Contrôle distant d’une flotte de robots

Dans le cadre du projet d'étude de quatrième année nous avons eu la chance de travailler sur ce projet en collaboration avec MITSUBISHI ELECTRIC R&D CENTRE EUROPE. Le but de ce projet est de développer les outils permettant de contrôler une flotte de robots. Les robots doivent être capables de suivre une trajectoire calculée dynamiquement et leur position sera mesurée en combinant les informations locales (odométrie, lidar, etc.) ainsi que celles en provenance d’un système vidéo. Le système est développé à partir des robots TurtleBot3 Burger car ils intègrent un lidar, un système d'odométrie au niveau des roues et sont compatibles avec *ROS*. *ROS* est souvent utilisé dans ce type de projet car c'est un ensemble d'outils informatiques open source permettant de développer des programmes pour la robotique. Pour visualiser les points des lidars de chaque robot et les cartes virtuelles créées avec SLAM (simultaneous localization and mapping) nous avons aussi utilisé *Rviz*. Dans les circonstances actuelles qui empêchent les membres de l'équipe d'interagir entre eux et avec les robots, nous avons choisi d'utiliser le logiciel *Gazebo* pour simuler les robots et leur environnement, pour tester nos programmes et filmer les démonstrations.

Notre objectif est d’organiser un réseau de robot capable de livrer des colis au sein de celui-ci. Ce réseau permet de pallier le problème de batterie des robots, en effet chacun est limité à un périmètre autour de son foyer ou il peut se recharger, les robots se passe les colis dans un ordre établit par un contrôleur global. Ce contrôleur surveille l’état de chaque robot et décide le chemin optimal pour livrer des colis d’un point à un autre, il ordonnance les tâches de façon à réduire le temps de livraison globale en utilisant le moins de robots possible. Le comportement de chaque robot est encadré par une machine à état, de cette façon le robot gère lui-même son déplacement d’un point à un autre, il se positionne dans l’espace par odométrie et prend en compte la cartographie globale et locale déterminée par *SLAM* afin d’éviter les murs et les obstacles dynamiques. Le chemin local emprunté par le robot est calculé avec l’algorithme *purepursuit* par modèle *Simulink*. Grâce à cette architecture, en cas de perte de contact avec le contrôleur, le robot retourne à son foyer (tout en gardant son comportement habituel) en attendant la résolution du problème.

Nous avons aussi développé une partie basée sur du traitement d'image pour déterminer la position de chaque robot et ainsi corriger les erreurs de trajectoire de ces derniers dus à l'odométrie des roues qui peut diverger. La difficulté rencontrée est de trouver des correspondances entre des points de l'environnement réel et leur projection d'image 2D pour en calculer des distances. Pour cela nous utilisons la bibliothèque graphique libre OpenCV et en particulier la librairie ArUco qui est à la base utilisée pour la réalité augmentée. Les marqueurs que fournit cette librairie ont suffisamment de correspondance pour déterminer la position de la caméra. Ainsi l'algorithme que nous avons développé permet de calculer la position réelle d'un marqueur, placé sur un robot, en temps réel, dans le référentiel de marqueurs de références qui sont placés au sol. Ce dernier est très robuste et prend en compte la perspective, l'inclinaison de la caméra et fonctionne avec un ou plusieurs marqueurs de référence sur autant de robots qu'on le souhaite. De cet algorithme, nous avons pu mettre en place d'autres applications qui utilisent le calcul de distance comme la fonction de parking autonome et de suivi de véhicule.

Cet algorithme de parking permet au robot d'aller chercher une place de parking dans une zone donnée. Pour faire cela, il détecte un marqueur ArUco placé devant l'emplacement qui va lui permettre de détecter que c'est une place de parking et de s'orienter pour se garer de façon autonome.

Enfin, pour gérer tous ces développements nous avons mis en place une interface homme-machine grâce à QtCreator. Elle permet d'envoyer l'ordre à un robot de récupérer ou poser un colis quelque part avec des coordonnées ou de la transmettre a un robot dans le réseau (pas forcement à un voisin). Elle permet aussi de visualiser les états de chaque robot en temps réel et de savoir quel colis ils transportent.

En conclusion ce projet pose les bases du contrôle distribué et à encore beaucoup de potentiel d'évolution. En effet, on peut imaginer par la suite ne plus seulement se déplacer sur une surface plane mais sur des reliefs voir même remplacer les TurtleBot par des drones et ainsi illustrer les développements actuels des entreprises comme Amazon qui espèrent mettre en place des livraisons de colis par drone dans plusieurs pays.