Rapport de Stage

Overworld: Raisonnement géométrique pour la robotique d'interaction

5 septembre 2024

Emilie LEDOUSSAL
Sous la direction de
Guillaume Sarthou et d'Aurélie Clodic

Résumé

L'objectif de ce stage réalisé au LAAS-CNRS à Toulouse est d'améliorer les capacités de raisonnement géométrique du logiciel Overworld. Ce logiciel étant dédié à des applications d'Interaction Humain Robot, le but sera de prendre en compte la perspective du partenaire humain dans le calcul de relations égocentrées ainsi que planifier des positions d'approche du robot en fonction de la position de l'humain, tout ça de manière dynamique.

Remerciements

Je remercie d'abord mes encadrants, Guillaume Sarthou et Aurélie Clodic pour m'avoir permis de réaliser ce stage très intéressant et instructif, et un bon encadrement. J'ai beaucoup appris grâce à eux et ils m'ont permis de développer un grand intérêt pour la robotique d'interaction humain-robot. Même si cela n'a pas abouti, je les remercie profondément pour avoir pris le temps de réaliser un sujet de thèse et de m'avoir aidé à candidater pour une bourse.

Je remercie les personnes qui ont partagés mon open-space et qui ont apportés leurs bonne humeur au quotidien. Je remercie les doctorants Adrien Vigné et Bastien Dussard pour leurs précieux conseils. Je les remercie eux, ainsi que Stephy, Fadma, Marie, Dorian et Yiguo pour leurs amitiés qui me sont très précieuses. Je remercie toutes les personnes avec qui j'ai pu avoir des conversations intéressantes pendant les pauses.

Je remercie à plus grande échelle l'équipe RIS et tout le LAAS-CNRS pour m'avoir donné un très bon environnement de travail.

Coté ENSTA Bretagne, je remercie mes enseignants pour m'avoir donné des acquis précieux ayant permis que ce stage se passe bien.

Table des matières

1 Introduction

1.1 Le LAAS-CNRS

FIGURE 1 – le LAAS-CNRS à Toulouse

Le Laboratoire d'Analyse et d'Architecture des systèmes (LAAS-CNRS) est un laboratoire français faisant parti du CNRS. Fondée en 1967 par Jean Lagasse et George Giralt, sous le nom de "Laboratoire d'Automatique et de ses Applications Spatiales". À l'aube des années 1970, les programmes ≪ d'applications spatiales ≫ avec le CNES se font plus rares. À partir de ce moment, il est décidé d'orienter davantage les recherches du laboratoire vers l'étude de l'automatique et des systèmes complexes. Ainsi, le LAAS est rebaptisé "Laboratoire d'automatique et d'analyse des systèmes" en 1973, avant de devenir plus tardivement le "Laboratoire d'analyse et d'architecture des systèmes".

Cette UPR (Unité Propre de Recherche) conduit des recherches dans le domaine de l'informatique, robotique, automatique et micro et nano systèmes. Au sein du département de robotique du LAAS, il y a trois équipes, l'équipe RIS (Robotique et InteractionS), l'équipe RAP (Robotique, Action et Perception) et l'équipe GEPETTO qui se consacre à l'analyse et la génération de mouvement des systèmes anthropomorphes.

1.2 L'équipe RIS

Ce stage a été réalisé dans l'équipe RIS qui développe un projet de recherche portant essentiellement sur les machines autonomes intégrant des capacités de perception, de raisonnement, d'apprentissage, d'action et de réaction. Les sujets de recherche notamment menés dans cette équipe sont liés aux Architectures Décisionnelles et les Interactions et Coopérations Humain Robot. On peut notamment trouver au sein de cette dernière thématique plusieurs aspects décisionnels, tel que :

- la gestion de connaissance
- l'évaluation de situation
- la surveillance des taches et des actions
- la communication
- la supervision (superviser la qualité des interactions et la gestion des attentes)
- la planification.

Plusieurs défis sont pris pour développer une interaction humain-robot efficace, notamment celui de prendre en compte l'interaction de manière globale en rendant le robot capable de faire ses taches, tout en surveillant celles de l'humain. Au sein de cette équipe, il y a un travail pluridisciplinaire avec des psychologues et des philosophes, afin de s'inspirer de concepts en sciences cognitives sur la collaboration et la communication. Le but

n'est pas d'imiter les interactions entre les humains, mais de comprendre ce que l'humain attend du robot, pour pouvoir être d'une plus grande aide.

L'équipe RIS travaille principalement au sein du bâtiment Adream. Celui-ci a la particularité d'offrir une large salle d'expérimentation au rez-de-chaussée, équipée notamment d'une volière pour les robots aériens (de type drones) et d'un décor de cinéma amovible, reproduisant un appartement de 50 mètres carrés, pour les travaux de recherches portant sur les interactions humain-robots (Figure 2).

FIGURE 2 – L'équipe RIS a à sa disposition un espace semblable à un appartement, permettant de reproduire des scènes du quotidien

1.3 Le domaine de la robotique HRI

La Robotique d'interaction Humain Robot est un domaine de la robotique qui s'axe sur les interactions, la communication et la collaboration entre des robots et des humains. Ce domaine étant pluridisciplinaire, on retrouve en son sein, en plus de la robotique, de la psychologie, de la philosophie et de l'ergonomie. Ces disciplines sont essentielles pour comprendre et améliorer les interactions humain-robots. Dans le cadre de la HRI, il est crucial de concevoir des robots capables de communiquer de manière naturelle, intuitive et efficace avec les humains. Cela inclut la compréhension des signaux sociaux et l'adaptation aux besoins des utilisateurs.

1.4 Cadre du stage

Un des projets au sein de l'équipe RIS est le développement d'une architecture robotique nommée DACOBOT (Deliberative Architecture for COllaborative roBOT) adaptée aux interactions Humain-Robot. Cette architecture est constituée de 7 modules liés entre eux, comme représenté dans la figure 3.

C'est dans le cadre de l'évaluation de situation (ou Situation Assessment en anglais) que s'inscrit ce stage. Il s'agit de la capacité de comprendre l'environnement autour de soi, de percevoir les objets qui le compose, d'identifier les différentes personnes et robots qui cohabitent dedans, et toutes les relations qui lient les différentes entités. Cette notion correspond au bloc violet dans la figure 3.

L'objectif de ce stage est d'améliorer les capacités de raisonnement géométrique du logiciel Overworld développé par l'équipe RIS au LAAS-CNRS, qui est la brique logicielle dédiée au Situation Assessment dans l'architecture DACOBOT.

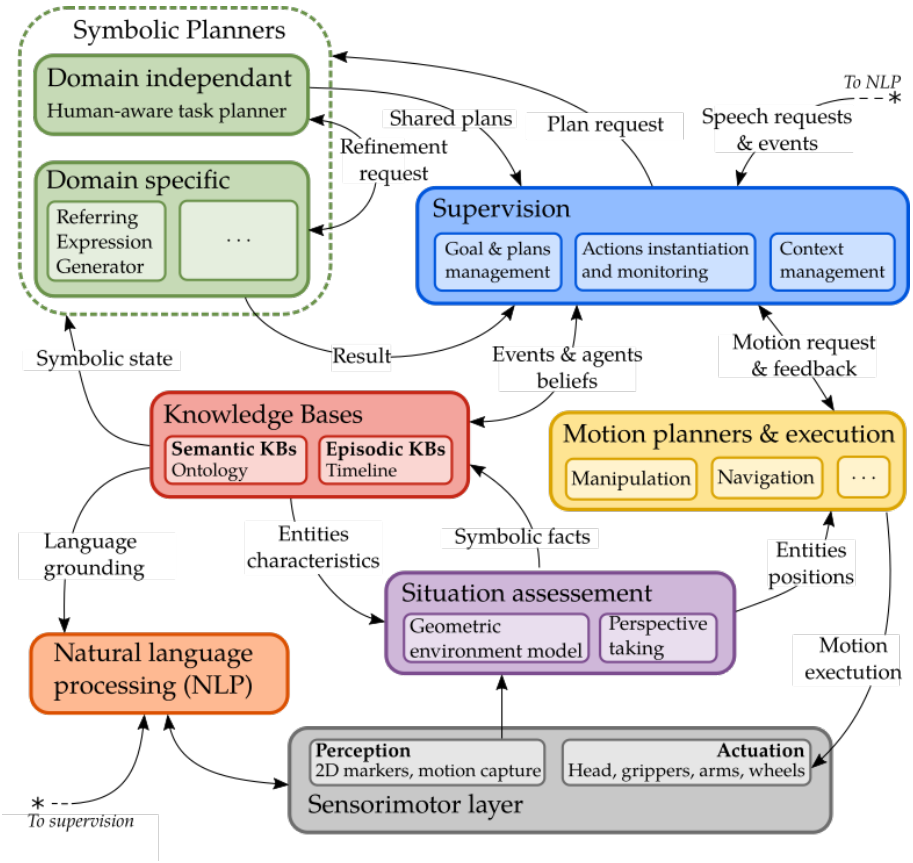

FIGURE 3 – L'architecture Dacobot développée au LAAS-CNRS dans l'équipe RIS

2 Architecture

Bien que mon stage soit focalisé sur le logiciel Overworld, et donc le bloc d'évaluation de la situation, de par son lien fort avec la base de connaissance j'ai également dû m'approprier l'utilisation du logiciel Ontologenius pour mener à bien les développement qui m'ont été confié. En effet, Overworld va à la fois utiliser les connaissances détenues par la base de connaissances Ontologenius pour effectuer ses calculs mais également apporter de nouvelles connaissances vis à vis de la situation qu'il évalue.

Dans cette partie je vais présenter plus en détails ces deux logiciels pour délimiter leurs buts propres ainsi que leurs liens.

2.1 Approche robotique à base d'Ontologies et Ontologenius

Une des particularité de l'équipe RIS sur la thématique de l'interaction humain-robot est l'utilisation du concept d'ontologies pour modéliser la connaissance au sein de l'architecture robotique. Initialement utilisé en majorité dans le Web-sémantique, les ontologies ont pour but de formaliser la représentation de connaissances expertes. Cette formalisation passe par la définition de concepts, de propriétés, d'instances, de relations entre instances mais également de règles. Par exemple, une instance pen_1 issue du concept de stylo peut être relié à une instance table_2, issue du concept de table, via la propriété "est dessus" pour créer la relation (pen_1, isOn, table_2). Grâce aux règles définies dans un ontologie, nous pouvons par exemple nous assurer qu'une telle relation n'est applicable que envers un support, ce qui ici est le cas d'une table. De plus, via une règle décrivant que la propriété "est dessus" est l'inverse de la propriété "est dessous", nous pouvons donc inférer que la relation (table_2, isUnder, pen_1) existe également. Grâce à ces deux exemples nous pouvons donc voir qu'outre maintenir des connaissances, une ontologie permet de valider les connaissances qu'elle représente ainsi que de réaliser des inférences dessus. Une

représentation graphique d'une ontologie est présentée en figure 4.

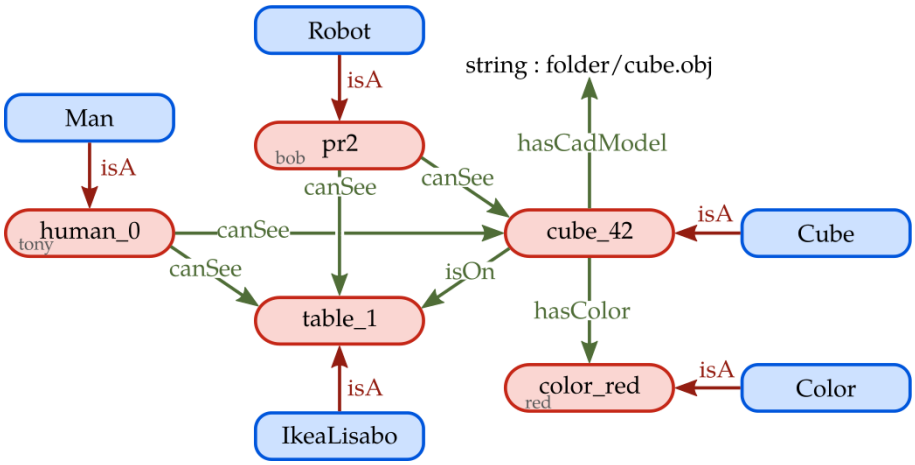

FIGURE 4 – Représentation de la connaissance dans une ontologie, comprenant des classes (en bleu), des entités (en rouge) et relations (en vert).

Une des particularité du logiciel Ontologenius, développé au LAAS-CNRS, est ses fonctionnalités propres à l'interaction humain-robot. Tout d'abord, la représentation de connaissance sous forme d'ontologie a été démontré comme étant proche de notre conceptualisation des connaissances en tant que humain avec entre autre une hiérarchisation des concepts. De plus Ontologenius n'a pas vocation uniquement à représenter les connaissances du robot mais également celles que le robot estime des humains avec lesquels il interagit. Ce principe issue de la psychologie cognitive s'appelle la théorie de l'esprit et tend à formaliser la capacité des humains à estimer es connaissances de autres pour interagir au mieux avec eux.

Ontologenius a été conçu comme étant un serveur unique et central à l'architecture robotique pouvant être questionné et mis à jour par l'ensemble des composants de l'architecture. Pour se faire il a été connecté à ROS. Une représentation d'une instance d'Ontologenius est illustrée en figure 5.

Au cours de mon stage j'ai été amenée à modifier des ontologies pour représenter de nouvelles connaissances et à utiliser les requêtes pour guider mes algorithmes. En plus de cela, une des tâches de mon stage a été de calculer de nouvelles connaissances pour mettre à jour cette base de connaissance.

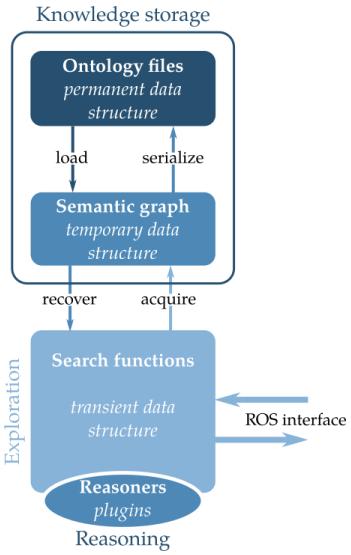

FIGURE 5 – Architecture d'Ontologenius

2.2 Architecture Overworld

Afin qu'un robot puisse interagir avec des humains dans un environnement donné, un besoin clé est de comprendre cet environnement, c'est à dire les objets qui le composent, les autres agents qui agissent dedans (humains ou robots) ainsi que les relations qui lient toutes ces entités. Cette capacité est appelée Évaluation géométrique de situation (Geometrical Situation Assessment) et est globalement liée au raisonnement dans l'espace en temps réel.

Overworld est un framework léger et open-source, développé au LAAS-CNRS, qui fusionne les principales fonctionnalités de l'évaluation géométrique de situation, issues d'une décennie de recherche. Ce logiciel maintient en permanence un état géométrique du monde du point de vue du robot, en utilisant des informations perçues à partir de plusieurs sources, et en les analysant pour créer une représentation cohérente de l'environnement. De plus, Overworld implémente la prise de perspective en émulant la capacité des humains à percevoir afin d'estimer l'état du monde de leurs point de vue. Enfin, un fort lien avec un framework d'ontologies (ici Ontologenius) garantit la cohérence des connaissances dans toute l'architecture robotique.

Overworld s'inscrit dans un effort plus large pour développer une architecture décisionnelle robotique complète, stable et partageable pour l'interaction homme-robot. ROS est utilisé pour faire le lien entre Overworld et les différentes parties de l'architecture.

La figure 6 présente de quelle manière sont liés les différentes briques autour de l'évaluation de Situation. Considérons cette image de bas en haut. À partir des informations issues des capteurs, Overworld crée des entités à partir des objets perçus et les envoie à Ontologenius qui les stockera. Après cela, Overworld pourra demander à Ontologenius les caractéristiques des entités, ce qui lui permettra de calculer différents faits symboliques qui seront eux même envoyés à Ontologenius et stockés. Le travail réalisé au cours de ce stage se situe au niveau de la sous-brique "Geometric reasoning" (Raisonnement Géométrique) au sein du Situation Assessment.

Une des principales taches de ce stage aura été d'améliorer certaines parties de ce logiciel pour permettre de calculer des relations entre les entités.

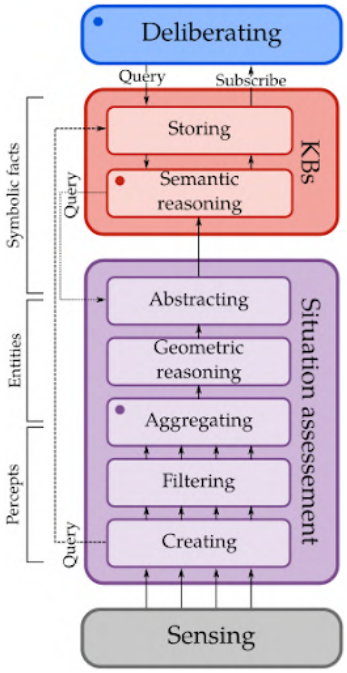

FIGURE 6 – Lien entre le Situation assessment Overworld et la base de connaissances Ontologenius au sein de l'architecture Dacobot

3 Travaux préliminaires

Afin de me familiariser avec les différents logiciels relatifs à ce stage, j'ai effectués quelques travaux préliminaires afin de mettre en place des outils qui simplifierons leurs utilisations.

3.1 Automatisation de création d'environnements de test

Blender est un logiciel open-source de modélisation 3D. Au LAAS-CNRS, il est utilisé pour créer des environnements et modéliser les différents objets qui composent le milieu où évoluent les robots. Dans la figure 7, nous avons un exemple du bâtiment Adream qui a été modélisé.

FIGURE 7 – L'appartement Adream sous Blender

Les environnements de tests que nous utilisons dans Overworld sont définis dans un fichier au format YAML. Ce format permet de lister des données de manière lisible par l'humain et est donc souvent utilisé dans les fichiers de configuration. Le format YAML s'organise en structure hiérarchique dont les indentations représente des relations entre des éléments. La figure 8 est un exemple de fichier YAML comme ils sont utilisés pour configurer les environnements pour Overworld. On considère les entités, leurs positions et leurs orientations.

Afin de simplifier la mise en place d'environnement, il est possible d'automatiser l'exportation d'environnements en utilisant des scripts Python. La plupart des actions qui peuvent être réalisée à la main sur Blender peut également être réalisée par script. Cette exportation doit enregistrer les différents modèles 3D dans des dossiers nommés convenablement, et écrire les positions de chaque objet dans un fichier YAML qui sera utilisable pour la définition d'un environnement. L'une de mes premières tâches a été de mettre en place un tel script.

Ce script exportera essentiellement les objets sélectionnés dans le logiciel Blender, ce qui est utile pour extraire seulement une partie d'un environnement. La première étape est de créer des dossiers par collections d'objets. Il est possible sur Blender de rassembler plusieurs objets dans une collection, permettant de les classer. Par la suite, tout les objets seront traités dans une boucle. Chaque modèle 3D d'objet sera exporté dans le dossier créer à cet effet, on conservant le nom de l'objet. Le script exporte soit au format OBJ soit au format STL. Toujours dans cette même boucle, le script créé un dictionnaire contenant le nom de l'objet, et des sous-dictionnaires contenant sa position et son orientation. Ce dictionnaire sera sauvegardé dans une liste qui sera transformé en un fichier YAML à la fin de la boucle, lorsque tout les objets auront été traités.

Après avoir mis en place une première version de ce script pouvant exporter la position et l'orientation, ce qui correspondaient à ce qui pouvait être lu par Overworld avant le

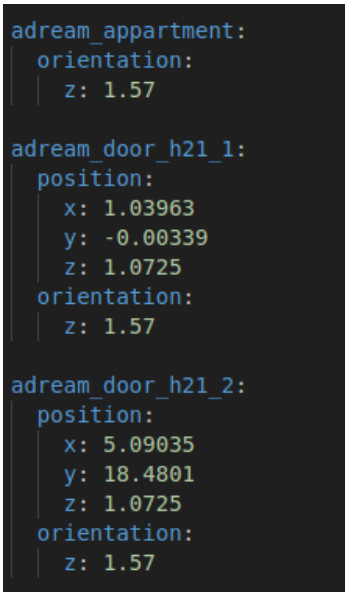

FIGURE 8 – Exemple de l'utilisation des fichiers YAML dans ce projet

début du stage, on a souhaité avoir la possibilité d'exporter des échelles en x, y et z dans le fichier YAML, ce qui pourrait être utilisé pour modifier la taille d'un objet dans une seul dimension. On calcule pour les 3 dimensions les échelles en divisant la dimension de l'objet dans Blender par la dimension du mesh 3D dont il est issu. On crée un sous-dictionnaire supplémentaire pour ranger les échelles dans le fichier YAML.

Afin de simplifier la lecture des fichiers YAML qui peuvent rapidement devenir inutilement longs, on décide d'ajouter une autre fonctionnalité. À l'intérieur des fichiers YAML, certains objets peuvent avoir des caractéristiques nulles, un objet dont l'orientation n'est pas changé ou dont l'échelle n'a pas été modifiée. Un fichier qui contient toutes ces valeurs nulles est beaucoup plus difficile à lire. Les sous-dictionnaires de position, d'orientation et d'échelles ne seront ajoutés au fichier YAML que si au moins une de leurs valeurs n'est pas nulle (0.0 pour la position et l'orientation, 1.0 pour l'échelle).

3.2 Lecture des fichiers au nouveau format de YAML

Parallèlement aux améliorations apportées, il convient de modifier certaines parties du programme pour qu'Overworld soit capable d'utiliser le nouveau format de fichier YAML. Dans la fonction utilitaire pour lire les YAML, on crée une fonction pour vérifier l'existence d'une clé (titre d'un dictionnaire) dans le fichier.

On modifie le module permettant d'ajouter des objets dans l'ontologie à partir de leurs caractéristiques afin d'inclure la notion d'échelle. Le code cherche la mention de position, d'orientation et de scale, extrait les valeurs si elles existent, et crée l'objet. Sans mention de ces clés, l'objet est crée avec les valeurs par défaut mentionnées dans le partie précédente.

On ajoute aussi une fonctionnalité permettant de lire une clé "include" suivie d'un lien vers un autre fichier de configuration qui sera également lu et utilisé, permettant de faire des surcharges de fichiers. Il aura également été nécessaire de documenter ces changements.

Ces travaux préliminaires ont été une prise en main qui a permis une transition plus facile vers le coeur du sujet.

4 Implémentations des Relations Géométriques

Ce que nous utilisons pour définir les propriétés des entités sont ce qu'on appelle des "faits sémantiques". Ils sont parfaits pour être gardés en mémoire par l'ontologie. Ils peuvent caractériser plusieurs propriétés. Un exemple de fait sémantique serait "pen_1 hasColor red" qui permet de définir la couleur d'un objet. Parmi ces faits sémantiques, se trouvent les relations spatiales, qui permettent de situer des objets entre eux, elles seront une notion centrale dans ce stage.

Pour les caractériser, il est nécessaire de structurer l'information en trois éléments clés : un sujet (ou relatum), tel que "La chaise", un objet (ou référent), comme "Le ballon", et une relation (ou prédicat), par exemple "est à droite de". Cette structure est essentielle pour définir les relations géométriques et permettre leur intégration efficace dans le système développé.

Une des taches majeures de mon stage aura été de mettre en place le calcul de relations géométriques. Dans cette partie, nous abordons leurs complexités et leur traitement dans le cadre de ce stage.

4.1 Comprendre la perception des relations géométriques par l'humain

Si pour un humain, s'orienter dans l'espace et distinguer sa gauche de sa droite peut sembler simple, la transposition de ces concepts à la robotique, tout en assurant une cohérence avec la perception humaine, requiert de prendre en compte plusieurs subtilités. Nous nous appuierons sur les travaux en sciences cognitives de Levelt 7 qui traitent de la perception de l'environnement chez l'humain. On souhaite, au cours de ce stage, suivre son principe du "thinking for speaking" et faire en sorte que les relations que l'on va calculer soit compréhensible et fassent sens pour les humains.

4.1.1 Différents types de perspective

Il existe plusieurs façons de se repérer dans l'espace. Prenons comme exemple la scène illustrée en figure 9, composée d'une chaise et d'un ballon. Nous pouvons distinguer trois types de perspectives, parmi lesquelles nous en implémenterons deux.

La première perspective est la perspective absolue, qui repose sur un repère externe à la scène, tel que les points cardinaux. Cette approche est particulièrement adaptée pour les robots mobiles dont l'utilisation n'est pas destinée à interagir avec des humains. Communiquer une position à un humain en termes de coordonnées x, y et z n'est généralement pas très intuitif pour lui.

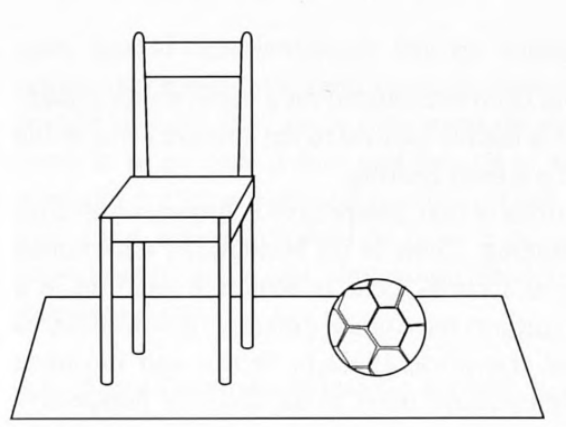

FIGURE 9 – Cette scène peut être décrite de plusieurs façon différente selon la perspective utilisée

La deuxième perspective, plus pertinente pour la robotique interaction humain-robot, est la perspective déictique, qui adopte un point de vue externe, par exemple celui du robot ou d'un humain. Dans le cadre de la scène décrite, en se plaçant du point de vue de l'agent observant la scène, on pourrait dire que "le ballon est à droite de la chaise". Cette perspective est plus appropriée pour faciliter la communication entre un humain et un robot.

Enfin, la troisième perspective est la perspective intrinsèque, qui décrit la scène du point de vue d'un objet. Dans cet exemple, si l'on se place du point de vue de la chaise, on peut dire que "le ballon est à gauche de la chaise". Toutefois, seuls certains objets permettent d'adopter une telle perspective. Par exemple, il est difficile de définir la gauche et la droite d'un ballon, car sa forme sphérique ne s'y prête pas. Contrairement à la perspective déictique, la perspective intrinsèque n'as pas de réciprocité. Si un écran est à la droite d'une chaise, cela ne signifie pas que la chaise est à la gauche de l'écran.

Le choix de ces perspectives est arbitraire, et peut être dépendant de la culture. Selon Levelt, certaines cultures utiliserons un système intrinsèque alors que d'autres préférerons un système déictique. Bien que les culture française et anglaise utilisent ces trois types de perspectives, le choix se fera au niveau de l'individu, selon ses préférences personnelles. Il est donc pertinent d'implémenter à la fois la perspective déictique et intrinsèque, pour faire en sorte que toute personne puisse utiliser ce système.

4.1.2 Principe de sens canonique

Comme évoqué précédemment, seuls certains objets peuvent servir de référence pour une relation géométrique intrinsèque. Ces objets possèdent un 'sens canonique', qui est étroitement lié à leur fonction. Par exemple, un écran a un sens canonique bien défini, car seule une face est fonctionnelle : nous avons collectivement déterminé ce qui constitue l'avant et l'arrière de l'écran. Il en va de même pour de nombreux objets, tels que les chaises, les bureaux ou certains bâtiments comme illustré sur la figure 10.

Il est important de noter qu'un objet mal orienté, c'est à dire dont l'axe vertical n'est pas aligné sur celui du repère dans lequel il se situe (dans le repère absolu dans la plupart des cas), peut perdre son sens canonique, rendant difficile la définition de ses relations intrinsèques. Par exemple, si une chaise est renversée au sol, il devient plus complexe de déterminer ce qui constitue sa gauche ou sa droite.

Un des objectifs de ce stage sera de trouver un moyen de définir ces sens canoniques et de les utiliser pour définir les relations géométriques intrinsèques.

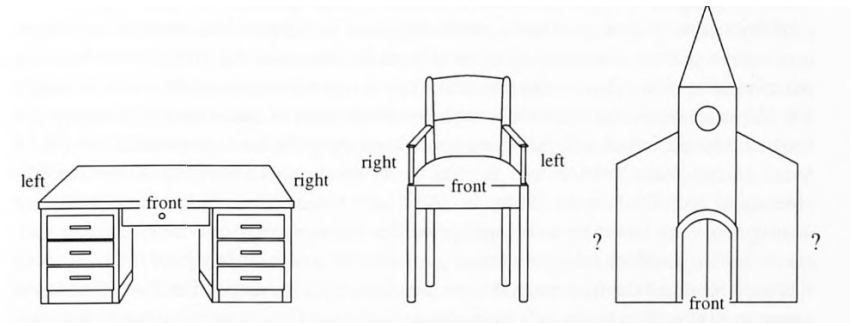

FIGURE 10 – Certains objets ont ce qu'on appelle des "sens canoniques"

4.1.3 Importance de la taille dans les relations géométriques

Levelt souligne également une autre subtilité : la taille des objets influence notre perception de leur proximité. Prenons l'exemple illustré dans la figure 11. Deux bâtiments de la taille d'une maison peuvent être perçus comme étant côte à côte à une distance de quelques dizaines de mètres. Cependant, si un petit objet, comme un crayon, est placé

à cette même distance, notre cognition ne les considère pas comme étant côte à côte en raison de l'écart de taille entre les objets.

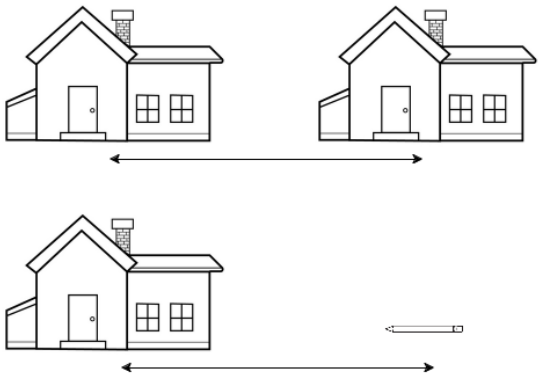

FIGURE 11 – La taille d'objets peut déterminer si on peut les considérer comme "à côté"

Il faudra donc adapter notre programme à ce problème pour qu'il prenne en compte la taille des objets.

4.2 Architecture du module de calcul des relations géométriques

Au cours de ce stage, un de mes objectifs était de revoir la structure du module permettant de calculer les relations géométriques. Nous aborderons ici les classes annexes qui nous ont permis de représenter les relations spatiales, ainsi que la réorganisation du module dans les grandes lignes.

4.2.1 Représentation des relations spatiales dans le programme

Une classe Fact a été crée pour pouvoir utiliser le concept de relations spatiales dans Overworld. Cette classe contient les trois éléments nécessaire à un fait sémantique sous la forme de chaînes de caractères. Dans cette classe, nous avons mis en place divers constructeurs et des fonctions de comparaisons qui serviront ultérieurement.

On mentionnera aussi l'existence d'une structure et d'un type de message ROS nommé un triplet qui sera identiquement constitué de trois string subject, predicate et object. Nous avons créer des fonctions pour passer de l'un à l'autre facilement, puisque les deux ont des usages différents.

4.2.2 Réorganisation du module de calcul des relations géométriques

Le module de calcul des relations géométrique d'Overworld, qu'on nommera RelationsSender, effectue sur demande des calculs qui sont coûteux en ressources et qui n'ont pas besoin d'être actualisés en permanence. Ce module sera donc uniquement utilisé lorsqu'il recevra une requête composé par le logiciel Ontologenius. Cette requête est sous la forme d'une liste de triplets qui peuvent être incomplets. Il est notamment possible de créer une requête en entrant seulement un prédicat et en demandant les relations dans lesquelles il est utilisé. L'interface de debug du logiciel Ontologenius permet à un utilisateur de composer facilement une requête comme nous pouvons le voir sur la figure 12. Après traitement de la requête, RelationsSender calculera les relations spatiales concernés et renverra à l'ontologie une réponse constituée de deux listes de faits sémantiques. Une liste de faits à supprimer et une liste de faits à ajouter.

Il existait déjà une base simple permettant de calculer les relations déictiques pour tout les objets présents, mais nous avons voulu en revoir la structure, pour ajouter plusieurs fonctionnalités qui étaient importantes au bon fonctionnement de cette partie d'Overworld. Avant le stage, il n'était pas possible de calculer les relations relatives à un seul objet,

les faits sémantiques étaient envoyés à l'ontologie sans vérifications qui aurait permis de savoir s'ils existaient déjà ou s'il fallait supprimer d'autres faits obsolètes de l'ontologie, et le calcul des relations intrinsèques n'étaient pas implémentées. Tout les changements apportés seront détaillés dans les parties futures.

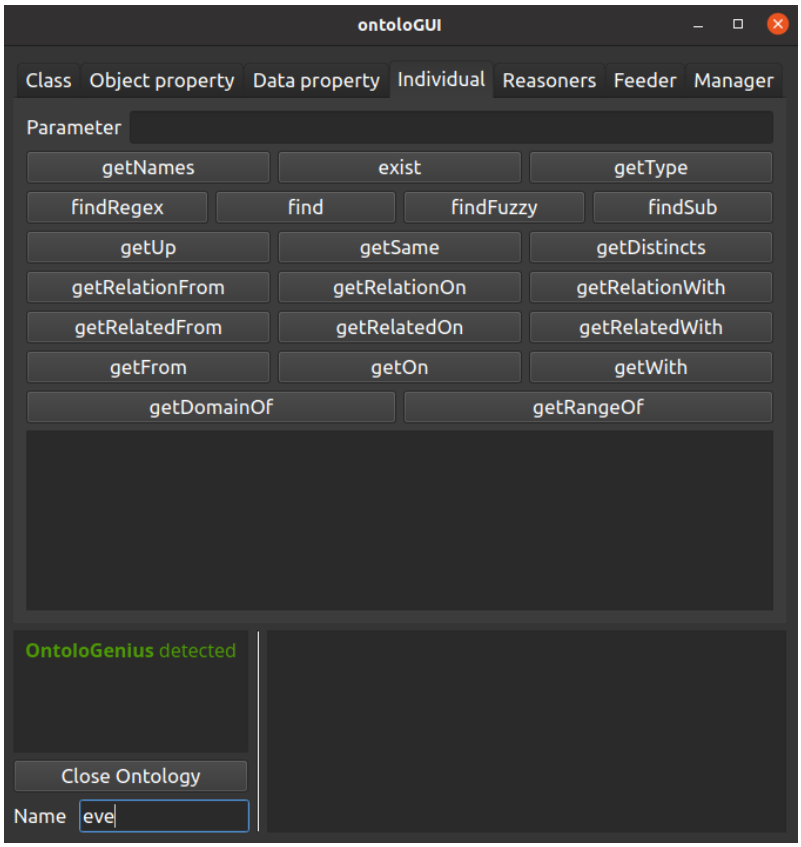

FIGURE 12 – Interface d'Ontologenius permettant de composer des requêtes

La nouvelle algorithmie du service permettant de calculer et d'envoyer les relations géométriques a été représenter dans la figure 13.

Dans un premier temps, il est essentiel d'analyser la requête pour déterminer quel type de relations il faut calculer. En analysant les prédicats dans la requête, on déduit s'il faut utiliser le référentiel déictique ou intrinsèque. Les prédicats de ces deux référentiels sont assez différents pour faire la distinction. Par exemple, un prédicat déictique sera "isAtLeft" tandis qu'un prédicat intrinsèque sera "isAtTheLeftOf". Si un prédicat déictique est présent dans la requête, les relations déictiques seront calculées.

Lors de la composition d'une requête, il est possible de ne pas mentionner un sujet, et juste écrire un prédicat. Si c'est le cas, on calcule les relations dans le référentiel de celui ci pour tout les objets présents. Si un sujet existe dans la requête, on calcule seulement les relations dans lesquelles il est compris.

On rend le calcul des relations plus modulaire dans cette nouvelle version, en utilisant les mêmes blocs de fonctions peu importe si le calcul ne concerne qu'un ou tout les objets.

Avant le calcul, il y a une phase de vérification pour s'assurer que chaque sujet est conforme. C'est à dire qu'il soit localisé, identifié et qu'il n'est pas manipulé par un agent. Si le sujet passe la vérification, il faudra ensuite s'assurer dans un second temps que les autres objets respectent certaines conditions qui dépendront du type de relations demandées dans la requête. Ceci sera fait dans une boucle.

Au sein de cette boucle, lorsque les deux objets pouvant former une relation sont conforme, un calcul est réalisé pour déterminer le prédicat correspondant. Suite à ce calcul, un fait sémantique est formé. L'étape d'après est de garder ce fait en mémoire, et de supprimer les faits qui ne sont pas compatibles avec ce nouveau fait. Pour cela, on utilisera

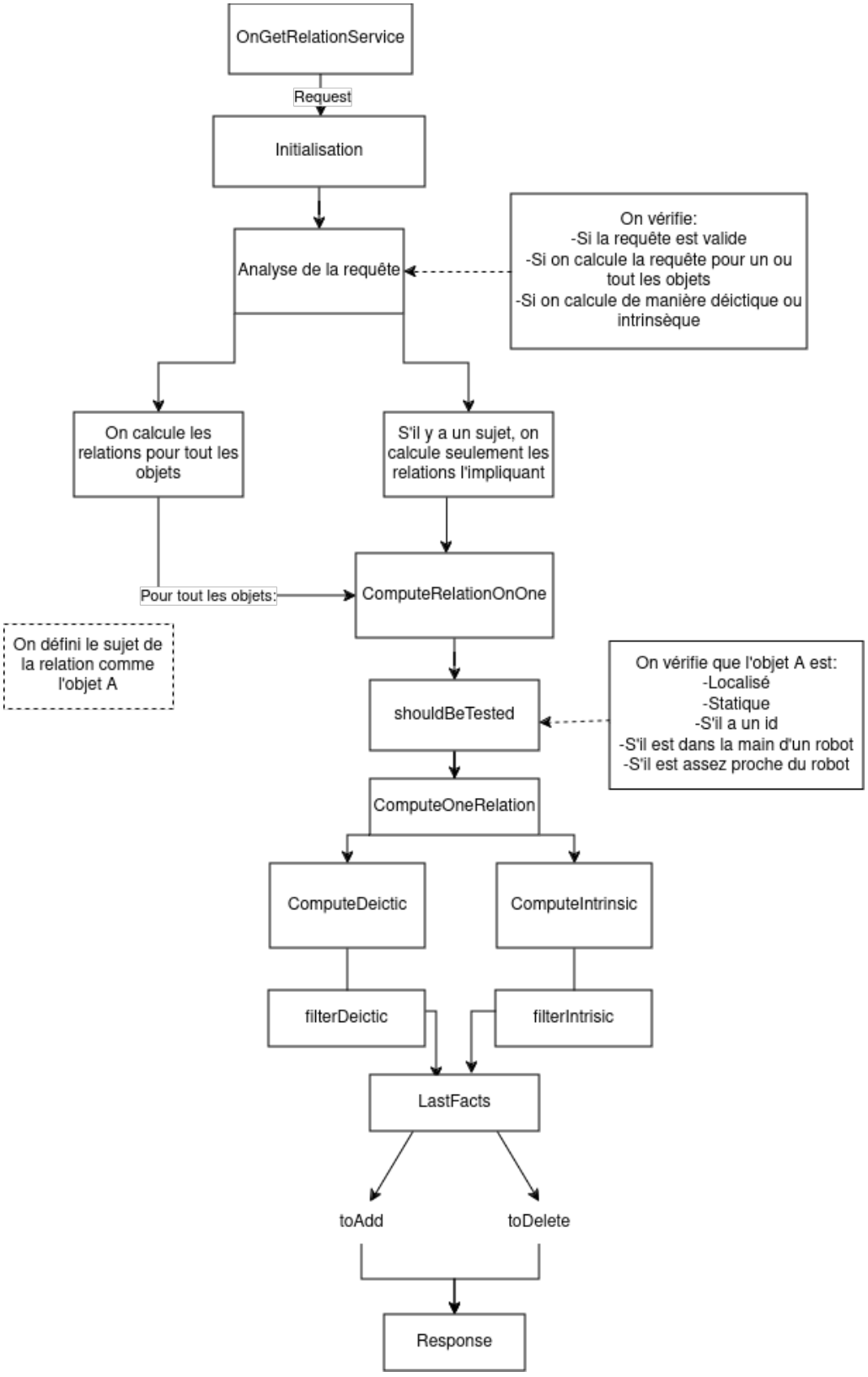

FIGURE 13 – Architecture réorganisée du service de gestion des relations géométriques

une fonction de "tri" qui sera détaillée plus tard. La dernière étape est de composer la réponse et de l'envoyer à l'ontologie, en envoyant les faits à ajouter dans une liste "toAdd" et les faits à supprimer dans une liste "toDelete".

4.3 Détermination des relations spatiales Déictiques

4.3.1 Conditions de calculs des relations spatiales Déictiques

Dans certaines dispositions, il est nécessaire d'éviter le calcul entre deux entités. Dans le cas des relations déictiques, on souhaite notamment éviter de calculer des relations déictiques entre un objet et le meuble sur lequel il est posé. Pour cela, on s'assure que les boites englobantes des deux objets ne se juxtaposent pas sur les axes X et Y, signifiant que les objets sont au même endroit. Une boite englobante (bounding box) est un parallélépipède rectangle qui englobe un objet dans un espace tridimensionnel, elle délimite le volume de l'objet, elle est couramment utilisée pour caractériser la place que prend un objet dans un environnement. Dans un autre temps on vérifie qu'elles ne se juxtaposent pas non plus sur l'axe Z, ce qui signifierai qu'un objet est au dessus d'un autre.

Comme expliqué plus haut, la taille influe sur la perception que des objets sont "à coté", pour palier à ceci, on estime que la distance entre les deux objets doit être inférieure à deux fois la taille de l'objet le plus grand. On choisi la dimension la plus grande de l'objet comme référence, puisque cette règle peut autant s'appliquer à une tour qu'à une piscine. La valeur choisie pour définir cette règle reste arbitraire et peut être éventuellement modifiée dans le futur.

La relation liant deux objets qui ne valident pas ces conditions ne sera pas calculée, et si deux objets après avoir été bougés, ne respectent plus ces règles, les fait sémantiques les concernant seront effacés, car obsolète.

4.3.2 Calcul géométrique déictique

Après que nous nous sommes assurés qu'il est bien nécessaire de déterminer les relations géométriques déictiques entre deux objets, nous passons au calcul. On appelera le sujet "l'objet A" et l'objet avec lequel il possède une relation, "l'objet B". Il est nécessaire de calculer le vecteur 2D entre l'agent et l'objet A(On appellera ce vecteur $\vec{v}$), puis le vecteur entre l'objet A et l'objet B ($\vec{u}$). On calcule ensuite l'angle entre ces deux vecteurs à l'aide de la formule suivante : $angle = atan2(\vec{u}_y, \vec{u}_x) - atan2(\vec{v}_y, \vec{v}_x)$.

A partir de cet angle, on détermine le prédicat qui correspond au triplet, en se placant sur un cercle trigonométrique, comme représenté sur la figure 14. Un angle inférieur à $\pi/4$ ou supérieur à $7\pi/4$ indiquera que le prédicat correspondant est "isBehind", puis que du point de vue du robot, l'objet b sera derrière le sujet. On appliquera ce principe à tout le cercle trigonométrique, en ayant préalablement fait en sorte que l'angle en radian soit positif.

4.4 Détermination des relations spatiales Intrinsèques

Dans un autre temps, nous avons voulu implémenter le calcul des relations géométriques intrinsèques, qui sont les relations du point de vue d'une entité possédant un sens canonique, tel une chaise ou un écran.

4.4.1 Implémentation d'un sens canonique dans l'ontologie

Il n'y avait aucune base du concept de sens canonique dans la structure d'Overworld jusqu'à présent. Nous avons dû réfléchir à la manière de l'implémenter.

Il a fallu créer une sous-classe de la classe Object nommée CanonicObject. En plus d'un objet normal, un objet canonique doit posséder un axe frontal et un axe vertical, qu'on nommera "frontAxis" et "upAxis".

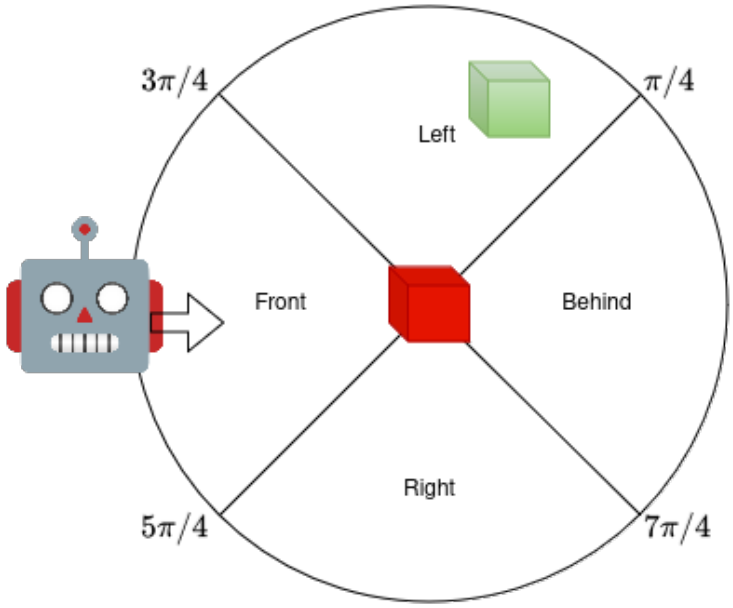

FIGURE 14 – Principe du calcul des relations géométriques déictiques

Une possibilité serait d'implémenter une structure pour les objets possédant un sens canonique sous la forme suivante :

```
CanonicObject:
    dataprop:
    frontAxis:x
    upAxis:z
```

Pour le moment, nous avons fait en sorte que les modèles 3D des objets soient enregistrés pour que le frontAxis soit par défaut x et l'upAxis z. À ce stade, on s'appuie surtout sur le fait que ces objets soient étiquetés "CanonicObject" et enregistrés exactement d'une manière précise.

Il faudra améliorer cette classe dans le futur pour apporter plus de liberté à la création de ces objets canoniques, permettant des axes plus particuliers.

4.4.2 Conditions de calculs des relations spatiales intrinsèques

Bien qu'on réutilise les conditions définies pour les relations déictiques, il est nécessaire d'ajouter quelques règles supplémentaires pour le calcul des relations intrinsèques. Comme expliqué précédemment, on ne peut calculer une relation intrinsèque que si le sujet possède un sens canonique. On exclu de la liste des objets à tester les entités qui n'ont pas l'étiquette "CanonicObject" que nous avons créer précédemment.

Pour la cohérence des relations intrinsèques, il était également nécessaire d'implémenter un moyen de vérifier que les sujets des relations intrinsèques soit positionnés dans un sens logique. On souhaite que l'axe Z de l'objet soit aligné avec l'axe Z de l'environnement. Pour cela, on utilise les estimations du Tangage et du Roulis qu'Overworld calcule en amont (Pitch et Roll en anglais). La figure 15 montre simplement à quoi correspond ces valeurs. Ces deux inclinaisons doivent être inférieure à une marge de 0.35 radian (soit 20°) pour que l'on considère que l'objet est assez droit pour conserver son sens canonique. Cette valeur de 0.35 radian est arbitraire et peut éventuellement être changée dans le futur.

4.4.3 Calcul pour relations intrinsèques

Le calcul des relations intrinsèques est sur le principe assez similaire au calcul des relations déictiques, mais il est nécessaire de faire quelques modifications. L'agent n'est

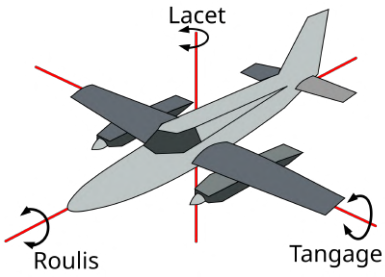

FIGURE 15 – Avec ce schéma, on peut voir que ce qu'on doit surveiller pour estimer l'inclinaison d'un objet est le Tangage et le Roulis

plus compris dans le calcul de la relation, il n'y a plus que le sujet de la relation qu'on appellera ici "Objet A" et l'autre objet qu'on appellera "Objet B". L'objet A devient le centre de la relation, il faut donc transposer l'objet B dans le repère de l'objet A. On utilise ensuite la fonction atan2 pour calculer l'angle entre le vecteur reliant la position de l'objet A et de l'objet B par rapport à la direction du sens canonique de l'objet. Comme précédemment, il convient de faire en sorte que cet angle soit positif.

Dans l'écriture d'une relation intrinsèque, la place du sujet et l'objet sont inversés, au lieu d'avoir "La chaise est à gauche de la balle" on dit "La balle est à la droite de la chaise". Il faut donc échanger l'objet A et l'objet B dans l'ordre de la relation telle qu'elle est utilisée dans le programme. Il faut aussi écrire de nouveaux prédicats, qui sont opposés à ce qu'il y avait pour les relations déictiques. Dans la figure 16, on peut voir les modifications que nous avons fait. La chaise est un objet canonique et est le centre du cercle trigonométrique, l'agent est présent mais n'est pas inclus dans le calcul de la relation. Les nouveaux prédicats sont aussi indiqués.

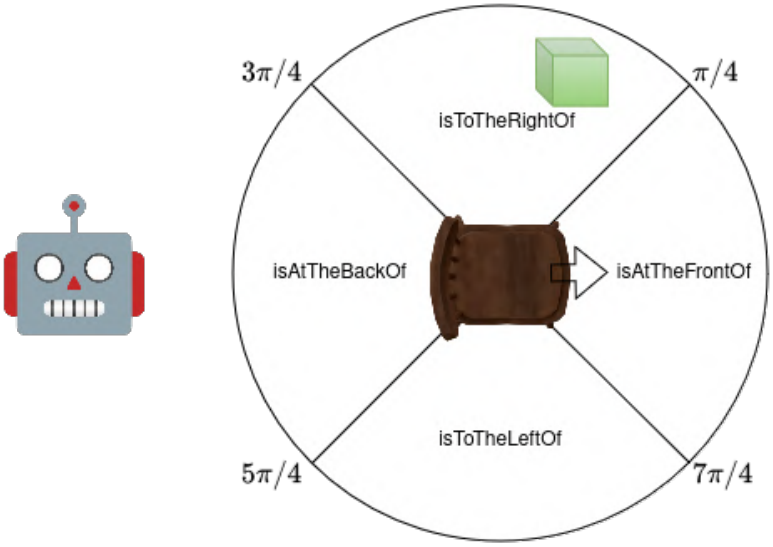

FIGURE 16 – Principe du calcul des relations géométriques intrinsèques

4.5 Envoi à l'ontologie

4.5.1 Liste locale pour les faits calculés

Pour garder la trace des faits sémantiques que nous avons calculés de manière plus locale sans avoir à interroger l'ontologie en permanence, on décide de mettre en place une liste locale pour chaque agent. On décide d'utiliser pour cela un conteneur de type "unordered_map" associant l'identifiant de chaque agent à un conteneur de type "unordered_set"

contenant les faits sémantiques le concernant. Lorsqu'on appelle le service d'évaluation des relations, on vérifie que la liste n'est pas vide, si c'est le cas, on considère qu'il s'agit d'une première itération. Cette structure permet de disposer d'une liste indépendante pour chaque identifiant d'origine (donc par agent), facilitant la gestion des données et permettant un parcours facile des relations, ce qui sera utile dans les parties futures, notamment pour le tri des faits sémantiques qui est nécessaire à la fin de chaque calcul.

4.5.2 Implémentation du tri des faits sémantiques

On rajoute aussi la fonctionnalité de pouvoir trier les faits sémantiques, c'est à dire, supprimer de l'ontologie des faits qui ne sont plus corrects, qui contredisent une relation qui viens tout juste d'être calculée, ou éviter d'envoyer à l'ontologie un fait qui existe déjà, même sous une forme différente, ce qui serait redondant. Nous avons une liste qui contient les faits qui sont considérés comme connus. Avant ce stage, le fait sémantique calculé était ajouté peu importe s'il existait déjà ou s'il y avait un fait contradictoire. Rien n'était implémenté pour contrôler ce qui était ajouté. Nous devons décider quels éléments doivent être supprimés. La version décrite par la suite correspond à une version "entière" utilisée pour le tri des relations déictiques et leurs ajouts dans l'ontologie.

Une relation géométrique devient obsolète lorsque le sujet et l'objet sont impliqués dans une nouvelle relation avec un prédicat différent, tout en conservant le même ordre. Dans ce cas, l'ancien fait doit être supprimé. C'est également le cas si le sujet et l'objet sont croisés, mais que le prédicat est identique. Dans le cas où le fait sémantique est déjà dans l'ontologie, on ne l'ajoute pas de nouveau, c'est le cas si sujet, objet et prédicat sont tous identiques, ou si une relation croisée possède le prédicat opposée au fait qu'on compte ajouter. On n'ajoute ce fait à aucune liste. Si les faits obsolètes ont été supprimés, ou s'il n'y a pas eu de problèmes, on peut ajouter le nouveau fait dans la liste des triplets à ajouter à l'ontologie. Cette logique est représentée dans la figure 17.

Pour appliquer les règles que nous venons de poser, nous créons une fonction filterTriplet qui considérera un Fact passé en entrée et qui parcourra la liste locale des Facts afin de déterminer le devenir de chaque fait dans la liste. Un fait à supprimer sera placer dans une liste temporaire "à supprimer" jusqu'à ce que la boucle soit terminée, afin d'éviter de provoquer des erreurs de mémoire au sein de la liste locale. Le fait à ajouter ne sera ajouté qu'à la fin de la boucle.

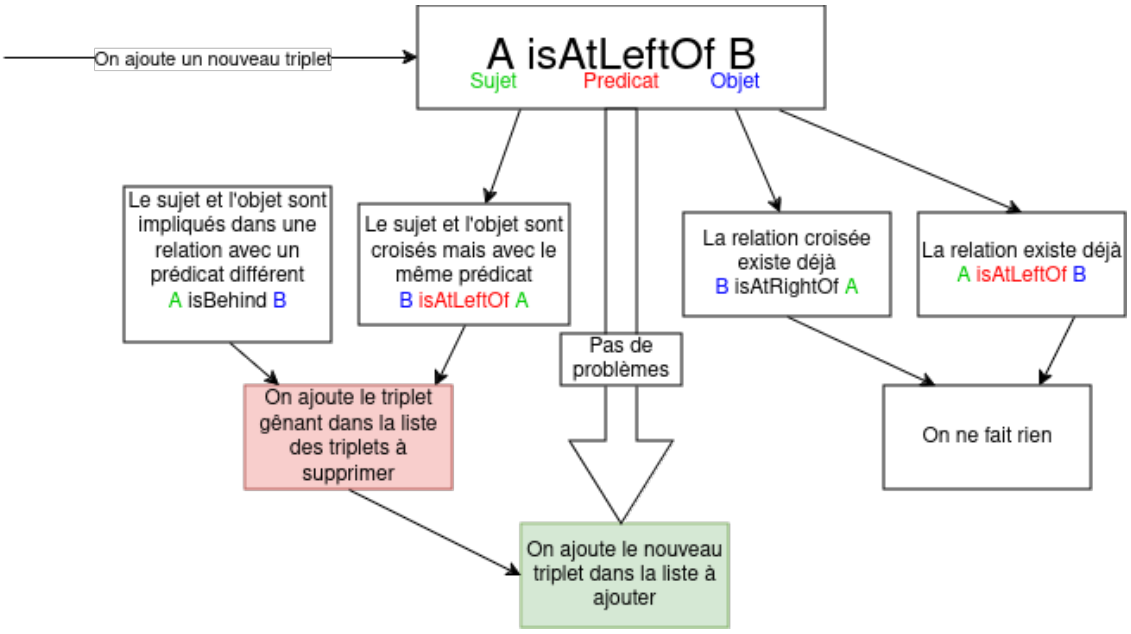

FIGURE 17 – Tri des faits sémantiques lors de l'ajout d'un nouveau fait déictique

Comme expliqué précédemment,la notion de réciprocité n'est pas valable pour les relations intrinsèques, il convient donc de réaliser une fonction de tri similaire mais simplifiée pour leur tri, ne prenant pas en compte la suppression des relations dont le sujet et l'objet sont croisés, on ne considère pas non plus la relation croisée pour décider si on ajoute le fait sémantique dans l'ontologie.

5 Implémentation de la Planification des points d'approche

Une autre tâche a été la création d'un module de calcul de positions d'approche pour un robot en fonction des obstacles de l'environnement et des humains présents. Pour cette partie, on veut pouvoir trouver à partir d'un objet cible (posé sur une table par exemple) et d'une aire où le robot peut circuler, une position pour laquelle il peut attraper cet objet. Pour cela, il faut procéder en plusieurs étapes : Trouver le support sur lequel serait placé l'objet cible, calculer plusieurs points d'approche, appliquer les contraintes d'aires (si un des points d'approche n'est pas dans une aire valide, on ne peut pas l'utiliser), puis lorsqu'on a éliminé tout les points hors d'accès du robot, on choisi la position la plus proche de l'objet. Pour obtenir ce résultat, on découpe ce procédé en plusieurs étapes.

5.1 Détermination du support

Dans un premier temps, on souhaite trouver l'objet sur lequel est posé l'objet cible ou qui le contient. Une première approche a été l'utilisation des bounding box (boite englobante) de nos objets pour déterminer de manière géométrique les objets en dessous de l'objet cible. Mais il s'est avéré qu'il était plus simple de déterminer le support de manière sémantique, en questionnant les données contenues dans l'ontologie. En effet, Overworld possédait déjà un module de calcul de faits sémantiques pour les relation "est dessus" et "est dedans". On cherche les objets qui sont des meubles (étiquetés "Furniture" dans l'ontologie) et qui sont en dessous de l'objet cible (lié à la cible par l'étiquette "isAbove").

On vérifiera que ce support n'est pas au dessus d'un autre support. Si c'est le cas, on considère le premier support comme une autre cible et on recherche récursivement le dernier support qui n'est posé sur rien. C'est celui là qui sera considéré comme l'objet à approcher.

5.2 Calcul de positions

Après avoir déterminer le meuble qu'on souhaite approcher, on calcule un ensemble de positions autour de lui. On récupère les coins de sa boite englobante et on considère sa face inférieure. Autour de cette face inférieure, on trace un rectangle à une distance d, qu'il faudra ultérieurement adapter et fixer en fonction du robot. On calcule le point central de chaque segment du rectangle, et on retourne la liste de ces points. Sur la figure 18 est représenté ce calcul. Le support est le carré gris dont on récupère les coins, le rectangle rouge est celui qui est tracé à une distance d, et les points p1, p2, p3 et p4 sont les points d'approche calculés qui seront retournés par la fonction de calcul.

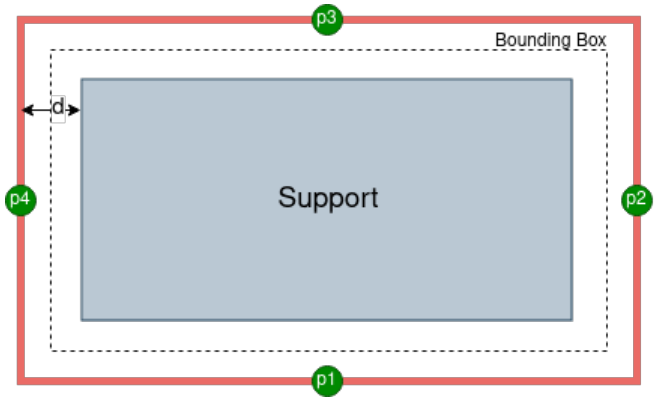

FIGURE 18 – Calcul des points d'approche

Ultérieurement, il faudra prendre en compte la présence d'obstacles et notamment d'humains autour de la table pour pouvoir décaler ces points d'approche. Une hypothèse envisagée dans le futur serait la création d'une zone d'approche autour de la table.

5.3 Application de contraintes

Pour éviter que le robot choisisse comme point d'approche un point inaccessible, comme par exemple en se plaçant depuis une autre pièce, dans une zone où il n'a pas le droit d'aller ou tout simplement dans un endroit où il n'a pas la place de passer, il faut appliquer des contraintes permettant de discriminer les points d'approches.

Dans Overworld, il y a une classe qui permet de définir des zones, ce qui peut être utilisé pour définir les différentes pièces qui composent un environnement. En entrée du service permettant d'obtenir les points d'approches, en plus d'indiquer l'identifiant de l'objet cible, il est possible d'indiquer des contraintes de zones, ce qui permet de discriminer les points qui ne respectent pas ces conditions.

Pour éviter que le robot choisisse un endroit où il ne peut pas passer, on peut créer une zone correspondant à l'espace qu'il prend et s'assurer que si on place cette zone à la position des points d'approche, la zone soit complètement intégrée à la zone autoriser. Si ce n'est pas le cas, on élimine le point d'approche. Ceci est représenté sur le schéma de la figure 19. En reprenant la notation de la figure précédente, le point 4 est éliminé parce qu'il n'est pas dans l'aire autorisée et le point 3 est écarté car l'aire qui correspond au robot ne peut pas s'y placer sans dépasser hors de la zone.

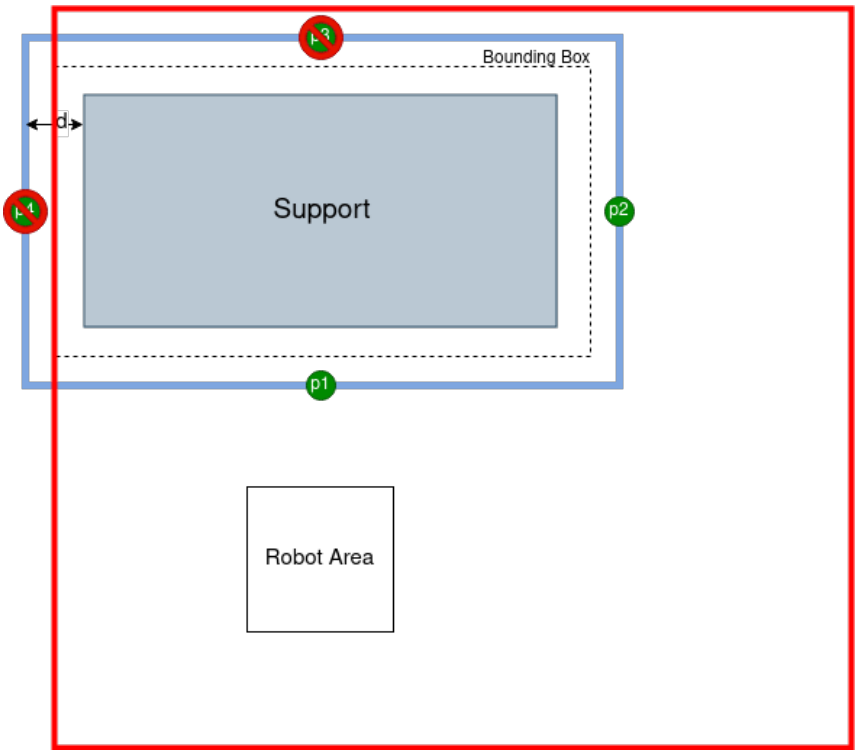

FIGURE 19 – Application des contraintes d'aires sur les points d'approche

Après l'application des contraintes, nous obtenons la liste des points qui sont approchables sans problèmes. Il reste un dernier choix à faire pour obtenir le meilleur point d'approche qui sera utilisé par le robot. On choisi le point le plus proche de l'objet cible qu'on souhaitait approcher et prendre. C'est ce point final qui sera retourné au service ROS dont le but est de fournir le meilleur point d'approche pour un objet.

6 Tests et résultats

6.1 Test en environnement virtuel

Tout au long du développement, nous avons utilisé un environnement de test virtuel en émulant les capteurs du robot pour qu'il puisse "imaginer" un monde autour de lui. Il a fallu pour cela créer des environnements de test pour chaque type de relations ainsi que pour les calculs d'approche.

Pour les tests des relations déictiques, notre environnement était parsemés d'objets simples dont certains était placés de manière à ce qu'ils soient considérés à coté, en terme de taille. Les objets étaient dispersés en plusieurs groupes, afin de pouvoir vérifier que des relations d'objets ne se calculent pas sur des distances incohérentes. Comme on peut le voir dans la figure 20, la nature des objets n'étant pas très importante dans ce cas de figure, nous avons utilisés des portes.

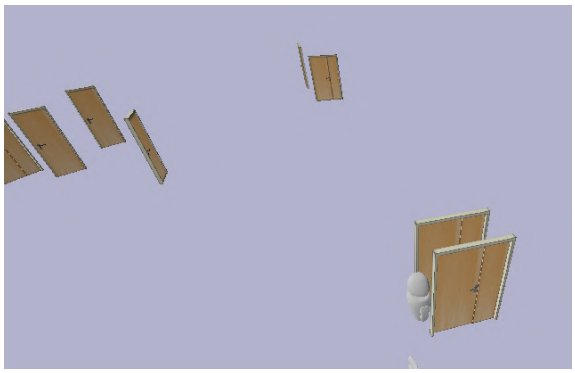

FIGURE 20 – Environnement de test pour les relations déictiques

Pour les tests des relations intrinsèques, nous avons pris un environnement similaire au premier, mais nous avons ajouté des entités possédant des sens canoniques, tel que des chaises de bureau et des écrans. Un des écrans était renversé pour vérifier qu'il était bien exclu des calculs de relations intrinsèques. Cet environnement est représenté sur la figure 21. L'écran blanc étant renversé tandis que la chaise et l'écran noir se font face.

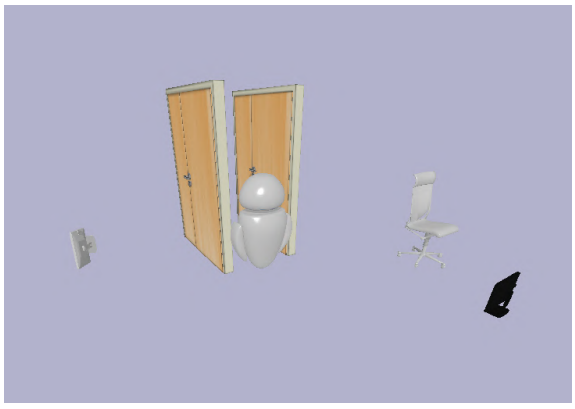

FIGURE 21 – Environnement de test pour le calcul des relations intrinsèques

Pour le calcul des points d'approche, nous avons créer un environnement de test avec une table, et 3 tasses sur celle-ci, afin d'obtenir des points d'approches différents pour chaque tasse . Nous avons aussi tracé une aire autour de la table pour pouvoir valider l'application des contraintes, la forme de cette aire permettant d'exclure certains coté de la table. La figure 22 est une capture d'écran de cet environnement.

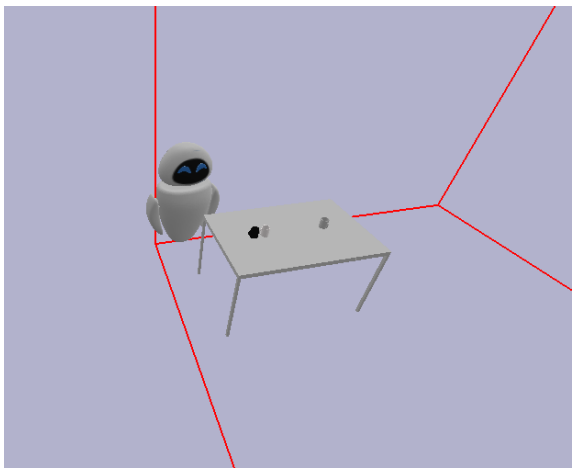

FIGURE 22 – Environnement de test prévu pour le calcul des points d'approches

6.2 Test des calcul des Relations spatiales Déictiques sur PR2

Dans un second temps, nous avons voulu valider les résultats sur les robots de la plateforme Adream. Nous avons testé directement sur un des robots PR2 de l'équipe RIS.

FIGURE 23 – Utilisation du robot PR2 dans un test de relations déictiques

Un premier test a été de présenter des objets au robot et de tester le calcul des relations de son propre point de vue. Les différents objets utilisés sur la plate-forme sont étiquetés par des QR codes qui sont détectés par des caméras qui permettent à Overworld de reconstituer la scène. On bouge les objets pour vérifier tout les cas de figures possibles. Il y a eu des problèmes à plusieurs reprise, ce qui a mené à des séances de débugage jusqu'à ce que le test soit concluant. On s'assure notamment que les relations soient correctes, que les objets ne prennent pas en compte leurs supports, et que la cohérence au niveau des tailles est respectée.

La deuxième partie des tests de relations déictiques a été d'évaluer la capacité du programme à estimer ce que perçoit un humain, notion importante pour la robotique IHR. Nous avons utilisé un casque de Motion Capture pour placer l'humain dans l'environnement. L'humain tourne autour des objets pour s'assurer que ce qui est calculé est cohérent pour lui. Avoir la perspective d'un humain est assez intéressant ici. On teste les mêmes choses que du point de vue du robot. Il a aussi fallu du débugage pour obtenir un bon résultat.

6.3 Autres tests futurs

Bien que nous n'en ayant pas eu le temps, on peut détailler ce qu'il faudra tester dans le futur.

Pour ce qui est des relations intrinsèques, il faudra les tester en conditions réelles avec le robot PR2 sur la plateforme Adream. Les tests seront à peu près similaires à ceux des relations déictiques, mais il faudra mettre en place quelque chose pour indiquer le sens canonique des objets réels qu'il faudra utiliser. Un QR code comme ceux déjà utilisé pour situer les objets dans l'appartement Adream à l'aide des caméras pourrait convenir. L'utilisation d'écrans et de chaises serait un cas de figure idéal pour représenter un environnement domestique. Une fois ces préparations faites, il faudra tester que le robot calcule de manière cohérente les relations géométriques, en veillant bien que si un objet est renversé et ne peut pas respecter son sens canonique, le calcul ne se fasse pas. De la même manière qu'avec les relations déictiques, il faudra tester le point de vue de l'humain à l'aide d'un casque de Motion Capture et vérifier que tout soit cohérent.

Faute de temps et de débuguage à faire, il n'a pas été possible de tester le calcul des points d'approche que ce soit en environnement virtuel ou sur la plateforme Adream avec PR2. L'environnement de test a déjà été détaillé auparavant

6.4 Résultats

Dans l'environnement virtuel mentionnés plus haut, le calcul et l'envoi à l'ontologie des relations déictiques et intrinsèques fonctionnent de manière satisfaisante. Les faits sémantiques calculés sont parfaitement cohérents.

Les tests des relations déictiques sur le robot PR2 sont concluants et fonctionnels. Les faits sémantiques qui ressortent sont corrects et adaptés à la situation, à la fois du point de vue du robot et celui d'un humain. Les tests de relations intrinsèques restent à être faits et pourrons être réalisés ultérieurement, après la fin du stage.

L'algorithmie du calcul des points d'approche est écrite et est codée, mais une phase de débuguage est encore nécessaire pour pouvoir tester le code dans un environnement virtuel puis sur un robot de la plate-forme Adream.

Les ajouts réalisés au cours des travaux préliminaires ont été utiles tout au long du stage, notamment pour créer les environnements de tests.

7 Conclusion

Ce stage m'a offert une expérience enrichissante, tant sur le plan technique que personnel. J'ai eu l'occasion d'approfondir mes compétences en programmation, tout en développant une meilleure compréhension des enjeux liés à la robotique d'interaction Humain-Robot. J'ai notamment renforcé mes méthodologies de travail, en particulier dans la manière de nommer et d'organiser correctement les variables, ce qui est essentiel pour maintenir un code clair et évolutif.

Pour ce qui est du projet, certaines pistes d'amélioration pourraient être envisageable dans le futur, en plus de finir le module de calcul de points d'approche, il faudra implémenter un moyen pour que le point de calcul soit décalé si un autre agent est dans le chemin. Un projet de thèse que nous avons essayé de monter, aurait été d'ajouter un aspect temporel, qui permettrait à un robot de conserver des connaissances passées et estimer des potentiels états futurs. Dans ce projet, on aurait aussi pu modéliser le manque de connaissance et les incertitudes dans Overworld, ce qui aurait pu servir à détecter des problèmes futurs.

Au-delà des aspects purement techniques, ce stage m'a également permis de découvrir le monde de la recherche et ses défis spécifiques. J'ai pu observer de près comment les innovations technologiques peuvent être appliquées à des problèmes réels, en particulier lors du Symposium 2024 Hi ! Paris à l'ENSTA Paris, où j'ai pu explorer diverses applications de la robotique et de l'intelligence artificielle dans des domaines variés. Cet événement

m'a non seulement donné un aperçu des perspectives futures de ces technologies, mais il a également renforcé mon intérêt pour la robotique d'interaction Humain-Robot.

En somme, ce stage m'a non seulement permis d'acquérir des compétences techniques précieuses, mais il a aussi éveillé en moi une réelle passion pour le domaine de la robotique d'interaction, où je compte essayer de continuer.

Références

Guillaume Sarthou. “Overworld : Assessing the geometry of the world for Human-Robot Interaction”. IEEE Robotics and Automation Letters, 2023

Sarthou, Guillaume Mayima, Amandine Buisan, Guilhem Belhassein, Kathleen Clodic, Aurélie. (2021). The Director Task : a Psychology-Inspired Task to Assess Cognitive and Interactive Robot Architectures. 10.1109/RO-MAN50785.2021.9515543.

Guillaume Sarthou : Knowledge representation and exploitation for interactive and cognitive robots. Diss. Université Paul Sabatier, France, 2021.

Guillaume Sarthou, Aurélie Clodic and Rachid Alami. Ontologenius : A long-term semantic memory for robotic agents. In International Conference on Robot Human Interactive Communication (RO-MAN). New Delhi, India, 2019.

Levelt, Willem JM. "Perspective taking and ellipsis in spatial descriptions." Language and space 77 (1996) : 108.