ÉCOLE NATIONALE SUPÉRIEURE DES TECHNIQUES AVANCÉES

ROB314
Architectures logicielles pour la robotique

Fabricio VELLONE
Mateus RICCI
Matheus MONTEVERDE

Palaiseau, le 18 août 2021

Table des matières

1 Introduction

L'objectif de ce travail était d'appliquer les connaissances acquises tout au long de la discipline, en utilisant tous les concepts appris précédemment pour la manipulation des capteurs et la communication entre les composants. En outre, les concepts d'un outil appelé ROS, qui est responsable de la communication et du flux de données entre les robots, ont été étudiés dans ce cours. Pour atteindre ces objectifs, le projet était composé de deux étapes, l'une dans un environnement simulé et l'autre plus pratique.

Ainsi, la première étape fait référence à la plupart des travaux réalisés, où les membres de cette équipe ont intégré un défi proposé par l'école polytechnique qui sera plus détaillé dans la section suivante.

La seconde moitié s'est déroulée de manière appliquée, où compte tenu des avancées réalisées dans l'environnement de simulation, les membres ont cherché à appliquer ces concepts adaptés au cas réel, afin de vérifier que les abstractions mises en œuvre fonctionnent dans la pratique. Certaines adaptations ont été nécessaires pour la partie reconnaissance d'images et d'autres problèmes se sont posés tout au long des applications, comme tout sera détaillé dans sa section respective.

2 Le Challenge

L'objectif de ce challenge est de conduire une mission de cartographie et de détection de danger et de personnes dans un territoire hostile, inconnue, afin de préparer l'arrivée d'une équipe d'intervention (militaire ou secours).

Les drones devront communiquer, collaborer entre eux, être capable de gérer des défaillances d'un ou plusieurs éléments de la flotte, pour conduire en mode autonome cette mission de cartographie d'un territoire inconnu, et de détection d'objets significatifs.

Le travail sur le challenge se fera principalement dans un environnement de simulation.

2.1 Point de départ

Chaque équipe dispose d'une flotte de 10 drones et 4 robots terrestres. La mission consiste à découvrir et identifier des personnes visibles ou cachées (derrière des murs, sous des toits, etc.). La découverte consiste en la détection puis l'identification d'une personne. La détection est faite par drones. Dès détection, la procédure est d'envoyer pour identification un robot terrestre. Dès que le robot est à moins d'une distance seuil de la personne, dans l'environnement de simulation, la personne sera considérée comme étant identifiées.

La flotte de drones et de robots terrestres doit être capable de se réorganiser en temps réel, afin de poursuivre le mission. Il est possible que certains drones ou robots soient mis hors d'état de fonctionner pendant la mission, car il y a certaines "kill zones", et il vous faudra réagir et conduire la mission du mieux possible.

La flotte dont vous disposerez sera de 10 drones et 4 robots terrestres.

Le nombre total de personnes détectables et identifiables n'est pas connu.

2.2 Spécifications du Projet

- Une caméra frontale de profondeur est attachée à chaque drone et chaque robot terrestre ;
- Une caméra infra-rouge perçoit les personnes à condition qu'ils soient à moins 10 mètres de hauteur et dans le champs de vision de la caméra ;
- Les personnes sont colorées en vert.
- Les personnes apparaîtrons dans 3 zones de difficulté croissante :
  - Niveau 1 : zone relativement dégagée et visible par une caméra simple ;
  - Niveau 2 : zone cachée par un toît mais relativement visible ;
  - Niveau 3 : zone cachée par un toi et possiblement des murs autour de la personne

2.3 Critères d'évaluation

L'évaluation se fera par ordre hiérarchique sur les critères suivants :

- Nombre de personnes identifiées.
- Nombre de personnes détectées.

En cas d'égalité parfaite sur les critères 1 et 2, le gagnant sera le groupe le plus rapide sur la dernière détection.

3 Le robot terrestre (husky)

Dans la simulation nous disposons de trois robots terrestres dont le modèle est le "husky", montré dans la figure 1. Ce modèle a été équipé d'une caméra frontale de profondeur et d'un "scan" laser qui permet de placer les obstacles autour du robot.

La stratégie à suivre pour les huskies est très simple : ils attendent que les drones détectent une personne, le drone envoie une position dans le repère globale avec une précision donné et le drone essaye de s'approcher de cette position en évitant les obstacles rencontré dans le parcours.

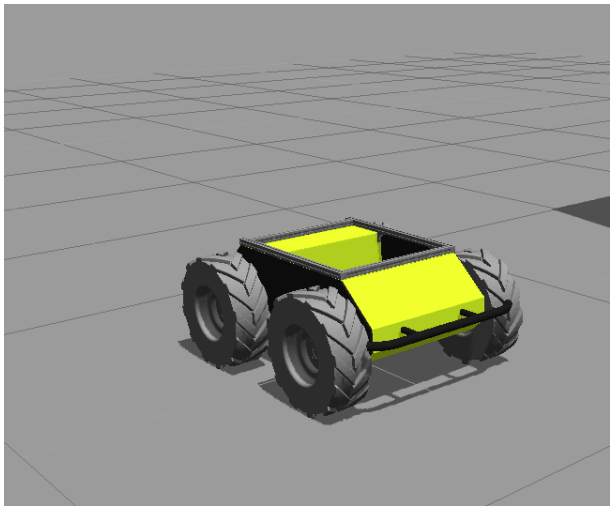


FIGURE 1 – Le modèle husky utilisé pour simuler le robot terrestre dans le challenge

Tout d'abord, afin d'atteindre cet objective, il faut équiper le robot d'un système de détection et évitement d'obstacles. Le paquet ROS que nous avons utilisé et qui permet de faire cela s'appelle "move_base". Ce paquet permets de créer les cartes de coûts qui, en utilisant les capteurs du robot, permettent de définir la présence des obstacles par rapport à la position du robot.

Compte tenu qu'on ne dispose pas d'une carte tout algorithme utilisé doit être implémenté d'une manière locale, c'est-à-dire, on ne pourra pas faire des calculs de planification de chemin globales. Dans notre cas, nous avons utilisé la carte de coûts locale (local costmap) et la planification de trajectoire locale (local path planning).

Toute les informations dont nous disposons, pour faire arriver le robot terrestre jusqu'à la personne, celles indiquées dans la figure 2. Dans cette figure, la zone circulaire noire autour du husky est ce qu'on appelle la carte de coûts lobale (local costmap). La carte de coûts utilise l'information prise par les capteurs pour créer une "occupancy grid" qui définit :

- Les endroits dans l'espace où il existe des obstacles (violet).
- Il existe un risque de collision selon l'orientation du robot (bleu foncé).
- Les zones libres d'obstacles (noir).
- Les zones bleu clair et gris sont pour les endroits inconnus.

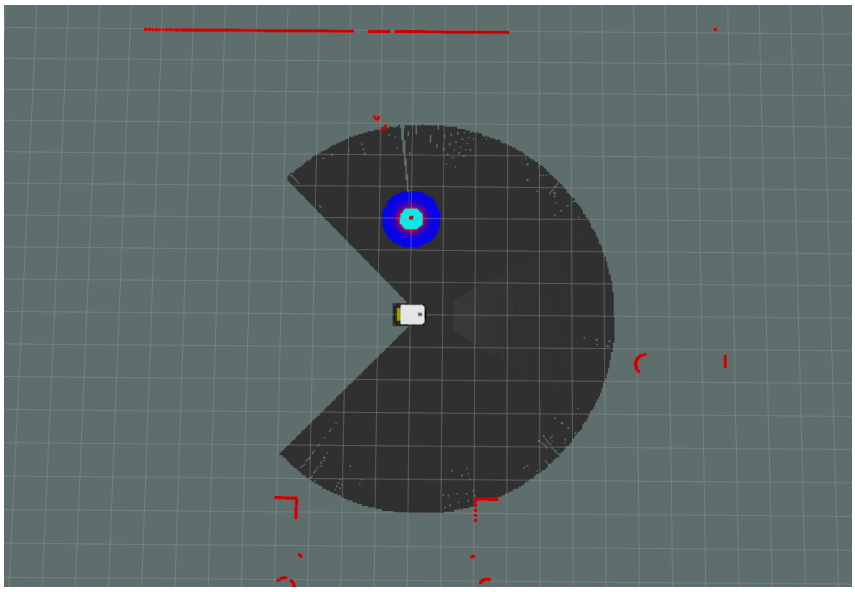


FIGURE 2 – La "costmap" locale qui définit la présence d'obstacles et les régions de collision

Une fois que la carte des coûts a été définie, l'algorithme de planification de chemin locale ("local path planning") utilise cette information pour calculer la trajectoire du robot en évitant les obstacles (la ligne verte dans la figure 3). La ligne en rouge devant le rover indique la direction que le robot doit prendre pour s'adapter à la trajectoire définie.

Dans ce cas particulier l'algorithme peut calculer directement le chemin qui permet d'éviter les obstacles. Néanmoins, si la position fourni au husky est suffisamment loin, l'algorithme calculera le chemin pour éviter les obstacles perçus, mais une fois en dehors de la carte locale, il dessinera une ligne droite jusqu'à la position cible. Au fur et à mesure que le robot avance, le logiciel calcule la nouvelle costmap et avec cette information il pourra recalculer le chemin localement. De cette façon, le robot terrestre arrivera à la position cible en évitant les obstacles.

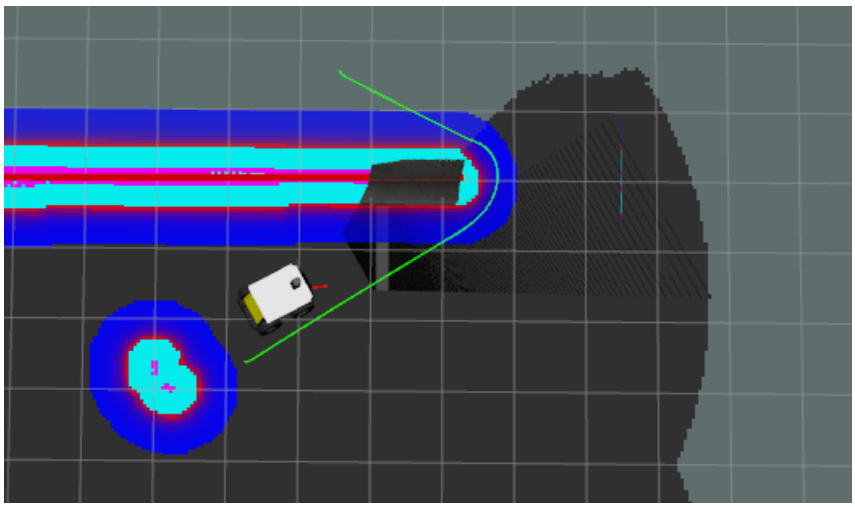


FIGURE 3 – Le en suivant le chemin local (vert) calculé pour éviter les obstacles

Il faut noter, que ce processus pourrait être fait d'un manière plus optimale si on disposait de la carte, car on pourrait calculer la trajectoire optimale du robot pour atteindre la cible sans s'écraser et en réduisant le temps au minimum.

4 Les drones (hector_quadrotor)

Dans la simulation nous disposons de dis drones dont le modèle est le "hector_quadrotor", montré dans la figure 4. Ce modèle a été équipé d'une caméra frontale de profondeur, d'une caméra pointant ver le bas, d'un capteur infrarouge pointant vers le bas et d'un "scan" laser qui permet de placer les obstacles autour du robot.

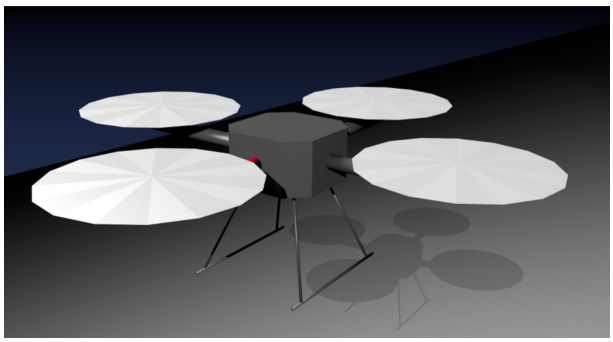


FIGURE 4 – Le modèle "hector_quadrotor" utilisé pour simuler le drone dans le challenge

Comme 10 drones ont été prévus pour la mission de détection et d'identification de personnes hostiles, ceux-ci ont été répartis en 3 groupes :

1. Reconnaissance de l'environnement et détection approximative des personnes (4 drones)
2. Précision de la position des personnes trouvées (4 drones)
3. Backup pour la reconnaissance, en cas de drones détruits (2 drones)

5 Stratégie pour le reconnaissance de l'environnement

Le schéma ci-dessous représente l'environnement de la simulation. La stratégie pour la reconnaissance de personnes est divisé en deux partie principales. La reconnaissance primaire qui a comme but estimer à peu près la position et la reconnaissance de précision qui va diminuer l'incertitude de l'estimation pour envoyer aux Huskys.

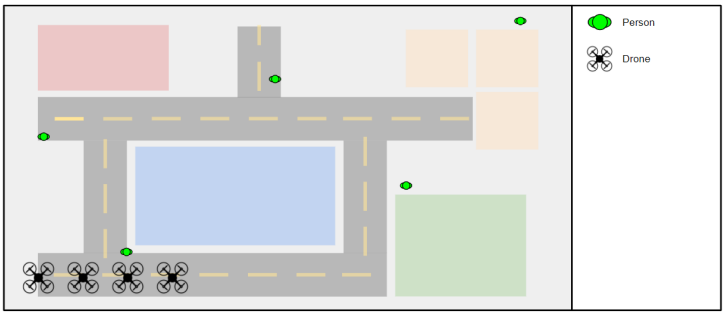


FIGURE 5 – Schéma de l'environnement

5.1 Reconnaissance Primaire

Pour la reconnaissance primaire, l'environnement est répartie en quatre zones et chacune des ces zones est attribué à un drone.Les drones vont parcourir sa respective zone de la façon représentée ci-dessus et utilisent des capteurs infra-rouge pour détecter les personnes hostiles. De cette façon, au final de la trajectoire on aura des premières positions estimés pour les personnes.

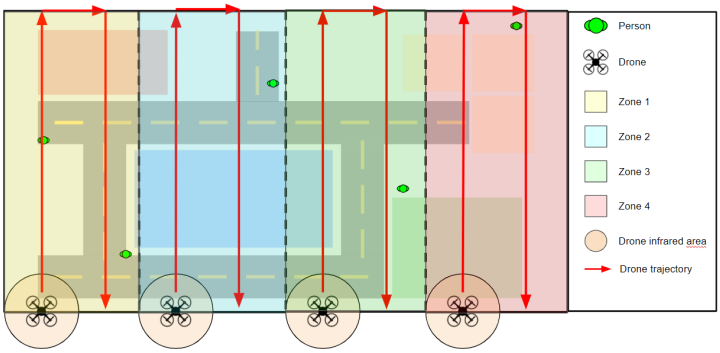


FIGURE 6 – Schéma de la stratégia pour la reconnaissance primaire

5.2 Reconnaissance de Précision

À partir des premières estimations de la position après la reconnaissance primaire, le drone responsable pour la reconnaissance de précision se dirige vers la position estimé par le reconnaissance primaire.

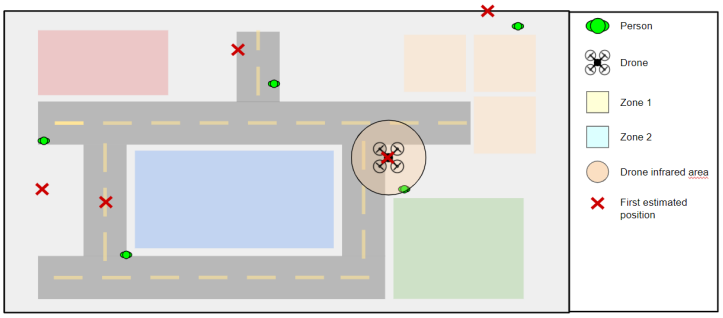


FIGURE 7 – Schéma de l'environnement

Ensuite, le drone responsable pour la reconnaissance de précision exécuté une trajectoire de recherche autour de la personne pour préciser son position. Cette trajectoire de recherche consiste dans le déplacement du drone dans les direction vertical et horizontal jusqu'à ce qu'il arrive plus à détecter la personne. De cette façon, il est possible de faire une triangulation pour estimer la position de la personne plus précisément.

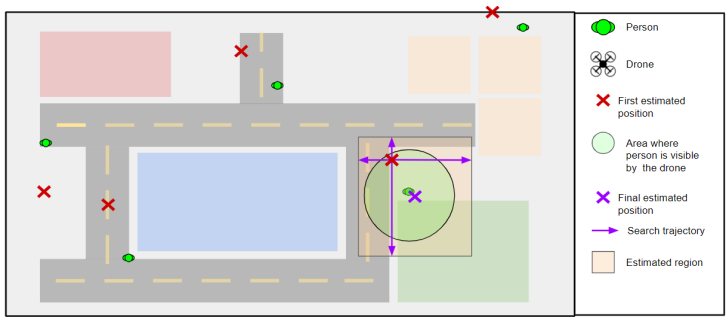


FIGURE 8 – Schéma de l'environnement

5.3 Résultats Détection Personnes

Dans la section suivante, on présente les résultats obtenus pour la détection des personnes et leur position pour le cas d'un environnement avec et sans killzone. Les killzones sont des zones présentes dans l'environnement où les drones son abattus et restent inutilisés pour le reste de la simulation.

5.3.1 Détection sans Killzone

	Hostile 1 (Drone 3)	Hostile 2 (Drone 4)	Hostile 3 (Drone 2)	Hostile 4 (Drone 3)	Hostile 4 (Drone 2)	Hostile 5	Hostile 6 (Drone 4)	Hostile 7 (Drone 4)	Hostile 8
Position réelle	X: -3.84 Y: -12.78	X: 21.72 Y: 41.11	X: -11.66 Y: 35.61	X: -7.40 Y: 33.55	X: -7.40 Y: 33.55	X: -33.12 Y: 50.56	X: 24.32 Y: -27.77	X: 30.82 Y: 18.74	X: 113.69 Y: 13.71
Position estimée	X: 0.30 Y: -20.31	X: 23.71 Y: 31.47	X: -19.85 Y: 27.56	X: -16.08 Y: 39.24	X: 0.88 Y: 24.49	Non trouvé	X: 20.37 Y: -36.82	X: 33.40 Y: 23.61	Non trouvé
Position précise	X: 0.30 Y: -13.31	X: 25.71 Y: 37.47	X: -11.85 Y: 32.56	X: -7.08 Y: 38.24	X: -4.11 Y: 33.49	Non trouvé	X: 24.37 Y: -27.82	X: 30.40 Y: 18.61	Non trouvé

FIGURE 9 – Schéma de l'environnement

5.3.2 Détection avec Killzone

	Hostile 1 (Drone 3)	Hostile 2 (Drone 4)	Hostile 3 (Drone 2)	Hostile 4 (Drone 3)	Hostile 4 (Drone 2)	Hostile 5	Hostile 6 (Drone 4)	Hostile 7 (Drone 4)	Hostile 8
Position réelle	X: -3.84 Y: -12.78	X: 21.72 Y: 41.11	X: -11.66 Y: 35.61	X: -7.40 Y: 33.55	X: -7.40 Y: 33.55	X: -33.12 Y: 50.56	X: 24.32 Y: -27.77	X: 30.82 Y: 18.74	X: 113.69 Y: 13.71
Position estimée	X: 1.28 Y: -27.06	X: 32.53 Y: 21.29	X: -20.40 Y: 28.41	X: 8.76 Y: 19.33	X: -17.64 Y: 40,31	Non trouvé	X: 18.35 Y: -45.69	X: 40.89 Y: 12.82	Non trouvé
Position précise	X: 3.72 Y: -13.06	X: 23.61 Y: 38.73	X: -11.40 Y: 35.41	X: -7.64 Y: 33.31	X: -5.92 Y: 31.47	Non trouvé	X: 24.35 Y: -27.69	X: 31.40 Y: 17.16	Non trouvé

FIGURE 10 – Schéma de l'environnement

5.3.3 Analyse des Résultats

Tout d'abord, on peut observer que la stratégie établie dépend de l'environnement. Cela est dû au fait que les drones n'utilisent pas de capteurs de profondeur pour éviter les obstacles. En outre, selon les positions de la Killzones, les résultats peuvent varier considérablement.

Néanmoins, pour l'environnement et les positions des Killzones considérés, les résultats finaux des positions des personnes hostiles étaient très proches des positions réelles. Cela valide la stratégie adopté de la reconnaissance primaire, suivi par la reconnaissance de précision.

Toutefois, la détermination des positions prends beaucoup de temps.

5.4 Stratégie pour la Killzone - non implémentée

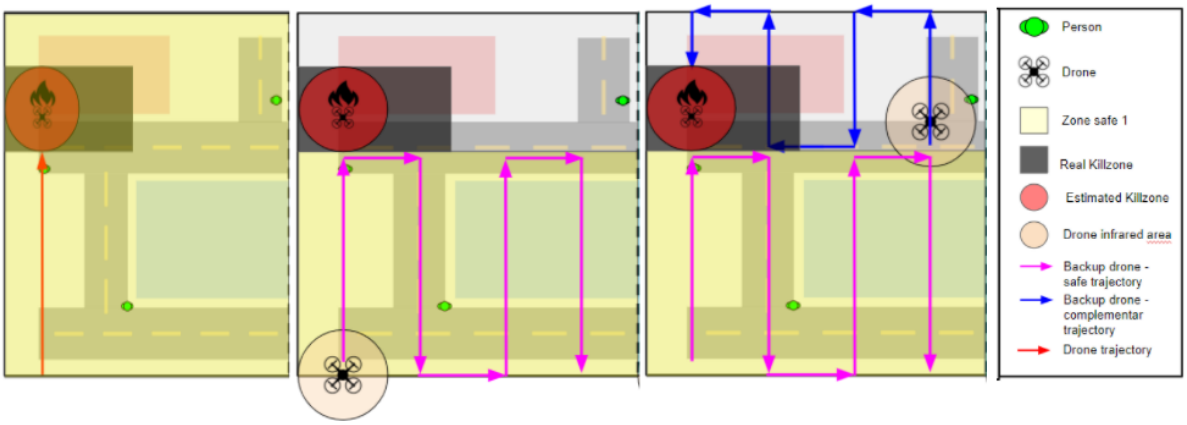


FIGURE 11 – Schéma de l'environnement

Dans le schéma ci-dessus on suppose qu'une drone a été abattu dans un killzone pendant son trajectoire de reconnaissance primaire. Une fois abattu, on sauvegarde la position où le drone a été abattu et on estime une zone que doit être évité.

Ensuite, on utilise le backup drone pour continuer la reconnaissance primaire. On calcule une nouvelle trajectoire pour éviter la killzone. Le drone va rechercher des personnes seulement dans la nouvelle zone safe 1.

Finalement, après avoir fait la recherche dans la zone safe 1. Le backup drone va exécuter la trajectoire complémentaire indiqué par le chemin bleu. Dans le cas montré, le drone serait abattu, parce que la killzone est plus large que l'estimation fait.

6 La communication

Dans la communication entre les drones et huskies, nous allons avoir une machine d'état qui fera la liaison entre ces deux interfaces. Pour y arriver à connecter, nous avons dessiné deux états principaux pour notre projet.

L'idée est avoir au moins une état d'exploration et une état de recherche. La figure 12 expose le fonctionnement de notre machine :

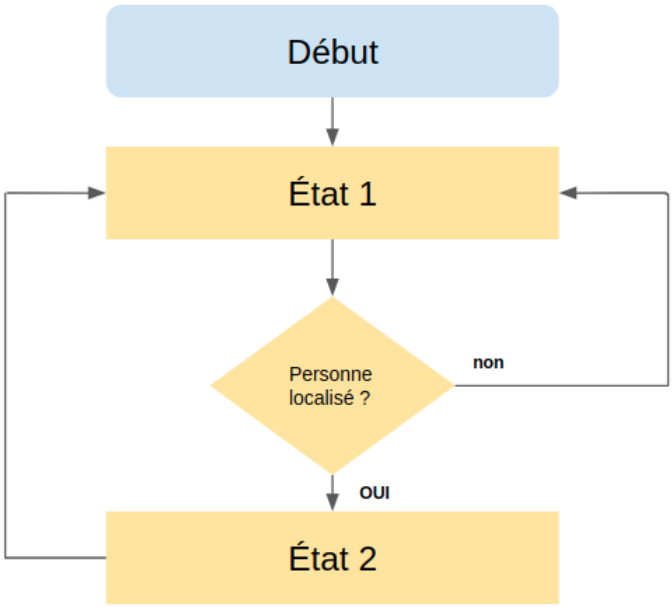


FIGURE 12 – Fonctionnement de la machine d'états

La première état va rester dans une espèce de "mode de surveillance" où le husky va vérifier à la fin de chaque boucle de notre machine, si les drones ont émis une vecteur qui continent la position d'une ou plusieurs personnes. D'autre côté, la responsabilité de cette état est toujours si déplacer en petites mouvementes aléatoires. Ultérieurement, l'idée c'était pour implémenter dans cette même état une fonction qui va utiliser des caméras disponibles sur le véhicule pour vérifier autour de son chemin.

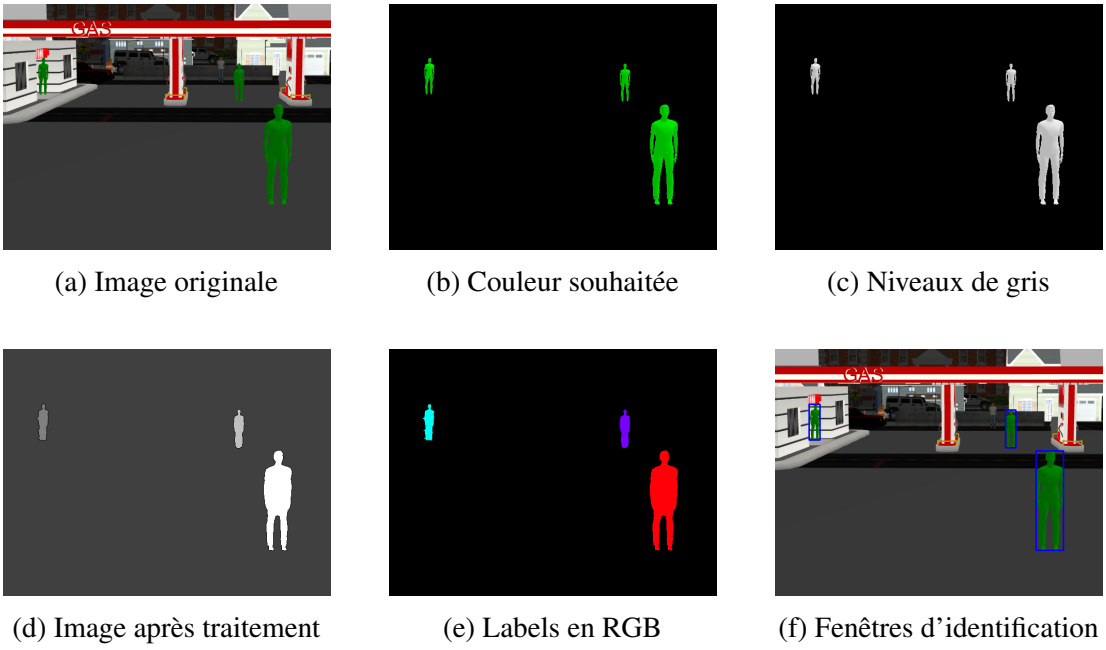
D'ailleurs, si une ou plusieurs personnes sont trouvé par les drones, nos huskies vont passer à l'état suivant. Dans cette état, le "Goal" attendu va être passé comme paramètre au déplacement du husky et il va arriver jusqu'à la cible souhaitée. Puis, en arrivant quelques mètres proche de la cible, nous avons bien identifié la personne.

7 Idées étudiées mais non réalisées

7.1 La reconnaissance d'images

Dans un premier temps, on a étudié la possibilité d'identifier des personnes hostiles dans l'environnement par la reconnaissance d'images obtenues par les caméras RGB des drones et des huskies. Comme les personnes ont une couleur spécifique (le vert) qui les distingue du paysage environnant, l'idée était de les retrouver dans l'image. Pour ce faire, une série de traitements et de transformations ont été appliqués aux images originales obtenues par les caméras jusqu'à ce que les personnes soient isolées et que leurs positions dans l'image soient extraites.

L'exemple suivant illustre le processus de traitement d'une image représentant la vue de la caméra avant d'un drone, jusqu'au point où les personnes présentes dans l'image sont encadrées par une fenêtre d'identification.



(a) Image originale (b) Couleur souhaitée (c) Niveaux de gris

(d) Image après traitement (e) Labels en RGB (f) Fenêtres d'identification

FIGURE 13 – Traitement d'image de la camera du drone

Tout d'abord, à travers l'image initiale (13a), il a été possible d'extraire les pixels verts représentant les personnes par un seuil (13b). L'image 13b a ensuite été convertie en niveaux de gris (13c), qui a elle-même été soumise à une série de transformations morphologiques jusqu'à l'application des algorithmes Connected Components et Watershed. Alors que les transformations morphologiques ont été employées afin d'extraire le bruit et d'améliorer les formes des nuages de pixels extraits de l'image originale, les deux algorithmes ont été utilisés pour la segmentation et l'identification des personnes présentes dans les scènes. L'image 13d représente la scène après l'emploi des deux algorithmes mentionnés. Après avoir identifié chaque personne dans la scène, des fenêtres ont été tracées sur chacune d'elles afin de trouver la position XY du centre de chaque personne dans l'image (13f).

7.1.1 Machine à états pour les drones

Après avoir défini la position des personnes dans l'image, une machine à états a été développée pour le drone afin d'estimer la position XYZ dans l'environnement. Cette machine d'état

effectuerait les opérations suivantes.

1. Stratégie de recherche de personnes
2. Une fois que le drone trouve une ou plusieurs personnes grâce à sa caméra, il tourne jusqu'à ce que la personne la plus proche (la plus grande fenêtre) soit au centre de l'image
3. Il s'élève ensuite jusqu'à une hauteur de 9 mètres (hauteur à laquelle il est possible de détecter des personnes avec le capteur infrarouge)
4. Avance linéairement jusqu'à ce que le capteur infrarouge détecte une personne
5. Il envoie sa position au robot terrestre

Ces états seront exécutés selon le diagramme d'état illustré par la figure suivante.

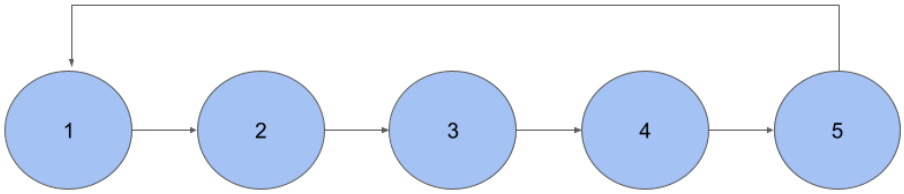


FIGURE 14 – Diagramme d'état pour le drone

Malheureusement, nous avons réalisé que cette méthode prendrait trop de temps et rencontrerait des difficultés avec des personnes complètement ou partiellement obstruées. Néanmoins, le code est utilisé pour la détection des personnes par le robot terrestre lorsqu'il se trouve à proximité d'une personne.

7.2 La cartographie

D'après ce qu'on a expliqué à la fin de la section du robot terrestre on sait que disposer d'une carte de l'endroit permet d'optimiser le déplacement des huskies. La carte nous permet aussi de calculer la position des différents robots et cibles avec une meilleur précision.

À la fin de cette section nous allons exposer les raisons pour lesquelles on n'a pas été capable de mettre en œuvre une méthodologie pour cartographier le terrain. Mais d'abord nous allons parler des pas que nous avons suivi pour étudier comment cartographier la simulation avec les drones.

D'abord, nous avons évalué le niveau de performance des paquets de ROS "hector_mapping" et "gmapping", mais avec le husky parce qu'il existait déjà des exemples plus adaptés à ce modèle. Dans un premier instant, nous avons essayé "gmapping" avec le robot terrestre et nous avons obtenu, d'une façon autonome, la carte montré dans la figure 15. Étant donné le bon résultat, nous avons décidé d'étudier leur implémentation sur les drones.

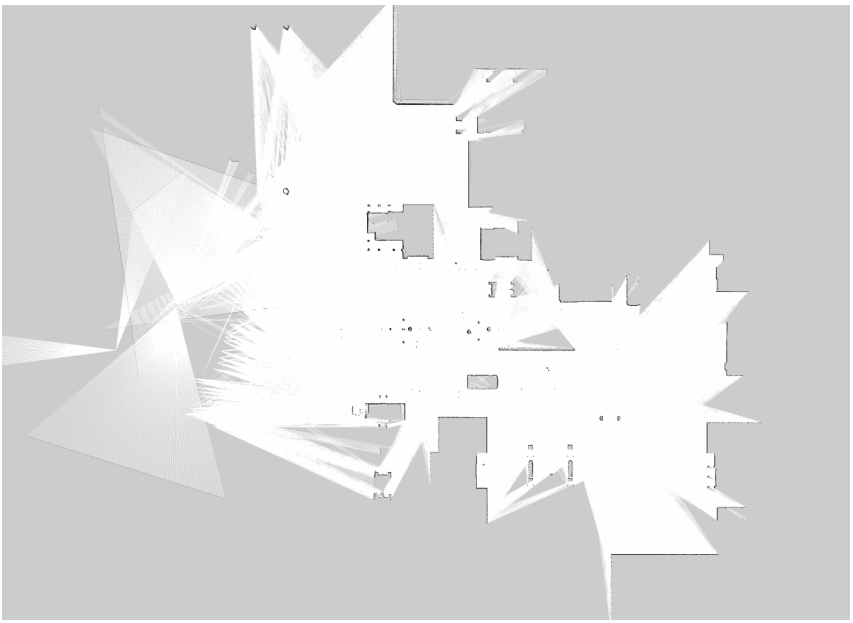


FIGURE 15 – La carte de la première simulation faite avec un husky de façon autonome

Finalement, nous avons utilisé le paquet “hector_mapping” avec le modèle du drone et on a adapté les codes existants pour le "husky" qui permettent la navigation dans la carte et l’évitement d’obstacles ("move_base" et "amcl"). Avec tout cela bien adapté a chaque modèle du drone, dans la figure 16 on observe une première dessin de la carte dessinée par un drone avant le décollage.

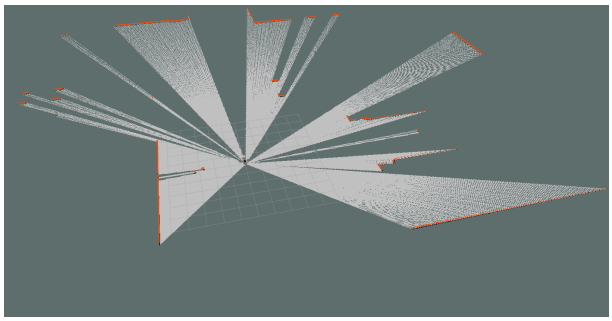


FIGURE 16 – Cartographie instantanée du drone sans avoir décollé

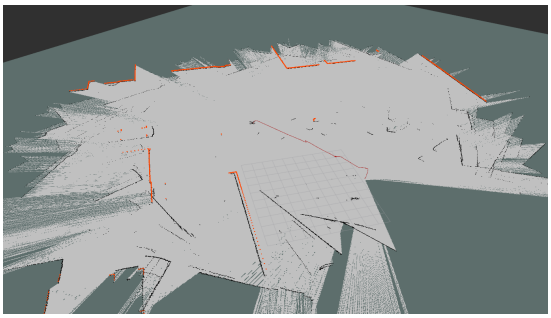


FIGURE 17 – Cartographie instantanée du drone après avoir décollé et après avoir bougé quelques mètres

Cependant, dans la figure 17, nous observons le résultat de la carte dessinée par un drone après avoir decollé et après avoir bougé quelques mètres dans le plan de vol. On observe comment le résultat est completement absurde. Cela arrive car la dynamique du drone lui fait s’incliner vers l’avant quand il avance et les capteurs, qui devraient pointer vers l’avant, pointent vers le bas. À cause de cela, l’algorithme interprète le sol comme un obstacle.

De plus, il faut noter qu’on a trouvé des problèmes pour coller les cartes créées par chaque drones. Nous avons essayé d’utiliser le paquet “multirobot_map_merge” mais nous avons pas réussi a le faire fonctionner proprement.

Finalement, le dernier problème est qu’il faut voler suffisamment bas pour que la carte soit valable pour le robot terrestre, ce qui incapacité le capteur d’infrarouge des drones jusqu’au moment où on a fini la tâche de cartographie.

8 Mise en oeuvre pratique

Après toutes les applications en environnement simulé, l'objectif était d'amener toutes les connaissances appliquées à l'environnement pratique, où il serait possible d'appliquer et de transformer notre projet en une application réelle. L'objectif de cette étape était de tester les codes d'analyse d'image, de mouvement et les machines d'état implémentées. Pour ces mises en œuvre, un drone Tello a été utilisé, ainsi que sa bibliothèque d'implémentation en python pour faciliter sa manipulation.

8.1 Analyse d'image

Tout d'abord, on a cherché à adapter les algorithmes utilisés dans la simulation pour qu'ils fonctionnent dans un environnement pratique avec des interférences externes telles que la clarté, les délais, la qualité de l'image, etc. Ensuite, pour définir les cibles, nous avons continué à placer des rectangles autour de la cible souhaitée, et nous avons à nouveau essayé de filtrer les cibles souhaitées par la couleur en question, étant à nouveau une nuance de vert la couleur choisie.

Pour obtenir une bonne détection, les problèmes liés à la clarté ont été affectés, et pour obtenir une bonne forme de l'objet, plusieurs types de traitement ont été utilisés pour obtenir une bonne évaluation. Pour atteindre le suivi, les mêmes procédures que celles présentées pour l'environnement de simulation ont été utilisées. Dans la Figure 18, est exposé le résultat obtenu pour cette partie de l'implémentation

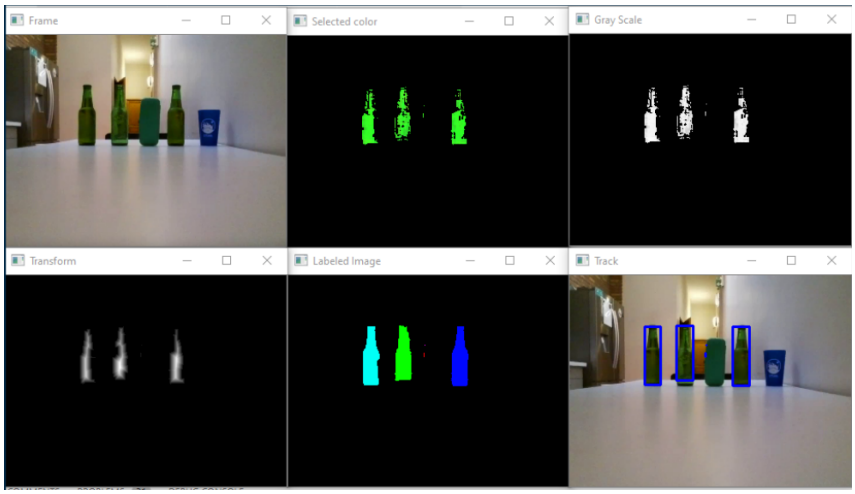


FIGURE 18 – Images acquises par le drone

8.2 Mouvement et Machine d'état

Cette étape de mouvement et l'étape de la machine d'état n'ont finalement pas été mises en œuvre en raison de problèmes sur le parcours de test. Pour l'étape de mouvement, nous avons essayé d'implémenter la routine de mouvements adoptée par les drones de reconnaissance de l'environnement virtuel. Cependant, des erreurs de manipulation de la bibliothèque propre au drone, ajoutées au peu de temps pour cette implémentation finale, ont entravé les avancées de cette partie.

Des problèmes tels que le rendu de l'image en même temps que l'exécution des mouvements, qui n'ont pas été effectués en même temps, et la lenteur des tests due à des charges de batterie courtes, ont retardé les applications. Et pour appliquer correctement la machine à états ordonnant

ces deux parties correspondant au mouvement et au rendu de l'image, il n'y avait pas assez d'états pour sa composition, elle n'a donc pas été appliquée à la fin

9 Travail ultérieur

À partir du code que nous avons développé il existe des améliorations qui peuvent être mises en œuvre directement, sans avoir besoin de nouveaux paquets de ROS. Avec un peu plus du temps, on aurait été capables de le faire nous mêmes.

- Pour la machine d'états du husky, on pourrait ajouter reconnaissance d'images développée pour s'approcher à la personne et concevoir un nouvel état (dans la machine à états du husky) qui lui permet de faire cela quand il est suffisamment proche de la position fournie par le drone.
- Pour les drones, on pourrait définir une stratégie de réorganisation pour la flotte de drones quand on perd un ou plusieurs drones dans une zone hostile.
- La communication entre le drone et le husky peut être amélioré, pour avoir une liste globale de cibles qui nous permet de :

  Éviter que deux huskies aillent vers la même personne.

  Éviter qu'un husky aille deux fois vers la même personne.

  Envoyer le husky le plus proche de la cible ou celui qui n'a pas de tâches en cours.

10 Conclusion

Pour les applications de niveau théorique dans un environnement simulé, même s'il y avait des problèmes différents et un niveau élevé de complexité, l'objectif de l'application des concepts ROS pour la bonne communication entre les deux parties était de pouvoir rassembler dans un topique plusieurs informations qui comprenaient des informations précises sur les positions ou simplement des informations sur la gestion des états ; Malgré quelques problèmes liés au mouvement des robots terrestres, le nombre de personnes détectées a été important, ce qui a permis d'obtenir un bon résultat final de notre algorithme et, malgré le retard pris dans l'exécution des tâches en raison de la stratégie adoptée, la précision des positions est toujours bonne.

Cependant, en ce qui concerne la partie pratique, en raison du peu de temps pour les tests en situation réelle, et des difficultés de manipulation avec les drones, le résultat n'a été que la mise en œuvre de techniques visuelles utilisées tout au long du défi, mais qui ont tout de même obtenu de bons résultats dans ce défi. Compte tenu de la section précédente, ces améliorations prévues contribueraient grandement à l'avancement positif et à l'obtention de meilleurs résultats par rapport à ceux présentés jusqu'à présent.