ROS pour Unreal Engine 4

Adrien Vigné, Guillaume Sarthou

ROS pour Unreal Engine 4

Adrien Vigné, Guillaume Sarthou

Mai 2022

Dans le cadre de la réalisation d'études utilisateurs, l'utilisation d'environnements virtuels immersifs permet d'une part d'assurer un contrôle complet de l'environnement avec des conditions identiques pour chaque utilisateur et permet d'autre part la récupération de données (mouvement, direction du regard, etc) [1]. Dans le cadre de la robotique, l'utilisation de simulateurs permet de s'abstraire de contraintes techniques liées aux systèmes physiques (avec la possibilité d'émuler certaines fonctionnalités) et permet de s'affranchir de l'utilisation de plates-formes robotiques réelles onéreuses [7]. Dans les deux cas, l'utilisation d'environnements simulés et immersifs offre la possibilité d'un partage avec la communauté et donc la réplication aisée d'expérimentations. Pour le domaine de l'Interaction Humain-Robot, la mise en place de telles technologies prend donc tout son sens en fiabilisant la partie matérielle, en permettant un contrôle des études utilisateurs, en récupérant des données de qualité et en permettant la réplication d'expérimentations fortement sensibles aux changements.

Le défi sous-jacent est la mise en place d'un lien entre l'environnement immersif et l'architecture robotique via ROS pour récupérer les données de perception (vis-à-vis des objets et des humains), envoyer les consignes de contrôle du robot et s'assurer de la cohérence entre la représentation des connaissances du robot et celle utilisée dans l'environnement immersif. Pour illustrer notre contribution sur ce lien, nous avons choisi d'utiliser le moteur de jeu vidéo Unreal Engine (supportant le lien avec un système de réalité virtuelle), l'architecture robotique DACOBOT [5] basée sur ROS et la tâche du Directeur sur laquelle l'architecture a été testée. La figure 1 illustre la place d'Unreal Engine dans l'architecture. Unreal Engine remplace la couche sensori-moteur ainsi que l'environnement en tant que tel.

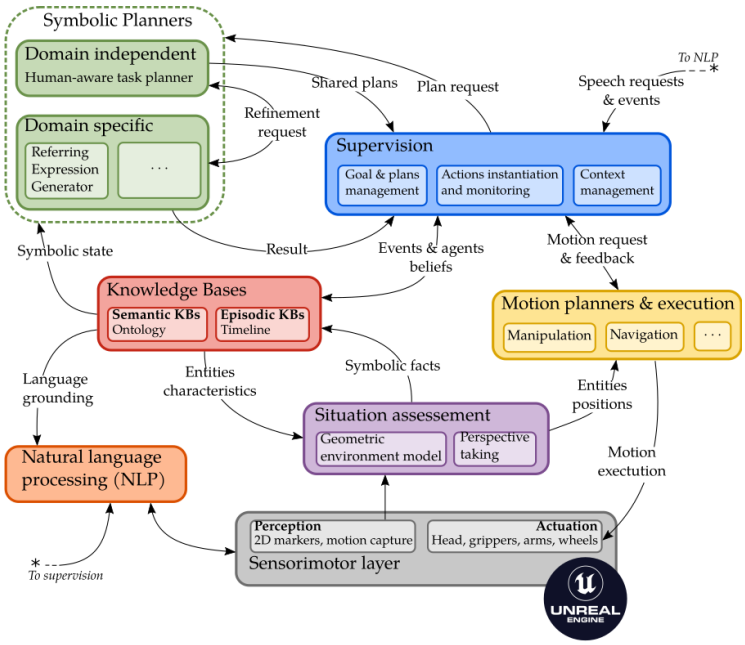

Figure 1: Vu d'ensemble de l'architecture DACOBOT avec la couche sensori-moteur remplacée par Unreal Engine. Chaque lien entre les composants est basé sur ROS.

Afin de rendre l'utilisation d'Unreal Engine possible avec l'ensemble de l'architecture, nous avons développé un plugin permettant de faire le lien avec ROS. Ce dernier fournit des fonctionnalités similaires à ROScpp avec une

interface la plus proche possible de ce dernier. Nous avons identifié deux besoins principaux à l'utilisation de ROS dans Unreal Engine : le remplacement de la couche sensori-moteur et la configuration des entités du monde virtuel.

Les topics sont utilisés en émissions pour fournir les données de perception et en réception pour les commandes de contrôle. Les services sont quant à eux pour le moment utilisé pour permettre à Unreal Engine de questionner la base de connaissances du robot afin d'initialiser l'environnement et d'assurer une cohérence dans le système global.

Le lien entre Unreal Engine et ROS n'est pour autant pas nouveau. Nous pouvons déjà trouver en open-source trois projets majeurs: ROSIntegration [2] et UROSBridge [3] développés par l'Institut pour l'Intelligence Artificielle de l'université de Brême et ROSBridge2Unreal [6] maintenue par RWTH Aachen University. Cependant, ces projets existants ne correspondent pas à l'ensemble de nos besoins. Outre des complexités d'utilisation et pour les deux premiers une utilisation en C++ peu adaptée, le problème principal des solutions existantes est la non-synchronisation des services. Cela a pour effet que l'utilisation des services ROS pour des requêtes à une base de connaissances par exemple, est difficilement faisable. Ce dernier point nous a motivé pour proposer un nouveau plugin, se rapprochant davantage du fonctionnement de ROScpp et utilisable à la fois en C++ et en Blueprint (programmation graphique dans Unreal Engine).

Le plugin résultant, ROS4Unreal[1], s'appuie sur le paquet ROSBridge qui permet au travers du protocole réseaux d'accéder aux noeuds et services ROS. Au niveau réseau, la communication s'effectue au travers du protocole WebSocket.

Pour illustrer les nouvelles possibilités de ce lien entre Unreal Engine et ROS, nous nous focalisons sur la configuration automatique d'un environnement à partir de la base de connaissances du robot. La base de connaissances utilisée dans notre architecture robotique est gérée par le logiciel Ontologenius [4], qui est basé sur l'utilisation d'une ontologie. Ontologenius fonctionne en tant que serveur, interrogeable et modifiable via ROS par l'ensemble des composants de l'architecture. L'utilisation d'une ontologie permet une description sémantique de l'ensemble des entités (physiques ou conceptuelles) de l'environnement. Pour une entité donnée, nous avons dans la base de connaissances la hiérarchie de ses types et un ensemble de relations vers d'autres entités (A isOnTopOf B) ou vers des données brutes (A hasMass 0.2).

Le processus permettant la configuration automatique d'un environnement est illustré en figure 2. La scène utilisée est entièrement modélisée en utilisant le logiciel Blender. Nous l'utilisons à la fois pour modéliser les objets, les texturer et définir la position initiale des objets dans l'environnement. Par ailleurs, avec l'utilisation du mode animation, nous pouvons définir plusieurs configurations, représentant différents états initiaux. Pour un état donné, nous pouvons par la suite générer via un script python un fichier yaml décrivant pour chaque entité sa position dans l'environnement.

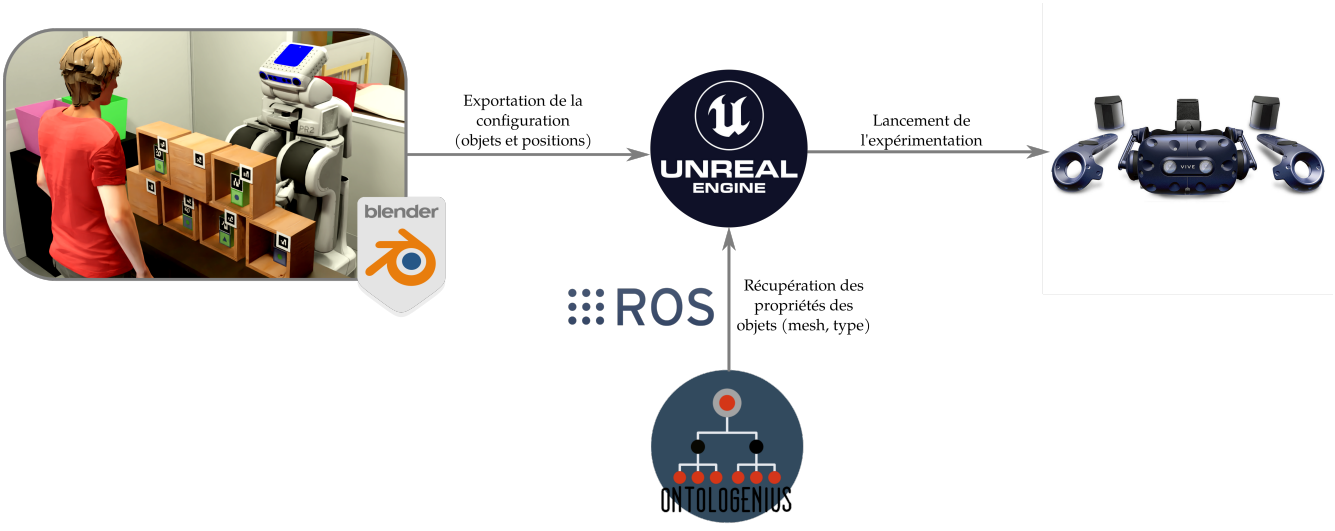

Figure 2: Schéma de la configuration automatique d'un environnement à partir de la base de connaissance du robot et d'un projet Blender.

Du côté d'Unreal Engine, nous avons intégré dans le code de lancement d'un jeu, la possibilité de venir récupérer les informations de ce fichier. Cependant, il ne contient que les positions des entités. Pour connaître le mesh à utiliser,

[1] https://github.com/RIS-WITH/ROS4Unreal

la masse à appliquer ou si l'objet est statique ou préhensible, nous passons donc par l'utilisation de l'ontologie. Grâce à l'implémentation des services ROS avec une synchronisation et une interface similaire à ROScpp, l'appel de la fonction call est bloquant et le résultat est directement récupérable en sortie d'appel. Avec les solutions existantes précédemment, le programme aurait continué sans attendre le résultat du service. Le résultat aurait été accessible via le déclenchement d'un événement sans avoir connaissance de la source de l'appel.

Ce développement s'inscrit dans la pile logiciel dédiée à l'interaction Humain-Robot développée dans l'équipe RIS (LAAS-CNRS). L'objectif à terme de la mise en place de ce lien entre ROS et Unreal Engine est de pouvoir partager des expérimentations entières afin de les reproduire dans leur globalité ou pour tester des composants avant une intégration sur une plate-forme robotique réelle. Là ou de simple simulateurs permettaient déjà cela pour des systèmes robotiques pures, l'utilisation de la réalité virtuelle offre cela pour la communauté HRI.

References

[1] Yasaman Jabbari, Darren M Kenney, Martin von Mohrenschildt, and Judith M Shedden. Testing landmark-specific effects on route navigation in an ecologically valid setting: a simulated driving study. *Cognitive Research: Principles and Implications*, 2022.

[2] Patrick Mania and Michael Beetz. A framework for self-training perceptual agents in simulated photorealistic environments. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[3] RobcogIAI. Unreal ros bridge topic / service communication via websocket. `https://github.com/robcog-iai/UROSBridge`.

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

[5] Guillaume Sarthou, Amandine Mayima, Guilhem Buisan, Kathleen Belhassein, and Aurélie Clodic. The director task: a psychology-inspired task to assess cognitive and interactive robot architectures. In *International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021.

[6] RWTH Aachen University. Connects a rosbridge instance to the unreal 4 enginet. `https://github.com/VRGroupRWTH/ROSBridge2Unreal`.

[7] Tom Williams, Nhan Tran, Josh Rands, and Neil T Dantam. Augmented, mixed, and virtual reality enabling of robot deixis. In *International Conference on Virtual, Augmented and Mixed Reality*. Springer, 2018.