echeverria, Nicolas Lassabe, Arnaud Degroote et Séverin Lemaignan. *Modular open robots simulation engine : MORSE.* In ICRA, 2011. (Cité en page 47.)

[Echeverria 2012] Gilberto Echeverria, Séverin Lemaignan, Arnaud Degroote, Simon Lacroix, Michael Karg, Pierrick Koch, Charles Lesire et Serge Stinckwich. *Simulating Complex Robotic Scenarios with MORSE.* In SIMPAR, pages 197–208, 2012. (Cité en page 99.)

[Edlund 2006] Johan Edlund, Magnus Grönkvist, Andreas Lingvall et Egils Sviestins. *Rule-based situation assessment for sea surveillance.* In Defense and Security Symposium, pages 624203–624203. International Society for Optics and Photonics, 2006. (Cité en page 19.)

[Endsley 1995] Mica R Endsley. *Toward a theory of situation awareness in dynamic systems.* Human Factors : The Journal of the Human Factors and Ergonomics Society, vol. 37, no. 1, pages 32–64, 1995. (Cité en pages 17 et 18.)

[Endsley 2000] Mica R Endsley et DJ Garland. *Theoretical underpinnings of situation awareness : A critical review.* Situation awareness analysis and measurement, pages 3–32, 2000. (Cité en page 16.)

[Ferreira 2013a] Emmanuel Ferreira et Fabrice Lefèvre. *Expert-based reward shaping and exploration scheme for boosting policy learning of dialogue management.* In Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on, pages 108–113. IEEE, 2013. (Cité en pages 89 et 93.)

[Ferreira 2013b] Emmanuel Ferreira et Fabrice Lefèvre. *Social signal and user adaptation in reinforcement learning-based dialogue management.* In Proceedings of the 2nd Workshop on Machine Learning for Interactive Systems : Bridging the Gap Between Perception, Action and Communication, pages 61–69. ACM, 2013. (Cité en pages 89 et 93.)

[Ferreira 2015a] Emmanuel Ferreira. *Apprentissage automatique en ligne pour un dialogue homme-machine situé.* Avignon, 2015. (Cité en pages 88, 89 et 96.)

[Ferreira 2015b] Emmanuel Ferreira, Grégoire Milliez, Fabrice Lefevre et Rachid Alami. *Users' Belief Awareness in Reinforcement Learning-based Situated Human-Robot Dialogue Management.* In IWSDS, 2015. (Cité en pages 12 et 105.)

[Fiore 2014] Michelangelo Fiore, Aurélie Clodic et Rachid Alami. *On Planning and Task achievement Modalities for Human-Robot Collaboration.* In The 2014 International Symposium on Experimental Robotics, 2014. (Cité en pages 9, 23, 123 et 130.)

[Fiore 2015] Michelangelo Fiore, Harmish Khambhaita, Grégoire Milliez et Rachid Alami. Social robotics : 7th international conference, icsr 2015, paris, france, october 26-30, 2015, proceedings, chapitre An Adaptive and Proactive Human-Aware Robot Guide, pages 194–203. Springer International Publishing, Cham, 2015. (Cité en page 50.)

[Flavell 1977] John H Flavell. *The development of knowledge about visual perception.* In Nebraska symposium on motivation. University of Nebraska Press, 1977. (Cité en page 57.)

[Flavell 1992] J. Flavell. Perspectives on perspective taking, pages 107–139. L. Erlbaum Associates, 1992. (Cité en page 56.)

[Fracker 1988] Martin L Fracker. *A theory of situation assessment : Implications for measuring situation awareness.* In Proceedings of the Human Factors and Ergonomics Society Annual Meeting, volume 32, pages 102–106. SAGE Publications, 1988. (Cité en page 16.)

[Freese 2010] Marc Freese, Surya Singh, Fumio Ozaki et Nobuto Matsuhira. *Virtual robot experimentation platform V-REP : a versatile 3D robot simulator.* In Proceedings of the Second international conference on Simulation, modeling, and programming for autonomous robots, SIMPAR'10, pages 51–62, Berlin, Heidelberg, 2010. Springer-Verlag. (Cité en page 99.)

[Frick 2014] Andrea Frick, Wenke Möhring et Nora S Newcombe. *Picturing perspectives : development of perspective-taking abilities in 4-to 8-year-olds.* Frontiers in psychology, vol. 5, 2014. (Cité en page 56.)

[Gharbi 2015] Mamoun Gharbi, Raphael Lallement et Rachid Alami. *Combining symbolic and geometric planning to synthesize human-aware plans : toward more efficient combined search.* In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 6360–6365. IEEE, 2015. (Cité en page 62.)

[Gilson 1995] R. D. Gilson. *Situation Awareness.* Human Factors, vol. 37, pages 3–4, 1995. (Cité en page 16.)

[Gilson 1998] Mark Gilson. *A Brief History of Japanese Robophilia.* Leonardo, vol. 31, no. 5, pages 367–369, 1998. (Cité en page 6.)

[Ginsberg 1988a] Matthew L Ginsberg et David E Smith. *Reasoning about action I : A possible worlds approach.* Artificial intelligence, vol. 35, no. 2, pages 165–195, 1988. (Cité en page 42.)

[Ginsberg 1988b] Matthew L Ginsberg et David E Smith. *Reasoning about action II : the qualification problem.* Artificial Intelligence, vol. 35, no. 3, pages 311–342, 1988. (Cité en page 42.)

[Gmytrasiewicz 2004] Piotr J Gmytrasiewicz et Prashant Doshi. *Interactive pomdps : Properties and preliminary results.* In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3, pages 1374–1375. IEEE Computer Society, 2004. (Cité en page 122.)

[Grosz 1996] Barbara J Grosz et Sarit Kraus. *Collaborative plans for complex group action.* Artificial Intelligence, vol. 86, no. 2, pages 269–357, 1996. (Cité en page 144.)

[Grosz 1999] Barbara J Grosz et Sarit Kraus. *The evolution of SharedPlans.* In Foundations of rational agency, pages 227–262. Springer, 1999. (Cité en page 144.)

[Guitton 2012] J. Guitton, M. Warnier et R. Alami. *Belief Management for HRI Planning.* In BNC@ECAI, 2012. (Cité en page 23.)

[Han 2013] Jungong Han, Ling Shao, Dong Xu et Jamie Shotton. *Enhanced computer vision with microsoft kinect sensor : A review.* IEEE transactions on cybernetics, vol. 43, no. 5, pages 1318–1334, 2013. (Cité en page 27.)

[Harnad 1990] Stevan Harnad. *The symbol grounding problem.* Physica D : Nonlinear Phenomena, vol. 42, no. 1-3, pages 335–346, 1990. (Cité en page 24.)

[Heintz 2008] Fredrik Heintz, Jonas Kvarnström et Patrick Doherty. *Knowledge processing middleware.* In International Conference on Simulation, Modeling, and Programming for Autonomous Robots, pages 147–158. Springer, 2008. (Cité en page 24.)

[Heintz 2009] Fredrik Heintz. *DyKnow : A stream-based knowledge processing middleware framework.* Linköping University Electronic Press, 2009. (Cité en page 24.)

[Heintz 2010] Fredrik Heintz, Jonas Kvarnström et Patrick Doherty. *Bridging the sense-reasoning gap : DyKnow–stream-based middleware for knowledge processing.* Advanced Engineering Informatics, vol. 24, no. 1, pages 14–26, 2010. (Cité en page 24.)

[Higgins 2005] Robert P Higgins. *Automatic event recognition for enhanced situational awareness in uav video.* In MILCOM 2005-2005 IEEE Military Communications Conference, pages 1706–1711. IEEE, 2005. (Cité en pages 20 et 21.)

[Hoang 2013] Trong Nghia Hoang et Kian Hsiang Low. *Interactive POMDP Lite : Towards practical planning to predict and exploit intentions for interacting with self-interested agents.* In Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, pages 2298–2305. AAAI Press, 2013. (Cité en page 122.)

[Holzapfel 2004] Hartwig Holzapfel, Kai Nickel et Rainer Stiefelhagen. *Implementation and Evaluation of a Constraint-based Multimodal Fusion System for Speech and 3D Pointing Gestures.* In ICMI, 2004. (Cité en page 90.)

[Howard 2005] Catherine Howard et Markus Stumptner. *Situation assessments using object oriented probabilistic relational models.* In 2005 7th International Conference on Information Fusion, volume 2, pages 8–pp. IEEE, 2005. (Cité en page 21.)

[Jackson 1990] Peter Jackson et John Pais. *Semantic accounts of belief revision.* In Truth Maintenance Workshop, pages 155–177. Springer, 1990. (Cité en page 42.)

[Johnson 2005] M. Johnson et Y. Demiris. *Perceptual Perspective Taking and Action Recognition.* Advanced Robotic Systems, 2005. (Cité en page 58.)

[Karpas 2015] Erez Karpas, Steven J Levine, Peng Yu et Brian C Williams. *Robust Execution of Plans for Human-Robot Teams.* In Twenty-Fifth International Conference on Automated Planning and Scheduling, 2015. (Cité en pages 122 et 145.)

[Kluge 2001] Boris Kluge, Jörg Illmann et Erwin Prassler. *Situation Assessment In Crowded Public Environments.* In Proc. of Int. Conf. on Field and Service Robotics, 2001. (Cité en page 23.)

[Kollar 2010] T. Kollar, S. Tellex, D. Roy et N. Roy. *Toward understanding natural language directions.* In HRI, 2010. (Cité en page 24.)

[Koppula 2013] Hema S Koppula et Ashutosh Saxena. *Anticipating human activities using object affordances for reactive robotic response.* Robotics : Science and Systems, Berlin, 2013. (Cité en page 121.)

[Lallee 2013] S. Lallee*et al. Cooperative human robot interaction systems : IV. Communication of shared plans with humans using gaze and speech.* In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pages 129–136, 2013. (Cité en pages 144 et 151.)

[Lallement 2014] R. Lallement, L. de Silva et R. Alami. *HATP : An HTN Planner for Robotics.* In 2nd ICAPS Workshop on Planning and Robotics, 2014. (Cité en pages 106, 143 et 147.)

[Lemaignan 2011] S. Lemaignan, A. Sisbot et R. Alami. *Anchoring interaction through symbolic knowledge.* In Proceedings of the 2011 Human-Robot Interaction Pioneers workshop, 2011. (Cité en page 24.)

[Lemaignan 2012] Séverin Lemaignan, Raquel Ros, E Akin Sisbot, Rachid Alami et Michael Beetz. *Grounding the interaction : Anchoring situated discourse in everyday human-robot interaction.* International Journal of Social Robotics, vol. 4, no. 2, pages 181–199, 2012. (Cité en page 80.)

[Lemaignan 2014] S. Lemaignan, M. Hanheide, M. Karg, H. Khambhaita, L. Kunze, F. Lier, I. Lütkebohle et G. Milliez. *Simulation and HRI : Recent Perspectives.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, (SIMPAR2014), 2014. (Cité en pages 11 et 99.)

[Leslie 1988] Alan M. Leslie et Uta Frith. *Autistic children's understanding of seeing, knowing and believing.* British Journal of Developmental Psychology, vol. 6, no. 4, pages 315–324, 1988. (Cité en page 57.)

[Levin 1997] E. Levin, R. Pieraccini et W. Eckert. *Learning dialogue strategies within the Markov decision process framework.* In ASRU, 1997. (Cité en page 88.)

[Levine 2014] Steven James Levine et Brian Charles Williams. *Concurrent plan recognition and execution for human-robot teams.* In Proceedings of the Twenty-fourth International Conference on Automated Planning and Scheduling (ICAPS-14), 2014. (Cité en pages 122 et 145.)

[Lewis 2007] M. Lewis, J. Wang et S. Hughes. *USARSim : Simulation for the Study of Human-Robot Interaction.* Journal of Cognitive Engineering and Decision Making, vol. 2007, pages 98–120, 2007. (Cité en page 99.)

[Liu 2014] Rui Liu, Xiaoli Zhang et Songpo Li. *Use context to understand user's implicit intentions in Activities of Daily Living.* In Mechatronics and Automation (ICMA), 2014 IEEE International Conference on, pages 1214–1219, Aug 2014. (Cité en pages 122, 123 et 135.)

[Lomas 2012] M. Lomas*et al.* *Explaining Robot Actions.* In Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction, HRI '12, 2012. (Cité en page 168.)

[Lucignano 2013] Lorenzo Lucignano, Francesco Cutugno, Silvia Rossi et Alberto Finzi. *A dialogue system for multimodal human-robot interaction.* In Proceedings of the 15th ACM on International conference on multimodal interaction, pages 197–204. ACM, 2013. (Cité en page 99.)

[Matuszek 2010] C. Matuszek, D. Fox et K. Koscher. *Following Directions Using Statistical Machine Translation.* In Int. Conf. on Human-Robot Interaction. ACM Press, 2010. (Cité en page 24.)

[Mausam 2012] Mausam et Andrey Kolobov. Planning with markov decision processes : An ai perspective. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012. (Cité en page 127.)

[Mavridis 2005] N Mavridis et D. Roy. *Grounded situation models for robots : Bridging language, perception, and action.* In In AAAI-05 Workshop on Modular Construction of Human-Like Intelligence, 2005. (Cité en page 24.)

[Milgram 1984] P Milgram, R Vanderwijngaart, H Veerbeek et OF Bleeker. *Multi-crew Model Analytic Assessment of Landing Performance and Decision-making Demands.* 1984. (Cité en page 19.)

[Miller 1982] Randolph A Miller, Harry E Pople Jr et Jack D Myers. *Internist-I, an experimental computer-based diagnostic consultant for general internal medicine.* New England Journal of Medicine, vol. 307, no. 8, pages 468–476, 1982. (Cité en page 19.)

[Milliez 2014a] G. Milliez, M. Warnier, A. Clodic et R. Alami. *A framework for endowing an interactive robot with reasoning capabilities about perspective-taking and belief management.* In Robot and Human Interactive Communication, 2014 RO-MAN : The 23rd IEEE International Symposium on, pages 1103–1109, Aug 2014. (Cité en pages 11 et 69.)

[Milliez 2014b] Grégoire Milliez, Emmanuel Ferreira, Michelangelo Fiore, Rachid Alami et Fabrice Lefèvre. *Simulating Human Robot Interaction for Dialogue Learning.* In SIMPAR, pages 62–73, 2014. (Cité en pages 12, 98, 103 et 105.)

[Milliez 2016] Gregoire Milliez, Raphael Lallement, Michelangelo Fiore et Rachid Alami. *Using Human Knowledge Awareness to Adapt Collaborative Plan Generation, Explanation and Monitoring.* In ACM/IEEE International Conference on Human-Robot Interaction, HRI'16„ 2016. (Cité en pages 12 et 143.)

[Mohseni-Kabir 2015] A. Mohseni-Kabir, C. Rich, S. Chernova, C. L. Sidner et D. Miller. *Interactive Hierarchical Task Learning from a Single Demonstration.* In Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction, HRI '15. ACM, 2015. (Cité en page 167.)

[Murphy-Chutorian 2005] Erik Murphy-Chutorian et Jochen Triesch. *Shared features for scalable appearance-based object recognition.* In Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on, volume 1, pages 16–21. IEEE, 2005. (Cité en page 27.)

[Nagai 2015] T. Nagai, K. Abe, T. Nakamura, N. Oka et T. Omori. *Probabilistic Modeling of Mental Models of Others.* In Proceedings of the 24th IEEE International Symposium on Robot and Human Interactive Communication, Aug 2015. (Cité en page 122.)

[Nakaoka 2007] S. Nakaoka, S. Hattori, F. KANEHIRO, S. Kajita et H. Hirukawa. *Constraint-based Dynamics Simulator for Humanoid Robots with Shock Ab-*

sorbing Mechanisms. In IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS2007), 2007. (Cité en page 99.)

[Nau 1999] D. Nau, Y. Cao, A. Lotem et H. Muñoz-Avila. *SHOP : Simple Hierarchical Ordered Planner.* In IJCAI, pages 968–973, 1999. (Cité en page 147.)

[Ng 2000] Andrew Y Ng, Stuart J Russell*et al. Algorithms for inverse reinforcement learning.* In Icml, pages 663–670, 2000. (Cité en page 122.)

[Nilsson 1969] Nils J Nilsson. *A mobile automaton : An application of artificial intelligence techniques.* Rapport technique, DTIC Document, 1969. (Cité en page 21.)

[Noble 1989] D Noble. *Application of a theory of cognition to situation assessment.* Vienna, VA : Engineering Research Associates, 1989. (Cité en page 16.)

[O'Keefe 1999] J. O'Keefe. The spatial prepositions. MIT Press, 1999. (Cité en page 24.)

[Pacherie 2012] Elisabeth Pacherie. *The phenomenology of joint action : Self-agency vs. joint-agency.* Joint attention : New developments, pages 343–389, 2012. (Cité en page 8.)

[Pearl 1986] Judea Pearl. *Fusion, propagation, and structuring in belief networks.* Artificial intelligence, vol. 29, no. 3, pages 241–288, 1986. (Cité en page 20.)

[Petit 2013] M. Petit*et al. The Coordinating Role of Language in Real-Time Multimodal Learning of Cooperative Tasks.* Autonomous Mental Development, IEEE Transactions on, vol. 5, no. 1, pages 3–17, March 2013. (Cité en page 144.)

[Pew 1998] Richard W Pew, Anne S Mavor*et al.* Modeling human and organizational behavior : Application to military simulations. National Academies Press, 1998. (Cité en page 16.)

[Prommer 2006] Thomas Prommer, Hartwig Holzapfel et Alex Waibel. *Rapid simulation-driven reinforcement learning of multimodal dialog strategies in human-robot interaction.* In INTERSPEECH, 2006. (Cité en page 99.)

[Ramırez 2009] Miquel Ramırez et Hector Geffner. *Plan recognition as planning.* In Proceedings of the 21st international joint conference on Artifical intelligence. Morgan Kaufmann Publishers Inc, pages 1778–1783. Citeseer, 2009. (Cité en page 121.)

[Rao 1989] Anand S Rao et Norman Y Foo. *Minimal Change and Maximal Coherence : A Basis for Belief Revision and Reasoning about Actions.* In IJCAI, pages 966–971, 1989. (Cité en page 42.)

[Rieser 2008] Verena Rieser et Oliver Lemon. *Learning Effective Multimodal Dialogue Strategies from Wizard-of-Oz Data : Bootstrapping and Evaluation.* In ACL, pages 638–646, 2008. (Cité en page 99.)

[Ros 2010] Raquel Ros, Séverin Lemaignan, Emrah Akin Sisbot, Rachid Alami, Jasmin Steinwender, Katharina Hamann et Felix Warneken. *Which one ? grounding the referent based on efficient human-robot interaction.* In RO-MAN, 2010 IEEE, pages 570–575. IEEE, 2010. (Cité en page 58.)

[Russell 2003] Stuart Jonathan Russell et Peter Norvig. Artificial intelligence : a modern approach, volume 2. 2003. (Cité en pages 19 et 21.)

[Rusu 2007] Radu Bogdan Rusu, Alexis Maldonado, Michael Beetz et Brian P. Gerkey. *Extending Player/Stage/Gazebo towards Cognitive Robots Acting in Ubiquitous Sensor-equipped Environments.* In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) Workshop for Network Robot Systems, 2007. (Cité en page 99.)

[Sebanz 2006] Natalie Sebanz, Harold Bekkering et Günther Knoblich. *Joint action : bodies and minds moving together.* Trends in cognitive sciences, vol. 10, no. 2, pages 70–76, 2006. (Cité en page 145.)

[Shah 2011] Julie Shah, James Wiken, Brian Williams et Cynthia Breazeal. *Improved human-robot team performance using chaski, a human-inspired plan execution system.* In Proceedings of the 6th international conference on Human-robot interaction, pages 29–36. ACM, 2011. (Cité en pages 122 et 145.)

[Siciliano 2008] Bruno Siciliano et Oussama Khatib. Springer handbook of robotics. Springer Science & Business Media, 2008. (Cité en pages 25 et 27.)

[Siméon 2001] T. Siméon, J.-P. Laumond et F. Lamiraux. *Move3D : a generic platform for path planning.* In 4th International Symposium on Assembly and Task Planning, 2001. (Cité en page 60.)

[Singla 2011] Parag Singla et Raymond J Mooney. *Abductive Markov Logic for Plan Recognition.* In AAAI, 2011. (Cité en page 121.)

[Sisbot 2008] E. A. Sisbot, A. Clodic, R. Alami et M. Ransan. *Supervision and Motion Planning for a Mobile Manipulator Interacting with Humans.* 2008. (Cité en page 110.)

[Sisbot 2011] E Akin Sisbot, Raquel Ros et Rachid Alami. *Situation assessment for human-robot interactive object manipulation.* In RO-MAN, 2011 IEEE, pages 15–20. IEEE, 2011. (Cité en pages 37, 60 et 69.)

[Skubic 2004] M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska et D. Brock. *Spatial language for human-robot dialogs.* IEEE Transactions on Systems, Man, and Cybernetics, Part C : Applications and Reviews, vol. 34, no. 2, pages 154–167, 2004. (Cité en page 24.)

[Smith 1995] Kip Smith et Peter A Hancock. *Situation awareness is adaptive, externally directed consciousness.* Human Factors : The Journal of the Human Factors and Ergonomics Society, vol. 37, no. 1, pages 137–148, 1995. (Cité en page 17.)

[Sorce 2015] M. Sorce*et al.* *Proof of Concept for a User-Centered System for Sharing Cooperative Plan Knowledge Over Extended Periods and Crew Changes in Space-Flight Operations.* In IEEE RO-MAN, 2015. (Cité en page 144.)

[St. Clair 2015] Aaron St. Clair et Maja J Mataric. *How Robot Verbal Feedback Can Improve Team Performance in Human-Robot Task Collaborations.* In HRI, pages 213–220, 2015. (Cité en page 145.)

[Stanton 2001] Neville A Stanton, Peter RG Chambers et J Piggott. *Situational awareness and safety.* Safety science, vol. 39, no. 3, pages 189–204, 2001. (Cité en page 16.)

[Stiefelhagen 2007] Rainer Stiefelhagen, Hazim Kemal Ekenel, Christian Fugen, Petra Gieselmann, Hartwig Holzapfel, Florian Kraft, Kai Nickel, Michael Voit et Alex Waibel. *Enabling multimodal human–robot interaction for the Karlsruhe humanoid robot.* Robotics, IEEE Transactions on, vol. 23, no. 5, pages 840–851, 2007. (Cité en page 99.)

[Stiffler 1988] D Stiffler. *Graduate level situation awareness.* USAF Fighter Weapons Review, vol. 26, pages 2–15, 1988. (Cité en page 16.)

[Sundar 2016] S. Shyam Sundar, T. Franklin Waddell et Eun Hwa Jung. *The Hollywood Robot Syndrome : Media Effects on Older Adults' Attitudes Toward Robots and Adoption Intentions.* In The Eleventh ACM/IEEE International Conference on Human Robot Interaction, HRI '16, pages 343–350, Piscataway, NJ, USA, 2016. IEEE Press. (Cité en page 6.)

[Swartout 1985] William R Swartout. *Rule-based expert systems : The mycin experiments of the stanford heuristic programming project : BG Buchanan and EH Shortliffe,(Addison-Wesley, Reading, MA, 1984) ; 702 pages*, 1985. (Cité en page 19.)

[Talamadupula 2014] Kartik Talamadupula, Gordon Briggs, Tathagata Chakraborti, Matthias Scheutz et Subbarao Kambhampati. *Coordination in human-robot teams using mental modeling and plan recognition.* In Intelligent Ro-

bots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pages 2957–2962. IEEE, 2014. (Cité en page 122.)

[Tang 2012] Jie Tang, Stephen Miller, Arjun Singh et Pieter Abbeel. *A textured object recognition pipeline for color and depth image data.* In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 3467–3474. IEEE, 2012. (Cité en page 27.)

[Tellex 2010] S. Tellex. *Natural Language and Spatial Reasoning.* PhD thesis, MIT, 2010. (Cité en page 24.)

[Tenorth 2015] Moritz Tenorth et Michael Beetz. *Representations for robot knowledge in the KnowRob framework.* Artificial Intelligence, 2015. (Cité en pages 42 et 44.)

[Thomson 2010] Blaise Thomson et Steve Young. *Bayesian update of dialogue state : A POMDP framework for spoken dialogue systems.* Computer Speech & Language, vol. 24, no. 4, pages 562–588, 2010. (Cité en page 88.)

[Thrun 2005] Sebastian Thrun, Wolfram Burgard et Dieter Fox. Probabilistic robotics. MIT press, 2005. (Cité en page 27.)

[Tomasello 2005] M. Tomasello, M. Carpenter, J. Call, T. Behne et Moll H. *Understanding and sharing intentions : The origins of cultural cognition.* Behavioral and Brain Sciences, 2005. (Cité en pages 143 et 151.)

[Tomasello 2011] M. Tomasello. *Why We Cooperate, Michael Tomasello. MIT Press, 2009. xviii + 206 pages.* Economics and Philosophy, vol. 27, pages 183–190, 7 2011. (Cité en page 143.)

[Trafton 2005] J. Trafton, N. Cassimatis, M. Bugajska, D. Brock, F. Mintz et A. Schultz. *Enabling Effective Human-robot Interaction Using Perspective-taking in Robots.* IEEE Transactions on Systems, Man, and Cybernetics, 2005. (Cité en page 58.)

[Tulving 1972] Endel Tulving. *Episodic and semantic memory 1.* Organization of Memory. London : Academic, vol. 381, no. 4, pages 382–404, 1972. (Cité en page 20.)

[Tversky 1999] B. Tversky, P. Lee et S. Mainwaring. *Why do speakers mix perspectives ?* Spatial Cognition and Computation, vol. 1, no. 4, pages 399–412, 1999. (Cité en page 56.)

[Vernon 2014] David Vernon. Artificial cognitive systems : A primer. 2014. (Cité en page 44.)

[Warneken 2006] F. Warneken, F. Chen et M. Tomasello. *Cooperative activities in young children and chimpanzees.* Child Development, pages 640–663, 2006. (Cité en page 142.)

[Warneken 2007] F. Warneken et M. Tomasello. *Helping and Cooperation at 14 Months of Age.* Infancy, vol. 11, no. 3, pages 271–294, 2007. (Cité en page 142.)

[Warnier 2012] M. Warnier, J. Guitton, S. Lemaignan et R. Alami. *When the Robot Puts Itself in Your Shoes. Managing and Exploiting Human and Robot Beliefs.* In Proceedings of the 21th IEEE International Symposium in Robot and Human Interactive Communication, 2012. (Cité en page 69.)

[Wimmer 1983] Heinz Wimmer et Josef Perner. *Beliefs about beliefs : Representation and constraining function of wron g beliefs in young children's understanding of deception.* Cognition, vol. 13, no. 1, pages 103 – 128, 1983. (Cité en page 57.)

[Young 2010] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson et K. Yu. *The Hidden Information State model : A practical framework for POMDP-based spoken dialogue management.* Computer Speech and Language, vol. 24, no. 2, pages 150–174, 2010. (Cité en pages 81, 88 et 93.)