ROB314 - Projet

Suivi de personnes avec le robot Husky

COELHO RUBACK Arthur
SANTOS SANO Matheus

Palaiseau, France

Mars 2024

Table des matières

1 Introduction

Ces dernières années, grâce à des avancées significatives en matière d'intelligence artificielle, de capteurs et de systèmes embarqués, les systèmes autonomes sont apparus comme l'une des technologies les plus prometteuses et les plus transformatrices dans le domaine scientifique. Dans ce contexte, le projet vise à développer un système permettant à un robot de la famille Husky d'identifier une personne et de la suivre de manière autonome.

L'idée derrière ce projet est de transformer un robot en un système autonome afin qu'il puisse se déplacer sans que l'utilisateur ait besoin de le contrôler manuellement. Cette technologie pourrait être très utile dans le domaine de la sécurité, en particulier dans l'armée, où un robot à usage militaire pourrait suivre un agent sans que ce dernier n'ait à le contrôler manuellement. En outre, cette nouvelle technologie pourrait également être utile pour le transport et le stockage de marchandises dans le secteur industriel.

Pour faciliter la mise en œuvre et le développement du système autonome avec le robot Husky, le système d'exploitation ROS (Robot Operating System) a été utilisée. ROS est un ensemble d'outils informatiques open source qui offre une variété de ressources et de bibliothèques qui facilitent le développement de systèmes robotiques complexes. La mise en œuvre de ce projet à l'aide de ROS a permis l'intégration efficace de différents composants, tels que le capteur LiDAR et la partie commande du robot.
Les codes pour ce projet sont sur les dépôts github [1, 2].

2 Organisation du projet

Le projet a été divisé en deux parties : la partie identification et la partie contrôle. Chaque partie a été divisée en étapes qui ont été réalisées tout au long de la mise en œuvre du projet.

2.1 Partie identification

Le but principal de la partie identification était de faire en sorte que le robot identifie une personne. Pour ce faire, le robot disposait d'une caméra et d'un LiDAR qui pouvaient être utilisés pour identifier la personne. Cependant, nous avons choisi d'utiliser le capteur LiDAR parce qu'il est capable de représenter des objets en profondeur, y compris des objets à grande distance. Par ailleurs, les données LiDAR ne sont pas affectées par des aspects tels que le contraste de luminosité. Cela sera très important pour identifier la distance de la personne et ainsi contrôler le robot.

Les étapes de la partie identification sont présentées ci-dessous :

1. **Fonctionnement du LiDAR :** La première étape consiste à connecter le LiDAR à notre ordinateur, à extraire le nuage de points et à le convertir en un fichier ROS BAG pour créer un dataset ;

2. **Traitement du nuage de points :** Afin d'identifier une personne, il est nécessaire de traiter le nuage de points provenant du capteur LiDAR pour le rendre moins pollué et plus léger ;

3. **Regroupement en clusters (Clustering) :** Une fois le nuage de points correctement traité, il a été nécessaire de séparer les objets du nuage de points en clusters afin de

séparer l'individu des autres objets ;

4. **Identification des personnes :** Il est très important pour le robot d'identifier et de différencier une personne des autres objets capturés par le LiDAR. Ainsi, on a implémenté une algorithme qui fait l'identification d'une personne à partir des clusters.

2.2 Partie contrôle

L'objectif principal de cette partie est d'implémenter un contrôleur qui commande le robot de manière à ce qu'il puisse suivre une personne.

À cette fin, cette partie a été divisée en plusieurs étapes :

1. **Fonctionnement du robot Husky :** Avant d'implémenter le contrôleur, nous essayons d'abord de connecter notre ordinateur à la carte jetson du robot Husky afin de pouvoir contrôler manuellement le robot à partir de notre ordinateur ;

2. **Implémentation du contrôleur :** Après avoir connecté nos ordinateurs à jetson du Husky, deux contrôleurs proportionnels ont été implémentés. Un contrôleur pour la vitesse linéaire du robot sur l'axe X et l'autre pour la vitesse angulaire du robot sur l'axe Z.

3 Mise en oeuvre

3.1 Partie identification

La partie identification a été implémentée dans notre node *pc_pipeline*. La première étape de cette partie a été d'essayer de connecter le LiDAR à notre ordinateur. Grace à un driver ROS déjà existant [3], la configuration était minimale, nous limitant juste à trouver le bon IP. De cette manière, nous avons pu extraire un nuage de points du LiDAR et créer un fichier ROS BAG à fin de faciliter les testes avec un dataset propre.

Le nuage de points PCL devait donc être traité afin que nous puissions l'utiliser correctement pour identifier les personnes. Nous avons donc procédé à une série de filtres. Pour ce faire, nous avons d'abord dû convertir le message ROS en un nuage de points au format PCL (Point Cound Library). Le nuage de points est filtré en fonction du mode de pipeline défini :

- Mode 0 : Applique uniquement un filtre de grille de voxels (Voxel Grid Filter) pour réduire la densité du nuage de points en remplaçant les groupes de points voisins par un seul point représentatif. Il est très utile pour réduire le nombre de points du nuage à traiter. Il s'agit d'un processus de downsampling ;

- Mode 1 : Applique uniquement un filtre de passage (Pass Through Filter) pour sélectionner uniquement les points qui se trouvent dans certaines plages de valeurs dans une dimension donnée. Ce filtre est important pour supprimer les points qui sont en dehors de notre région d'intérêt, comme, par exemple, les points qui représentent le sol ;

- Mode 2 : Applique le filtre de passage direct, suivi du filtre de grille de voxels ;

- Mode 3 : Les deux filtres mentionnés ci-dessus et en plus le filtre statistique des valeurs aberrantes (Statistical Outlier Filters). Ce filtre est une technique permettant d'éliminer les points considérés comme aberrants du nuage de points, c'est-à-dire les points qui s'écartent de manière significative du comportement général du nuage de points. De cette manière, nous pouvons éliminer le bruit et les objets indésirables du nuage de points.

Chaque cas applique les filtres correspondants au nuage de points et convertit le résultat au format *PointXYZI*. Après le filtrage, le nuage de points est reconverti en un message ROS et publié en tant que */filtered_ cloud*.

Après le filtrage, le nuage est segmenté en clusters, qui sont extraites du nuage de points filtré à l'aide de l'algorithme d'extraction de clusters euclidiennes. Les clusters sont reconvertis en message ROS et publiés sous la forme d'un nouveau nuage de points, */velodyne_ cloud*. Ce nom peu conventionnel est du au paramètre d'entrée du module d'identification d'objets utilisé.

Une fois que le nuage de points filtré étant segmenté en clusters, il fallait un algorithme pour suivre le cluster représentant une personne. Pour ce faire, nous avons implémenté un simple filtre de Kalman qui suivait un cluster, par sa centroïde, qui nous avions sélectionné manuellement. Cependant, cet algorithme présentait deux limites. Tout d'abord, le robot ne pouvait pas déterminer si un cluster représentait nécessairement une personne ou non. Il était donc nécessaire de sélectionner manuellement le cluster représentant une personne pour que le filtre de Kalman la suive, ce qui n'est pas intéressant puisque l'objectif est que le robot soit suffisamment autonome pour identifier une personne et la suivre. En outre, lorsque la personne passait dans l'angle mort du LiDAR, le filtre de Kalman perdait le groupe et ne pouvait plus la suivre.

Par conséquent, afin d'affiner et d'améliorer la détection de la personne, en particulier après son passage dans un angle mort, on a amélioré l'application du filtre de Kalman en le mettant en parallèle.

Dans la partie estimation, le filtre est périodiquement mis à jour par un timer ROS. Pour éviter la modification simultanée du filtre de Kalman par plusieurs threads en même temps, un mutex est utilisé pour garantir qu'un seul thread met à jour le filtre. De cette manière, le filtre de Kalman prédit la prochaine position du centroïde du cluster et les coordonnées estimées sont encapsulées dans un message et publiées par la suite.

Dans la partie correction, les mesures, venues du module de détection, sont les coordonnées du centroïde du cluster, encapsulées dans un message. Par conséquent, chaque fois qu'un message est reçu, l'algorithme implémentée passe en revue toutes les coordonnées des possibles candidats reçues et calcule la distance entre la chacune et l'estimation actuelle du filtre de Kalman. Il sélectionne ainsi la position la plus proche de l'estimation du filtre de Kalman, dans une région définie par la covariance actuelle du filtre. Si aucune observation n'est trouvée dans la distance maximale déterminée, l'algorithme se rend compte qu'aucune mesure proche n'a été trouvée et qu'il y a eu un erreur sur la mesure. Chaque fois que cette erreur de mesure se produit, un compteur est augmentée. Si cet erreur se produit plusieurs fois, le filtre de Kalman est interrompu et effacé. Le robot donc s'arrête. Il s'agit d'une façon de sécurité au cas où le robot perdrait la personne de vue. Dans le bon cas, où il n'y a pas d'erreur et qu'une observation est sélectionnée, le filtre de Kalman est corrigé avec cette nouvelle mesure. Cette opération de correction est protégée par un mutex.

En résumé, nous sélectionnons manuellement au début un cluster qui représente une personne. Le filtre de Kalman prédit ensuite périodiquement la position du centroïde de ce cluster sélectionné. Lorsque nous recevons un message contenant différentes mesures de la estimation de position du centroïde, le filtre de Kalman corrige l'état estimé. Ce processus se déroule en parallèle, car au fur et à mesure que les observations sont reçues et traitées, le filtre de Kalman fait de nouvelles prédictions.

Si le filtre ne reçoit pas des observations du cluster pendant un certain temps, il se basera uniquement sur son estimation jusqu'à ce qu'il reçoive de nouvelles observations. Par conséquent, lorsque la personne est passée dans l'angle mort du LiDAR, le filtre s'est basé uniquement sur ses estimations jusqu'à ce que la personne quitte l'angle mort et que le filtre reçoive de nouvelles informations. Le problème de l'angle mort est ainsi résolu. Il faut bien sur que la personne aie un comportement linéaire.

L'identification des personnes a également posé un autre problème. Le nuage de points a été segmenté en clusters, mais l'algorithme n'a pas été en mesure d'identifier si le cluster représentait une personne ou non. Par conséquent, lorsque la personne était perdue par le robot, un autre cluster aléatoire était sélectionné, tel que des objets et des murs, et le robot se déplaçait vers ces objets au lieu de la personne. Pour résoudre ce problème, nous avons utilisé un SVM (Support Vector Machine) déjà implémenté via un lien GitHub, qui reçoit les clusters du nuage de points et essaie d'identifier si ce cluster est une personne ou non. Ce SVM a déjà été entraîné sur un ensemble de nuage de points pour identifier les piétons dans le cadre de véhicules autonomes. Notre code a donc été adapté pour identifier les personnes à l'aide de ce SVM.

L'une des principales difficultés de notre projet a été de trouver un code permettant d'identifier les personnes à partir du nuage de points. La plupart des modèles, tels que PointNet et ses variantes, ne sont pas entraînés avec des datasets contenant d'individus, mais que d'objets. Cependant, après une longue recherche, nous avons trouvé un papier [4] avec un lien GitHub [5] qui a un modèle SVM entraîné sur un dataset de piétons.

Une fois ce lien GitHub découvert, un autre problème qui a pris beaucoup de temps était l'installation des dépendances et des bibliothèques nécessaires pour exécuter le code obtenu sur GitHub. Comme le code n'est pas très récent, on avait des problèmes avec les versions des bibliothèques et, aussi, l'installation des dépendances n'a pas été clairement expliquée par les auteurs.

En bref, notre module d'identification (*pc_pipeline*) filtrera le nuage de points extrait du capteur LiDAR et regroupera le nuage filtré. De cette manière, l'ensemble des clusters formera le nuage de points qui sera introduit dans le SVM pour l'identification. Nous obtenons ainsi les positions 3D des clusters identifiées comme étant une personne et ces positions sont envoyées à notre module **pc_pipeline**. Cet ensemble de positions constitue les observations qui seront utilisées dans le filtre de Kalman.

Le filtre de Kalman a été modélisé avec 4 états : *x, y, vx, vy*. La matrice d'évolution est une simple intégration par rapport au temps. Le filtre tourne sur le repaire du capteur, alors il estime que la position et vitesse relatives. Vu que en pratique le filtre tourne à une fréquence relativement haute, le robot et l'humain se déplacent lentement, que les observations sont en général de qualité et que le bruit d'estimation a été bien réglé, cette question de repaire ne pose pas de problème lors d'un mouvement du robot.

3.2 Partie contrôle

En ce qui concerne le contrôle du robot, l'objectif principal était d'implémenter le contrôle de la vitesse du robot afin qu'il puisse se déplacer en suivant une personne en toute sécurité. Tout d'abord, il a fallu connecter le robot Husky à nos ordinateurs personnels.

Cette étape de connexion et de communication avec Husky a été l'une des plus difficiles de ce projet et a été un point bloquant dans le déroulement de notre projet, à cause de problème avec l'adressage IP et parce qu'il a fallu beaucoup de temps pour réaliser qu'il était nécessaire d'exécuter un paquet ROS de bas niveau sur le robot Husky afin qu'il puisse envoyer et recevoir des messages sur le système du robot. Après avoir compris qu'il ne fallait pas se connecter directement en serial avec le Husky mais de faire tourner les bons packets sur la Jetson et se communiquer via ROS.

Une fois que nous avons pu nous connecter au robot, le démarrage du système et la connexion de notre ordinateur personnel au Wi-Fi du robot ont été optimisés de manière à les rendre plus rapide, sans qu'il soit nécessaire de connecter le robot à plusieurs câbles pour le manipuler.

Après savoir comment connecter le système et comment le contrôler sans la présence de câbles, un système de contrôle a été implémenté pour que le robot puisse suivre une personne.

Deux contrôleurs ont été implémentés (ou un contrôleur double), un contrôleur proportionnel pour la vitesse linéaire du robot sur l'axe X (devant) et un contrôleur proportionnel pour la vitesse angulaire du robot sur l'axe Z. Chaque fois qu'un message contenant la localisation d'un centroïde du cluster est reçu, on calcule la distance entre le robot et la cible (personne) et la différence d'orientation du robot par rapport à la cible. Deux contrôleurs proportionnels sont ensuite appliqués. Un contrôleur pour la vitesse linéaire qui multiplie un gain K_D avec la distance entre le robot et la cible sur l'axe X. Un autre contrôleur proportionnel pour la vitesse angulaire qui multiplie un gain K_W avec la différence d'orientation du robot par rapport à la cible sur l'axe Z. Une commande de vitesse est ensuite émise pour que le robot suive finalement la personne. Il faut aussi mentionner que le robot essaiera de garder une distance de sécurité de la personne suivie.

4 Résultats

4.1 Partie identification

La première étape de l'identification consiste à connecter correctement le capteur LiDAR à nos ordinateurs personnels. Nous avons une blocage dans cette étape qui a été le problème de l'adresse IP du LiDAR. Au début, nous ne pouvions pas connecter nos ordinateurs au LiDAR correctement parce que nous ne nous connections pas à la bonne adresse IP du capteur. Une fois que nous avons trouvé cette erreur et l'avons corrigée, nous avons pu utiliser le LiDAR et en extraire le nuage de points brut, comme le montre la Figure 1.

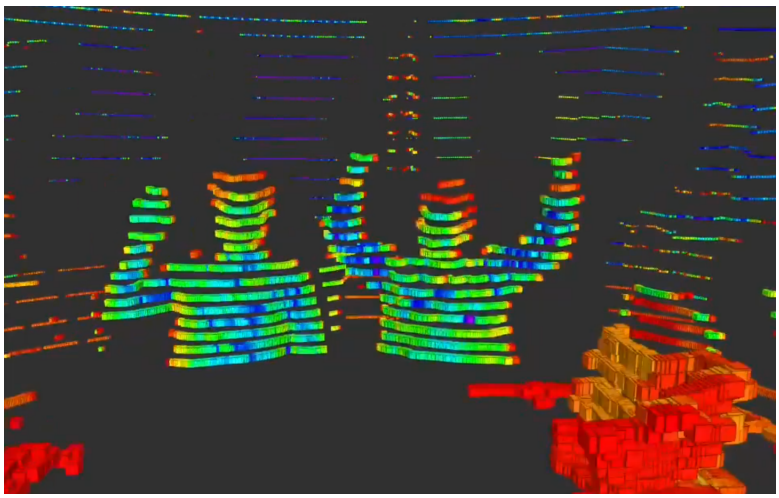
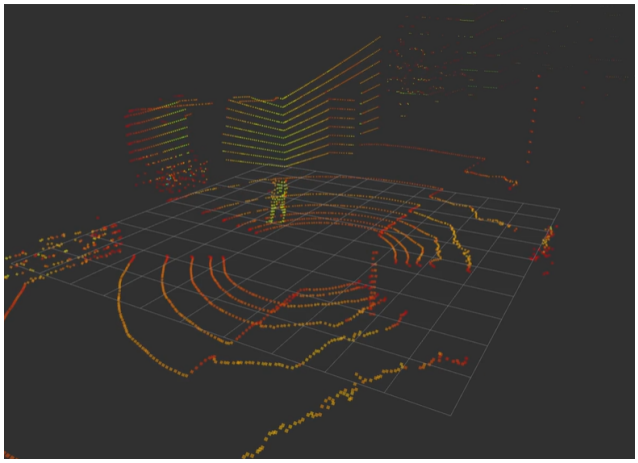
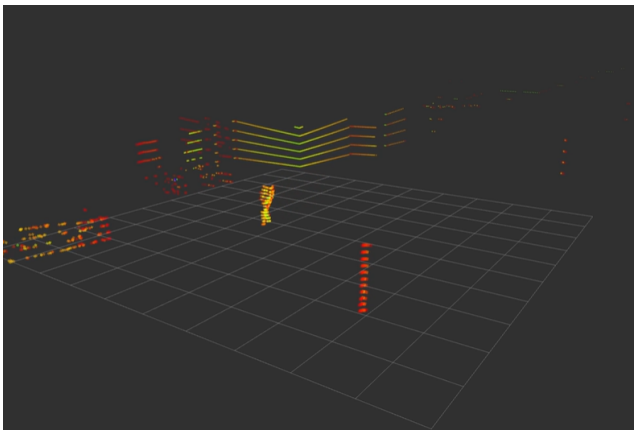


FIGURE 1 – Nuage de points brut obtenu à partir du capteur LiDAR.

Comme le montre la Figure 1, le nuage de points obtenu par LiDAR est très non-uniforme et un nuage pollué par de nombreux points rend difficile l'identification d'une personne. C'est pourquoi, avant même de procéder à l'identification proprement dite, le nuage de points a été filtré. Pour récapituler, 3 filtres ont été implémenté. Un filtre (Voxel Grid Filter) est chargé de réduire la densité des points et la homogénéisé par le downsampling. Un autre filtre (Pass Through Filter) permet de supprimer les points qui ne se trouvent pas dans la plage souhaitée. Un filtre statistique (Statistical Outlier Filter) pour supprimer les valeurs aberrantes. Après avoir passé notre nuage de points brut à travers ces filtres, nous obtenons un nuage beaucoup plus propre, prêt à être utilisé pour identifier des personnes. La Figure 2 montre qu'après le filtrage, le nuage devient moins dense et encore plus visible.



(a) Nuage de points après la filtrage par Voxel Grid Filter.



(b) Nuage de points après la filtrage par les 3 filtres.

FIGURE 2 – Nuages de points filtrés.

Ensuite, le nuage de points filtré a été segmenté en groupes. De cette manière, les ensembles de points sont agglomérés en clusters, ce qui permet de séparer les objets détectés présents dans le nuage de points filtré. Ainsi, un des clusters représente la personne à suivre. Dans la Figure 3, nous illustrons le cluster qui représente la personne, en la séparant du reste de l'environnement.

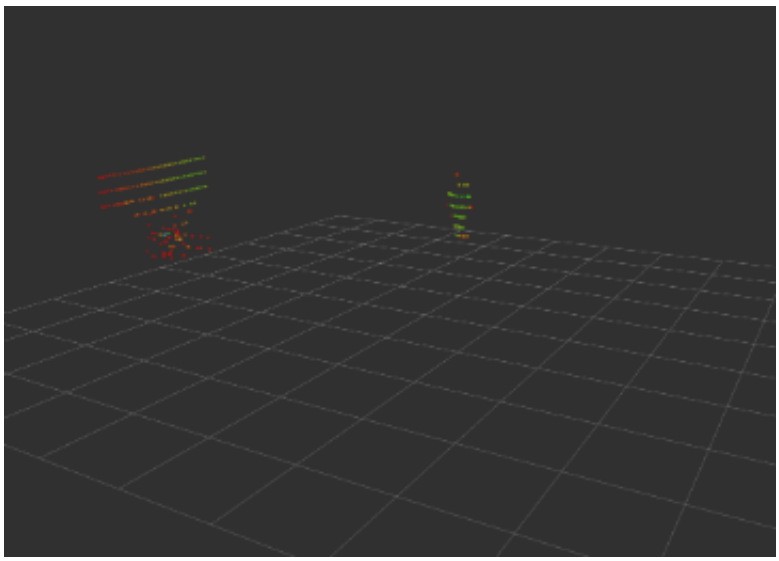


FIGURE 3 – Cluster qui représente la personne à suivre.

Une fois le nuage de points segmenté en clusters, nous devons identifier lequel représente une personne. Pour ce faire, nous avons utilisé un SVM déjà implémenté et entraîné, capable d'identifier les nuages de points représentant des piétons [5, 4]. Comme le montre la Figure 4, le cluster représentant une personne a bien été identifiée.

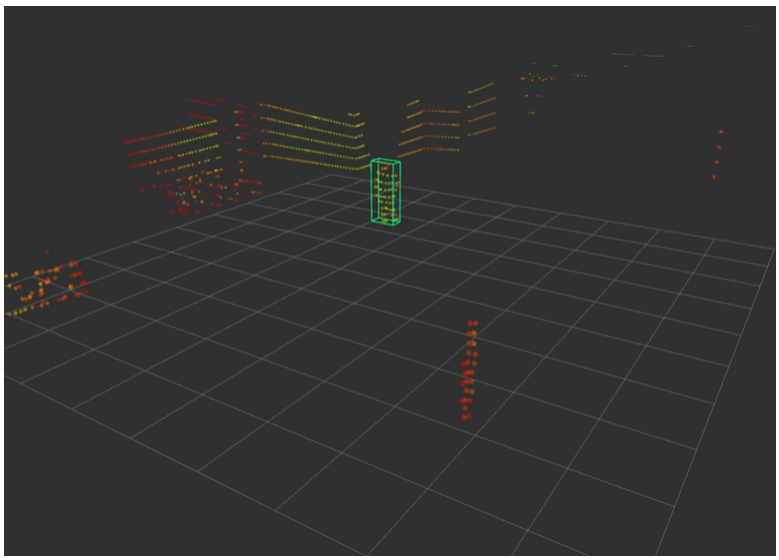
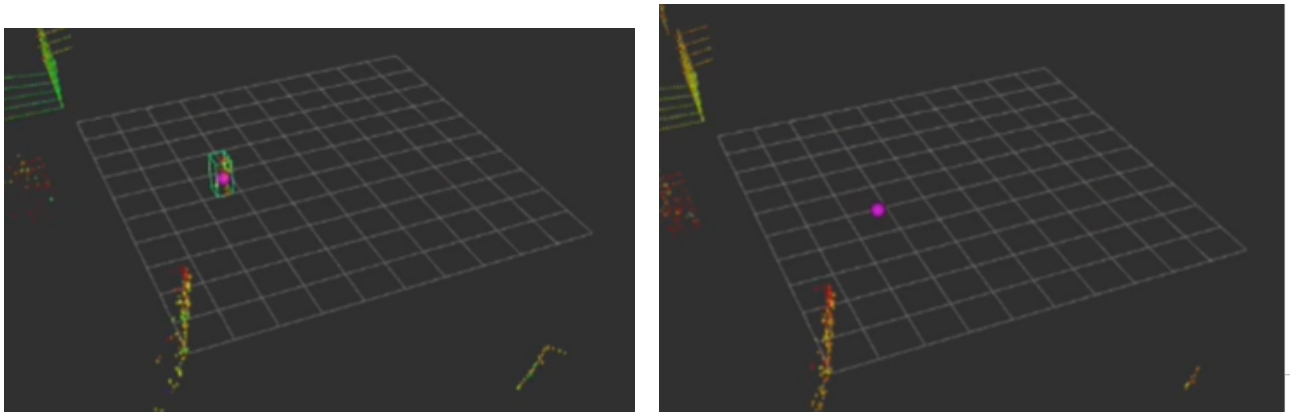


FIGURE 4 – Cluster qui représente la personne à suivre.

Par conséquent, le filtre de Kalman implémenté est destiné à suivre le cluster identifiée comme une personne. Au départ, notre filtre de Kalman a été exécuté dans un seul thread, à la fois pour la prédiction et la correction. Si la personne entrait dans une zone aveugle du LiDAR, le filtre perdait le cluster, ce qui l'amenait à suivre un autre cluster, telle qu'un mur ou un autre objet. Par ailleurs, les valeurs de bruit du filtre étaient mal calibrées, ce qui aggravait ses performances.

Par conséquent, la parallélisation de la prédiction et de la correction était très importante pour améliorer les performances du filtre de Kalman, en particulier dans les cas où la personne n'était plus visible. Après avoir amélioré le filtre et réglé le bruit, nous avons obtenu un résultat très satisfaisant, comme le montre la Figure 5.



(a) Prédiction avec la correction du Kalman (personne visible).

(b) Seulement la prédiction du Kalman (personne non visible).

FIGURE 5 – Fonctionnement du filtre de Kalman amélioré.

Dans la Figure 5a, on peut voir que le cluster de la personne sont visibles par le LiDAR. Par conséquent, le filtre prédit la position du centroïde de ce cluster et effectue également l'étape de correction avec l'observation. Dans la Figure 5b, le scénario est différent. La personne n'est plus visible par le LiDAR, car la personne est entrée dans une zone aveugle du capteur. Par conséquent le filtre prédit uniquement la position du centroïde à partir des positions précédentes.

4.2 Partie contrôle

Pendant la partie identification, la plupart des tests ont été effectués avec les nuages de points enregistrés dans des fichiers ROS BAG. L'implémentation des contrôleurs était donc très importante pour que nous puissions réellement tester l'intégration de toutes les fonctionnalités du robot dans une simulation en temps réel. Il convient de noter que pour que le contrôle fonctionne correctement, il est important que les points envoyées à lui soient sur un repaire relatif au robot.

Après l'implémentation des contrôleurs proportionnels, l'ensemble de l'intégration du système, l'identification et le contrôle, ont été testés dans des simulations en temps réel avec Husky dans un environnement ouvert. Ces tests ont été essentiels pour identifier les problèmes qui n'avaient pas été identifiés dans les simulations en RViz. L'un de ces problèmes est survenu lorsque la personne a traversé une zone aveugle du LiDAR et que le robot, sans trouver le cluster de la personne, est devenu égaré et s'est dirigé vers un autre objet, tel qu'un mur ou une arbre.

Cela a posé un autre problème, le manque de sécurité lorsque le robot est en mouvement. Bien que nous ayons déjà pensé à une marge de sécurité selon laquelle le robot devait être éloigné de la personne lorsqu'il la suivait, nous n'avions pas prévu d'action si le filtre perdait la personne de vue. C'est pour ça que le test avec le robot a été très important pour identifier ces erreurs.

Après avoir corrigé tous les problèmes et ajusté les gains des contrôleurs en testant le robot en temps réel, l'ensemble du système a fonctionné de manière satisfaisante. Le système a été capable d'identifier avec précision une personne dans un nuage de points extrait à partir du capteur LiDAR et de contrôler le robot de manière à suivre une en toute sécurité.

Pour conclure la mise en œuvre du projet, on a fait en sorte que toutes les tâches (filtrage des nuages, partie identification et contrôle du robot) soient effectuées sur la carte Jetson contenue dans le robot. Ainsi, la seule chose qui tourne sur nos ordinateurs personnels est le RViz. De cette manière, nos ordinateurs personnels ne sont pas surchargés par des tâches qui pourraient être exécutées sur la carte Jetson, le traitement se fait sans jitter donc, ce qui améliore l'exécution des tâches et rend le système plus élégant.

5 Conclusion

Le projet a été conçu comme une tentative ambitieuse d'appliquer à un vrai robot les connaissances acquises au cours de la formation en robotique de l'ENSTA Paris. Bien que faire suivre une personne par un robot semble être une tâche simple, sa mise en œuvre présente plusieurs défis et nécessite des connaissances en robotique qui ne sont pas élémentaires. De plus, le fait que nous n'avons jamais travaillé avec des capteurs LiDAR et ROS a rendu le projet beaucoup plus difficile.

Tout au long de nos études, nous avons eu le plaisir d'acquérir des connaissances dans différents domaines qui ont complété et enrichi notre formation. Ces connaissances étaient essentielles pour l'exécution et la réussite du projet.

Grâce à nos connaissances des étapes du filtre de Kalman et son implémentation, nous avons pu implémenter un filtre de Kalman pour le suivi d'un personne. Avec la parallélisation et la protection des ressources avec des mutex dans les systèmes temps réel, nous avons pu améliorer la performance du filtre de Kalman. De plus, des études dans le domaine de l'intelligence artificielle, et plus particulièrement de l'apprentissage automatique, nous ont permis de comprendre mieux le SVM implémenté [5] et d'adapter notre code pour l'utiliser. Par ailleurs, l'un des principaux outils utilisés dans ce projet est le ROS. Grâce à lui, il a été possible d'intégrer plus facilement toutes les parties du système du robot et de s'assurer de son bon fonctionnement.

Au final, ce projet a été l'occasion de nous défier et de découvrir les joies et les peines de la vraie Robotique. Nous tenons à remercier nos professeurs qui nous ont transmis des connaissances essentielles à notre formation, en particulier Mr. Battesti et Mr. Toralba pour leur aide tout au long de ce projet. Nous sommes très heureux et fiers du résultat du projet.

References

[1] Arthur RUBACK et Matheus SANO. *PC Pipeline*. 2024. URL : https://github.com/arthur-ruback/pc%5C_pipeline/ (visité le 25/03/2024).

[2] Arthur RUBACK et Matheus SANO. *Husky Controller*. 2024. URL : https://github.com/arthur-ruback/husky_controller/ (visité le 25/03/2024).

[3] ROBOSENSE. *ros_ rslidar*. 2020. URL : https://github.com/RoboSense-LiDAR/ros%5C_rslidar (visité le 25/03/2024).

[4] Zhi YAN, Tom DUCKETT et Nicola BELLOTTO. « Online learning for human classification in 3D LiDAR-based tracking ». In : *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017, p. 864-871.

[5] YZROBOT. *Online Learning*. 2020. URL : https://github.com/yzrobot/online%5C_learning (visité le 25/03/2024).