


Université de Montpellier
Master 2 EEA
Spécialité Robotique

Rapport de stage

Implémentation d'un algorithme de fusion de capteurs pour des robots mobiles sous-marins ou terrestres

Réalisé par :
Albane Ordrenneau

Encadré par :
Nicolas Boizot

Année 2017-2018

Remerciements

Je tiens à remercier Monsieur Nicolas BOIZOT, maître de conférence au sein du laboratoire LIS, mon tuteur de stage, qui m'a suivi tout au long de cette période et m'a conseillé sur l'orientation que celui-ci devait prendre.

Je remercie également Monsieur Éric BUSVELLE, professeur au sein du laboratoire LIS et Monsieur Vincent HUGEL, professeur au sein du laboratoire COSMER, pour leurs soutiens, pour avoir proposé ce stage dans le cadre de mon master et de m'avoir accepté pour travailler sur ce sujet.

Je voudrais également remercier Madame Aïda FEDDAOUI, doctorante en codirection dans les laboratoires LIS et COSMER, pour son soutien et pour avoir été au cœur de ce stage.

Je remercie aussi Madame Claire DUNE, maître de conférence au sein du laboratoire COSMER, qui a été présente et disponible tout au long de ce stage.

Et je remercie Adrien MOJIKA, stagiaire, de poursuivre ce stage ainsi que pour son enthousiasme et sa bonne humeur.

Par ailleurs je remercie Madame Élisabeth MURISASCO, responsable du laboratoire LIS, ainsi que Madame Adoration DI SANTI, gestionnaire du laboratoire LIS, pour m'avoir accueillie et me permettre d'effectuer mon stage.

Je n'oublierais pas de remercier l'ensemble des membres des laboratoires LIS et COSMER pour leur sympathie et leur disponibilité.

Sommaire

Table des Illustrations

Résumé

Ce document présente les différents travaux réalisés afin de mettre en place l'utilisation de plusieurs robots sous-marins et terrestres. Ces robots contribueront à tester l'algorithme de fusion de données asynchrone définit dans les travaux de Mme A. Feddaoui, doctorante au sein du laboratoire LIS, en condition réelles.

1. Introduction

Ce stage résulte de la collaboration de deux laboratoires de l'université de Toulon [1], d'une part le laboratoire LIS [2] qui a initié ce projet et mis en place la partie théorique et d'autre part le laboratoire COSMER [3] qui a fourni le matériel et les connaissances techniques.

Le laboratoire du LIS (Laboratoire d'Informatique et Systèmes) résulte de la fusion entre le laboratoire LSIS (Laboratoire des Sciences de l'Information et des Systèmes) de Toulon et le laboratoire le LIF (Laboratoire d'Informatique Fondamentale) de Marseille. C'est une Unité Mixte de Recherche sous tutelle du CNRS (Centre National de la Recherche Scientifique) rattachée à l'INS2I (Institut des sciences de l'information et de leurs interactions), à l'Université d'Aix-Marseille et à l'Université de Toulon. Le LIS du site de Toulon est divisé en trois pôles: DYNI (DYNamiques de l'Information), CDE (Contrôle & Diagnostic pour l'Environnement) et SIIM (SIgnal et IMages). C'est dans le pôle CDE que se déroule ce stage.

Jeune laboratoire créé fin 2014, le Laboratoire COSMER (Conception de Systèmes Mécaniques et Robotiques) se développe essentiellement sur la robotique, notamment sous-marine, la vision et la modélisation 3D. Cette équipe réunit des compétences en recherche et en enseignement dans les domaines de la mécanique et de la robotique.

Le but du ce stage est de tester et d'implémenter un algorithme de fusion de données asynchrones, mis au point par la doctorante Mme A. Feddaoui, sur un robot afin de vérifier et d'approuver le bon fonctionnement de cette nouvelle méthode [4]. Il faut également mettre en place et simplifier l'utilisation des différents robots dans le cas où il y ait besoin de faire d'autres tests par la suite. De plus, l'université de Toulon ouvrant un nouveau parcours Robotique à la rentrée 2018, essentiellement enseignée par les membres des laboratoires COSMER et LIS, il faudra rendre les différents robots le plus facilement maniable et modifiable en vue de leur utilisation dans différents travaux pratiques.

2. L'environnement de travail

Cette section présente les différents robots utilisés ainsi que le cadre dans lequel s'est déroulé ce stage. Il décrit aussi très rapidement les travaux réalisés par Mme A. Feddaoui au cours de sa thèse.

2.1. Lieux du stage

Comme ce stage comprend plusieurs encadrants appartenant à différents domaines et laboratoires, il s'est déroulé sur 4 lieux différents.

Le premier est situé dans les locaux du LIS (Figure 1-1). C'est dans ce lieu que tout ce qui est du domaine de la recherche a été étudié de par la proximité des enseignants chercheurs spécialisé dans le domaine du contrôle/commande.

Le second lieu est situé dans les locaux de COSMER (Figure 1-2), c'est là que se trouvent les robots. C'est donc là qu'ont été effectués les tests et les expérimentations sur les robots terrestres ainsi que toute la partie maintenance et montage des BlueRovs.

Le troisième lieu est le bassin de la faculté (Figure 1-3). C'est un bassin de 8*4m et d'une profondeur d'1m50. C'est là que toute la mise en place, la calibration et les expériences sur les BlueRovs ont été effectués.

Enfin, le dernier lieu est dans les locaux du centre de recherche IFREMER (Institut Français Recherche Exploitation Mer [5]) qui ne se situe pas dans les locaux de la faculté de Toulon car il n'y est pas rattaché, il est indépendant (Figure 1-4). Le nouveau Mini-Rov étant arrivé dans ces locaux, il nous faut nous déplacer en attendant que les documents pour le rapatriement du robot soient validés.

Figure 1 : Localisation des 4 lieux du stage

2.2. Les différents robots utilisés

Le laboratoire COSMER possède de nombreux robots : terrestres, sous-marins ou humanoïdes qui sont utilisés pour la recherche ou dans le cadre des travaux pratiques des étudiants. Cette section décrit ceux utilisés dans le cadre de ce stage.

2.2.1. Description des Turtlebots

Les Turtlebots sont des robots mobiles de type unicycle. Nous utilisons deux modèles de ces robots : le Turtlebot2 et le Turtlebot3 Waffle.

Le premier (Figure 2 - gauche) possède une caméra kinect ainsi qu'une IMU ('*Inertial Measurement Unit'*, centrale à inertie) et des pare-chocs au niveau des roues. Il est également branché sur secteur ce qui permet de tester plus facilement les codes de fonctionnement du robot car il n'y a pas de problème de perte de communication.

Le second (Figure 2 - droite) possède un capteur plus adapté au projet, un Lidar. Il possède également la caméra et L'IMU mais pas les pare-chocs. L'inconvénient de Turtlebot3 Waffle est qu'il n'est alimenté que par des batteries d'une faible autonomie (environ une heure) ce qui pose problème lors de l'élaboration du code et des différents tests. En effet, lorsque la batterie est presque vide, la communication avec le robot s'arrête et celui-ci continue d'effectuer le dernier ordre reçu sans que l'on puisse l'arrêter.

Les deux robots possèdent un système d'exploitation Ubuntu avec ROS, sur un ordinateur connecté au robot pour le premier et sur une carte arduino pour le second.

Figure 2 : Turtlebot2 (à gauche) et Turtlebot3 (à droite)

2.2.2. Description des BlueRovs

Les BlueRovs sont des robots mobiles sous-marin de type ROV (*Remotely Operated Vehicle)* connectés à un ordinateur via un câble Ethernet pour communiquer

avec la partie commande du robot. Certains, comme le Mini-Rov introduit dans 2.2.3, sont également connectés à la partie puissance qui se trouve hors de l'eau.

Il y a trois Rovs qui seront décrit ici, les BlueRovs 1 et 2 que possède COSMER ainsi que le nouveau Mini-Rov acquis lors de ce stage par un co-financement avec le centre de recherche IFREMER.

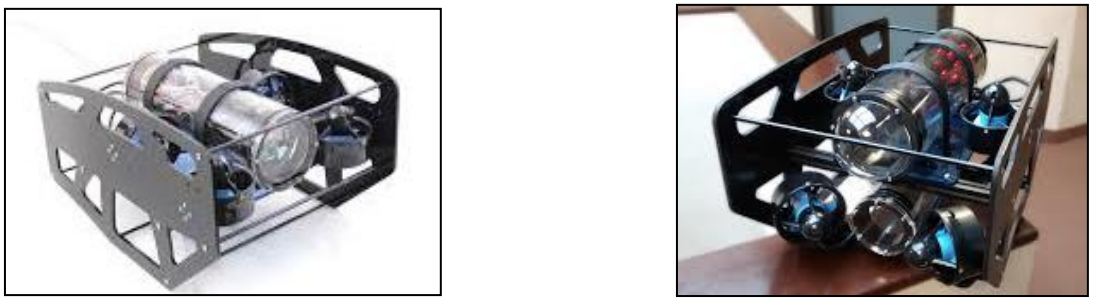

Figure 3 : BlueRov1 (à gauche) et BlueRov2 (à droite)

Les BlueRovs 1 et 2 (Figure 3) ont été assemblés et câblés par COSMER. Ils sont équipés de capteurs de profondeur absolue, d'une IMU et d'une caméra et sont alimentés grâce à des batteries LiPo d'une autonomie d'une heure. La principale différence entre ces deux robots est la position et l'orientation de leurs propulseurs. Cette différence (Figure 4) ne permet pas au BlueRov2 d'effectuer de tangage car les propulseurs 5 et 6 sont coaxiaux.

Le BlueRov2 a également deux compartiments étanches : un pour la partie commande et l'autre ne contenant que la batterie. Le BlueRov1 ne possède qu'un compartiment étanche dans lequel on retrouve les mêmes composants que le BlueRov2.

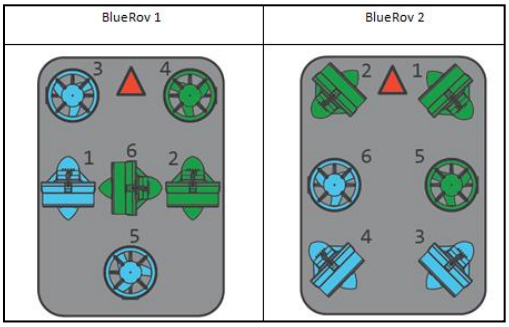

Figure 4 : Vue du dessus des différences de positions et d'orientation des propulseurs.

2.2.3. Description du Mini-Rov

Le Mini-Rov (Figure 5) acheté à la société SubSeaTech [6] par le laboratoire COSMER et le centre de recherche IFREMER possède deux caméras analogiques (avant et arrière), un sonar, une IMU, un capteur de profondeur relatif, un USBL pour la géolocalisation et un capteur DVL pour mesurer sa vitesse instantanée.

Figure 5 : Mini-Rov

2.3. Les logiciels et langages utilisés

Le langage utilisé au cours de ce stage est le langage Python (Figure 6-1) afin de rester cohérent avec les autres programmes déjà existants et le programme scolaire des étudiants qui pourraient travailler sur ces robots.

Afin de pouvoir communiquer avec les robots, d'acquérir les données capteurs et de pouvoir envoyer des commandes aux différents robots, on utilise les outils de ROS ('*Robot Operating System*'). Actuellement, la version recommandée pour ROS est ROS Kinetic (Figure 6-2). Les librairies ROS imposent de travailler sous un système d'exploitation Linux, c'est pourquoi nous travaillons avec la dernière version d'Ubuntu, la version 16.4.

La politique du laboratoire COSMER a été d'utiliser l'environnement ROS pour l'ensemble de ses robots, terrestres ou sous-marins. Ainsi, bien que l'algorithme de fusion de données à tester ait été pensé pour un robot sous-marins, il a été possible de le tester sur des robots terrestres (les Turtlebots), les tests et mises au point étant beaucoup plus facile que dans un environnement aquatique (très contraignant).

Les outils ROS permettent de créer des liens entre les capteurs, les actionneurs et le programme. Dans le programme, on crée des « *Subscribers* » qui vont s'abonner au capteur et recevoir leurs informations, on crée également des « *Publishers* » qui eux vont publier des informations aux actionneurs.

L'environnement de simulation Gazebo [7], compatible avec ROS, permet de modeliser et de tester le comportement de robot dans un environnement (Figure 6-3). Gazebo m'a permis de faire un premier test de mes programmes avant de passer sous les robots réels. Mais il permet également de nombreux autres usages car il possède de nombreuses structures préenregistrées et il permet également de créer de nouveaux milieux ou robots. Il suffit de lancer Gazebo avant de lancer son programme et de configurer le PC pour communiquer en local pour tester le même programme que celui qui sera implanté sur la machine. Le milieu par défaut de Gazebo est un espace vide. On peut y ajouter des meubles et des véhicules à notre convenance.

Pour porter dans un environnement temps réel le code existant de Mme A.Feddaoui (voir Section 2.4) il sera aussi necessaire d'utilisé Matlab (Figure 6-4) car c'est sur cette plateforme qu'une première simulation de son algorithme a été effectué.

Figure 6 : Logo de logiciels et langages utilisés (Python - ROS Kinetic – Gazebo - Matlab)

2.4. La thèse de Mme A. Feddaoui

Mme A. Feddaoui avait effectué ses études dans le domaine des mathématiques, ce qui implique qu'elle n'avait pas de compétences et d'antécédents en robotique. Mme A. Feddaoui a commencé son doctorat fin 2016 sur le sujet : "Observateurs non linéaires pour les systèmes à mesures asynchrones ou datations incertaines : application robotique sous-marine dédiée à l'asservissement référencé multi-capteurs". La preuve de convergence du filtre de Kalman est connue pour les systèmes continus et les systèmes continus discrets à mesures synchrones. Elle fait cependant défaut lorsque les mesures arrivent de manière asynchrone et on lui préfère souvent dans la littérature des observateurs de type Luenberger, dont la convergence est plus simple à prouver mais qui est plus sensible au bruit présent dans le système.

L'objectif est de proposer un formalisme adapté aux systèmes dynamiques linéaires dont les mesures sont échantillonnées à différentes fréquences ainsi que des éléments de preuve de la convergence de ce filtre. Les travaux récents de Mme A. Feddaoui ont donc consistés à prouver la convergence des filtres de Kalman dans le cadre de la théorie du grand gain et pour des systèmes non-linéaires avec des mesures asynchrones.

Ce stage permet donc de valider ses travaux avec des valeurs réelles, non-simulés et d'implémenter directement l'observateur dans la boucle de fonctionnement du robot. Les simulations qu'a déjà effectuées Mme A. Feddaoui ont été réalisées sous Matlab. Dans le but d'implémenter directement l'observateur dans le programme du robot, il est nécessaire de convertir les travaux de Mme A. Feddaoui en langage Python.

Dans un premier temps, le programme implémentant les travaux effectués par Mme A. Feddaoui a été traduit en python, de ce fait, nous pouvions comparer les résultats obtenus avec ceux existant sous Matlab pour valider la traduction. Par la suite, la partie observateur sera isolée et pourra être intégrée dans le programme de commande d'un des robots.

La simulation se passe comme suit, on suppose que deux balises (ou amers) sont installées à des emplacements connus (Figure 7). Les balises sont détectées

lorsque le capteur point vers elles. Si la première (l'amer A) est détectées, le capteur mesure l'angle et la distance de cette balise, si c'est la seconde (l'amer B) qui est détectée, il ne retient que l'information de l'angle (c'est un choix prit pour montrer les performances de l'algorithme).

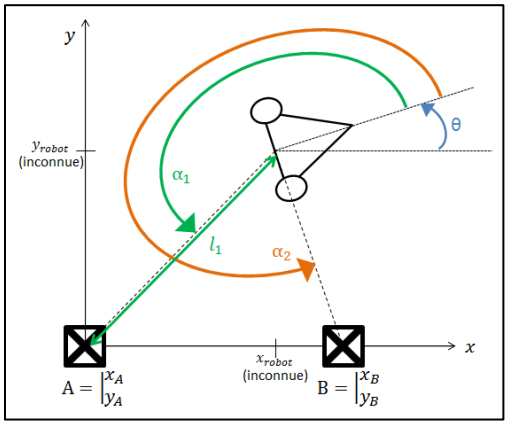

Figure 7 : Représentation de la simulation de Mme A.Feddaoui

Les informations reçues arrivent à des temps $\boldsymbol{\tau}_k$, le temps depuis la dernière fois où l'information de la balise A ou B est noté DT[1ou2]. On note z et S les paramètres du filtre de Kalman, z^- et S^- sont les valeurs avant correction et z^+ et S^+ sont les valeurs après correction. Les équations de l'observateurs sont définit comme dans l'équation (1).

$$\begin{cases} \dot{\boldsymbol{z}}(\boldsymbol{\tau}) = \boldsymbol{A}(\boldsymbol{u})\boldsymbol{z}(\boldsymbol{\tau}) + \mathbf{b}(\mathbf{z},\mathbf{u}) \\ \dot{\boldsymbol{S}}(\boldsymbol{\tau}) = -\big(\boldsymbol{A}(\boldsymbol{u}) + \boldsymbol{Db}(\boldsymbol{z},\boldsymbol{u})\big)'\boldsymbol{S}(\boldsymbol{\tau}) - \mathbf{S}(\boldsymbol{\tau})\big(\boldsymbol{A}(\boldsymbol{u}) + \boldsymbol{Db}(\boldsymbol{z},\boldsymbol{u})\big) - (\boldsymbol{SQS})(\boldsymbol{\tau}) \end{cases} \quad (1)$$

On définit également les équations de correction à une itération k comme dans l'équation (2).

$$\begin{cases} \boldsymbol{z_k^+} = \boldsymbol{z_k^-} - \boldsymbol{S_k^{+-1}}\boldsymbol{C'R^{-1}}(\boldsymbol{C} * \boldsymbol{z_k^-} - \boldsymbol{y_k}) * \boldsymbol{DT} \\ \boldsymbol{S_k^+} = \boldsymbol{S_k^-} + \boldsymbol{C'R^{-1}C} * \boldsymbol{DT} \end{cases} \quad (2)$$

Avec A(u) et b(z,u) et C les matrices de représentation du système et Db(z,u) la matrice Jacobienne de b(z,u). Les matrices R et Q sont définies positives. Ces équations sont à quelques variations près, ce sont les équations classiques du filtre de Kalman étendu. Les différences concernent l'introduction d'un paramètre de grand gain, le changement de coordonnées du système et l'utilisation de la variable DT dans les équations pour assurer la convergence du filtre (voir [4]).

3. Les travaux effectués

Avant de commencer à travailler sur les robots, j'ai étudié comment fonctionnait les outils ROS, d'une part en regardant quelques tutoriels sur internet et d'autre part en lisant « *A Gentle Introduction to ROS* » [8]. L'inconvénient de ce livre est qu'il explique comment utilisé ROS en C++ et il se trouve que la création et l'appel aux « *Subscriber* » et aux « *Publisher* » diffère grandement entre les langages Python et C++.

Afin de garantir le bon fonctionnement de tous les robots, il faut suivre un protocole précis pour les allumer ou les éteindre. Pour augmenter la longévité des batteries, il est nécessaire de ne pas les vider complètement et donc de chronométrer leur temps d'utilisation. De plus, afin que les systèmes d'exploitation installés sur les robots ne défaillent pas, il est impératif d'éteindre les robots avant de leur enlever leur source d'énergie. J'ai répertorié ces protocoles dans des modes d'emplois. Ces modes d'emploi expliquent également pas à pas l'installation des logiciels ainsi que l'utilisation des robots, les problèmes récurrents que j'ai rencontré et comment résoudre ces problèmes.

3.1. Sur le Turtlebot2

J'ai commencé à tester le fonctionnement de ROS et du Turtlebot2 en utilisant Gazebo. Pour tester les déplacements du robot, j'ai opté pour un espace fermé comprenant des obstacles en hauteur pour le lidar et des obstacles au sol, qui ne seront pas détectés par le lidar afin de tester le fonctionnement des parechocs. Dans la liste des objets déjà existants j'ai donc choisi des armoires et des morceaux de parpaings, j'ai fermé l'obstacle grâce à des murs mis bout à bout.

Le programme qui décrit le comportement des Turtlebots est structuré sous la forme d'un FSM (« *Finite State Machine* » - Machine à états fini) comprenant des suites d'actions séparées par des transitions. L'algorithme permettant de faire fonctionner le Turtlebot2 contient deux gROSses parties : le mode automatique et le mode manuel.

Dans le mode manuel, le robot est commandé grâce aux joysticks de la manette (Figure 8) connectée à l'ordinateur. Le joystick gauche commande l'avancée (vers le haut pour l'avancée positive et vers le bas pour l'avancée négative : recul) tandis que le joystick droit commande la rotation propre du robot (vers la gauche pour le sens antihoraire et vers la droite pour le sens horaire). Ces deux commandes étant dissociées, il est possible de les combiner. Le mode manuel n'est constitué que d'une action car il n'a qu'une fonction : transformer les données des joysticks en vitesse et les envoyer au robot.

Figure 8 : Manette utilisée pour le Turtlebot2

Le bouton triangle de la manette, permet de passer du mode manuel au mode automatique, les autres boutons ne sont pas utilisés pour ce robot mais peuvent être utilisés pour d'autre transition.

Le mode automatique est composé de plusieurs états. L'état de base envoie uniquement des données aux moteurs pour que le robot avance en ligne droite, mais lorsque les différents capteurs détectent un obstacle, l'état change. Dans le cas où ce sont les pare-chocs qui captent un obstacle, le robot va reculer puis tourner en fonction de l'endroit où il a détecté l'obstacle (si le pare-chocs droit détecte un obstacle, le robot tourne dans le sens antihoraire et vis-versa) puis il s'arrêtera et retournera dans l'état de base du mode automatique.

Dans le cas où la Kinect détecte un obstacle qui empêche le Turtlebot2 de continuer tout droit, le programme va calculer s'il a un espace assez grand pour que le robot passe. En fonction de la zone où se trouve cet espace, la vitesse des roues sera modifiée pour se déplacer vers l'espace libre. Dans le cas où il n'y a pas d'espace assez grand pour que le robot puisse passer, il va se mettre à tourner sur lui-même jusqu'à détecter un espace libre. Une fois qu'il peut continuer tout droit, il retourne dans l'état de base du mode automatique.

Pour retourner au mode manuel, il suffit d'actionner un des deux joysticks.

Afin de garantir la sécurité du robot et des utilisateurs, il est possible de retourner au mode manuel à n'importe quel moment en actionnant un des joysticks. On retourne également au mode manuel lorsque les roues du robot ne touchent plus le sol. Le fonctionnement de cette machine à état fini est décrit Figure 9.

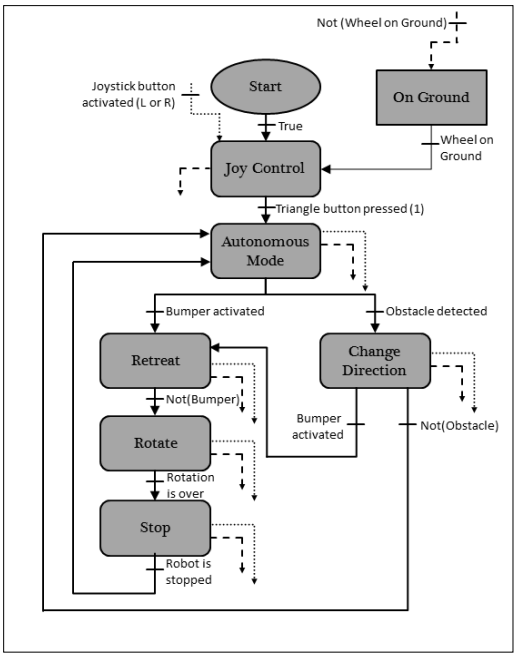

Figure 9 : Fonctionnement du Turtlebot2

3.2. Sur les BlueRovs

Les robots BlueRovs sont des robots sous-marins qu'il faut monter soi-même, des tutoriels sont disponibles sur internet [9]. Les câblages ont été effectués par des étudiants de SeaTech, l'école d'ingénieur qui occupe le même bâtiment. En réalisant ces câblages, ils n'ont pas pensé que les câbles et les composants allaient être enfermés dans un tube car certains câbles et fixations ne permettaient pas de fermer ce tube, c'est pourquoi j'ai rebranché et démêlé les câbles afin que ceux-ci rentrent plus facilement dans le tube et de faciliter le repérage des câbles en cas de maintenance. De plus, les bouchons d'étanchéités n'ont pas été installés correctement, certains n'avaient pas de joints d'étanchéité et d'autre n'avaient même pas été serrés.

Une fois les branchements effectués, il était temps de tester hors de l'eau que tous les composants fonctionnent correctement. Dans un premier temps, il faut connecter le robot à l'ordinateur. Etant un ROV, la connexion se fait grâce à un câble Ethernet, il suffit donc de créer une nouvelle connexion filaire. Au préalable, une version de Ubuntu et de ROS ont été installées sur la micro carte SD de la Raspberry du BlueRov.

Il suffit maintenant de lancer les capteurs et de vérifier qu'ils renvoient des informations et que celles-ci soient cohérentes. Par exemple, en observant l'IMU, les données du magnétomètre doivent varier lorsque l'on bouge le robot ou concernant le capteur de pression, la pression devait augmenter lorsque l'on appuyait dessus, ce qui

augmente la pression mesurée. Afin de simplifier la suite, il a été ajouté que ces capteurs se mettraient également en marche au démarrage du robot.

Ensuite, il faut vérifier que les moteurs fonctionnent en utilisant une commande qui permet d'effectuer les six déplacements primaires (les trois translations et les trois rotations : roulis, tangage, lacet).

Hors, lorsque le robot devait avancer, ce n'était pas les moteurs correspondants qui se mettaient en route. Pour régler ce problème, dans un premier temps, les problèmes physiques ont été éliminés en vérifiant que tous les moteurs étaient branchés dans le même sens et que chaque moteur était branché au bon port du contrôleur pixhawk. Puis les paramètres du contrôleur ont été vérifiés, chaque moteur devait tourner dans le même sens car ce sont les pales des propulseurs qui sont installées dans des sens différents (représenté par les différentes couleurs des propulseurs sur la Figure 4). Puis la matrice des gains des moteurs a été vérifiée. C'est une matrice qui définit quels moteurs sont actionnés en fonction du déplacement voulu. Les moteurs qui devaient fonctionner n'étaient pas les mêmes que ceux qui fonctionnaient réellement (par exemple, si le BlueRov1 doit avancer les propulseurs 1 et 2 doivent fonctionner, ce qui n'était pas le cas, les propulseurs 1 et 4 fonctionnaient) donc un nouveau mode a été créé, au lieu de faire fonctionner les moteurs en fonction des déplacements, ce mode permet de faire fonctionner les moteurs un à un.

Le robot a été redémarré avec ces nouveaux paramètres et il a été observé que les moteurs indiqués ne correspondaient pas, pour le BlueRov1 les propulseurs 3 et 4 étaient inversés comme indiqué Figure 10. La matrice de gain pour les déplacements a donc été réécrite en fonction des vrais emplacements des moteurs sans déplacer physiquement les moteurs. On suppose qu'il y a eu une erreur dans la programmation de base de cette matrice de gain ou dans le référencement des propulseurs sur la notice de montage des robots.

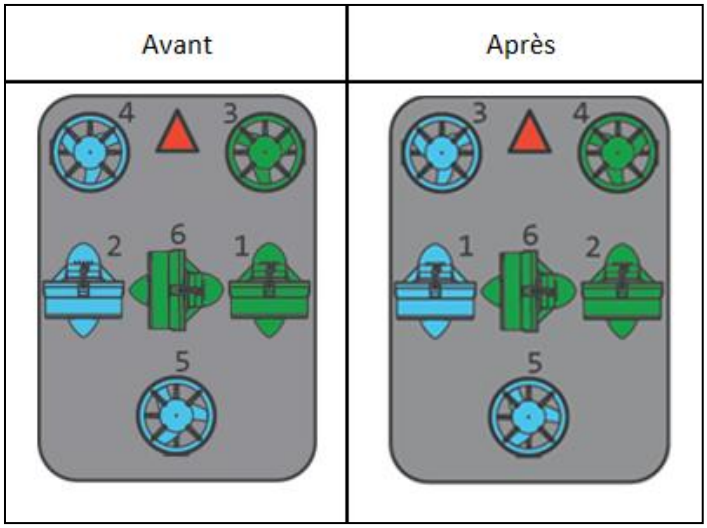

Figure 10 : Inversion des moteurs du BlueRov1

Maintenant que les robots fonctionnent comme prévu hors de l'eau. Il est nécessaire de tester l'étanchéité des tubes. Dans un premier temps, il faut vérifier que les bouchons sont bien installés et serrés. Puis refermer les tubes et tester l'étanchéité grâce à une pompe à vide. En pompant, on vide l'air dans le tube, donc la pression à l'intérieur du tube augmente, puis on attend et la pression ne doit pas diminuer.

Une fois les tests hors de l'eau effectués, le robot a pu être mis à l'eau afin d'effectuer les derniers réglages. Dans un premier temps, il fallait ajuster la flottaison du robot ainsi que son équilibrage en ajoutant des poids et des flotteurs.
Après quelques temps, il a été observé qu'il y avait une légère fuite à l'avant du robot. Le joint séparant l'avant du tube et sa partie centrale a été vérifié, mais la fuite n'était pas à la jonction des deux plexiglas mais uniquement sur la partie avant du tube. Afin d'identifier précisément l'emplacement de la fuite, nous avons rempli le tube et observé d'où sortait l'eau (Figure 11). Il se trouve que le plexiglas avait subi un léger choc. La partie endommagée ne pouvant être réparée, elle a été changée et les tests ont continué.

Figure 11 : Emplacement de la fuite du BlueRov1

La valeur matrice de gains a été légèrement modifiée afin que les déplacements soient le plus linéaire possible, de compenser les dérives du robot dus aux légères différences de puissance des moteurs. Même si ceux-ci sont théoriquement tous identiques, il y a une marge d'erreur lors de leurs fabrications, ce qui fait que les moteurs ne sont pas réellement similaires.

Pour effectuer des mesures sur la vitesse maximale du robot, une fonction permettant de garantir une profondeur constante a été créée. Cette fonction récupère les données du capteur de profondeur : la pression mesurée, les convertis en une profondeur (voir équation (3)) et met en route les propulseurs responsables de l'altitude (3,4,5 pour le BlueRov1 et 5,6 pour le BlueRov2, Figure 4) si la hauteur calculée dépasse un seuil prédéfini (Figure 12).

$$\boldsymbol{h} = \frac{(\boldsymbol{P_{mesurée} - P_{atmosphérique}})}{\boldsymbol{\rho * g}} \qquad (3)$$

Les déplacements des BlueRovs se font aussi avec une manette, elle ressemble à celle utilisée pour le Turtlebot2, mais parmi ses quatre boutons situés sur le dessus de la manette, les « gâchettes », il y en a deux analogiques et les deux binaires. Sur la manette utilisée pour le Turtlebot2, les « gâchettes » sont toutes binaires.

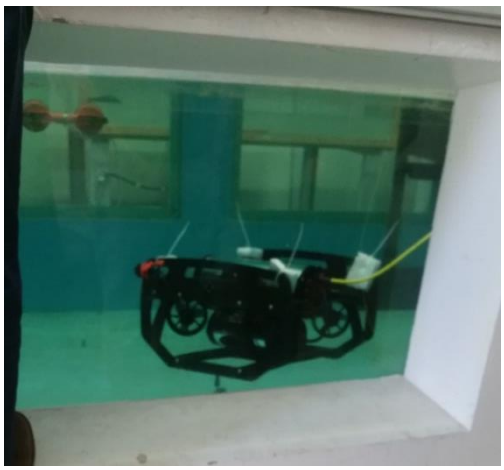

Figure 12 : Test BlueRov2 à profondeur constante

3.3. Sur le Mini-Rov

Le 18 mai, le Mini-Rov Coral est arrivé sur le site du centre de recherche IFREMER. De légers problèmes de l'entreprise SubSeaTech et des tests approfondis du robot ont retardé sa livraison d'un mois.

Il fallait assister à la recette afin de valider le bon fonctionnement et la bonne réception du nouveau robot. Aucun membre du laboratoire COSMER n'ayant d'expérience avec les librairies ROS ne pouvait être présent, c'est pourquoi j'ai été chargée de cette tâche. Au début de cette recette, l'alimentation a été vérifiée en commençant par tester l'alimentation sur secteur et le bon fonctionnement des sécurités (court-circuit, surtension) puis tester le bon fonctionnement de la batterie. Une fois la partie dangereuse vérifiée, le fonctionnement du robot a pu être testé, d'abord en le contrôlant avec le joystick et les autres boutons du panneau de contrôle (Figure 13). Il a été observé que la réactivité et la précision des déplacements et des capteurs étaient supérieurs à ceux des BlueRovs. Pour terminer, il fallait vérifier la bonne communication des fonctions ROS. J'ai observé que toutes les communications étaient actives et que la commande par ligne de code fonctionnait aussi bien que lors de la commande avec le panneau de contrôle.

Afin de pouvoir accéder au site d'IFREMER, ceux-ci nous ont demandé de remplir quelques formulaires sur les expériences que nous voulions effectuer avec le Mini-Rov. Il nous a également été demandé de remplir un plan de prévention et d'effectuer une inspection des lieux afin de connaître les règles de sécurité du site.

Malheureusement, une fois toutes les démarches administratives faites pour pouvoir accéder au site et manipuler le Mini-Rov, il a été observé par l'équipe d'IFREMER que le robot avait une fuite qui n'était pas présente lors de la recette et le robot fut renvoyé à SubSeaTech pour réparation. Le robot ne devrait être de nouveau opérationnel qu'à partir du 10 septembre 2018.

Figure 13 : Panneau de commande du Mini-Rov

3.4. Sur le Turtlebot3

Comme l'accès au Mini-Rov était ralenti par les démarches administratives et que le centre de recherche IFREMER était difficilement joignable en juillet, il a été décidé que des données seraient prélevées sur un Turtlebot3. Dans un premier temps, il a été convenu que l'on tenterait de reproduire l'expérience menée en simulation par Mme A.Feddaoui [4].Cette expérience consiste à se repérer en connaissant l'angle entre le robot et 2 balises et la distance par rapport à une des deux balises tout en sachant que les données sont asynchrones par nature. Le Turlebot3 permet de réaliser cette expérience, contrairement au Turtelbot2 ou aux BlueRovs, car le Turtlebot3 est équipé d'un lidar.

Comme les Turtlebots3 n'ont jamais été utilisés au préalable, ils n'y avaient que les communications avec les moteurs et l'IMU d'établies, ils ne disposaient pas encore de logiciel pour lancer le lidar et la caméra. J'ai cherché des pilotes compatibles et tester qu'ils fonctionnent correctement. Puis une fois le fonctionnement de ces capteurs validé, ils ont été ajoutés aux démarrages des robots.

Pour se servir aisément du Turtlebot3, j'ai adapté l'algorithme de fonctionnement du Turtlebot2 au fonctionnement de celui-ci. Dans un premier temps, il a fallu adapter les noms des capteurs pour la communication avec ROS. Certains capteurs, comme les pare-chocs et ceux permettant de vérifier le contact roue/sol, étant absent du Turtlebot3, il a fallu supprimer les actions qui les utilisaient.

La manette utilisée pour le Turtlebot3 est la même que celle des BlueRovs afin d'augmenter les possibilités de manœuvre avec le Turtlebot3 et d'harmoniser les déplacements des différents robots.

Lors des premiers tests de déplacements du Turtlebot3 avec la manette, il a été observé que si on lui demande d'avancer en tournant, le robot ne tournait pas, alors que s'il ne fait que tourner ou qu'avancer, tout se passe bien. Après avoir vérifié les vitesses maximales du robot (0.26m/s en translation et 1.82rad/s en rotation) sur la documentation [10], nous avons testé un déplacement en arc de cercle mais à une vitesse plus lente. Nous avons conclu que les valeurs indiquées ne correspondaient pas aux valeurs constatées expérimentalement. Nous avons donc mesuré nos vitesses maximales linéaire et angulaire. Pour la linéaire, nous avons chronométré le temps que mettait le robot à parcourir une distance connue à sa vitesse maximale et pour l'angulaire, nous avons chronométré le temps que mettait le robot à faire un tour, toujours à vitesse de saturation. Il s'est avéré que les vitesses maximales mesurées sont de 0.188m/s en translation et 1.297rad/s en rotation. Avec ces nouvelles données, nous avons pu mettre à jour le programme du robot afin de pouvoir effectuer n'importe quel déplacement.

Afin de pouvoir vérifier les déplacements du robot et valider une vérité terrain, il a été ajouté des mouvements prédéfinis, effectuer un cercle, un carré ou un triangle. Les déplacements ont été calculés en fonction de la vitesse et du temps. Par exemple, il a été défini que le carré ferait 1m de côté. Connaissant la vitesse du robot et la distance à parcourir, on a calculé le temps qu'il faudrait pour effectuer ce déplacement. A la fin du premier côté, le robot effectue un quart de tour en suivant le même procédé et continue jusqu'à finir le quadrilatère.

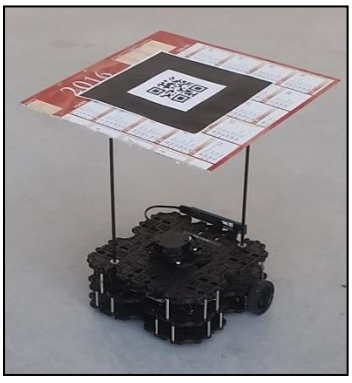

Figure 14 : Turtlebot3 avec QR-Code

Pour rendre plus fiable la vérité terrain, une caméra externe a été ajoutée, celle-ci observe la scène et donc les déplacements du robot. Comme les autres capteurs, la caméra a été ajoutée au programme via un « *Subscriber* » afin de pouvoir comparer les données reçues avec celle du robot (IMU, odométrie, etc…). Nous avons trouvé une

librairie capable de détecter le robot à l'aide d'un QR-Code "*visp-auto-tracker*" [11]. Le QR-Code a été installé sur le robot, grâce à un support, pour qu'il gène au minimum le fonctionnement du lidar (Figure 14). Il a fallu mesurer les dimensions du QR-Code afin que le nouveau programme puisse calculer la distance séparant la caméra du QR-Code. Une fois cette distance connue, il fallait faire un changement de repère pour connaitre la position et l'orientation du centre de rotation du robot. L'orientation du QR-Code reçue est sous la forme de quaternions. Pour simplifier les calculs de changement de repère, les données renvoyées pour la fonction "*visp-auto-tracker*" ont été mises sous forme de matrice de transformation M^{C}_{m0} puis multipliée par la matrice constante de changement de repère entre le centre du QR-Code et le centre des roues du Turtlebot3 M^{m0}_{r0} afin de connaitre la matrice de position du robot M^{C}_{r0}. Pour connaitre le déplacement que le robot a effectué à chaque instant, cette matrice est recalculée à chaque déplacement M^{c}_{mk} et comparée à la position initiale, équation (4).

Enfin, pour pouvoir comparer la position du robot, la matrice calculée est retransformée en deux vecteurs de trois valeurs représentant la position et l'orientation du robot.

$$\boldsymbol{M^{C}_{r0} = M^{C}_{m0} * M^{m0}_{r0}}$$

$$\boldsymbol{M^{r0}_{rk} = {M^{C}_{r0}}^{-1} * M^{C}_{mk} * M^{mk}_{rk} \quad sachant\ que\ M^{mk}_{rk} = M^{m0}_{r0}} \qquad (4)$$

Le déplacement du robot s'est avéré plutôt cohérent avec la commande demandée. Lorsque l'on demande un déplacement d'un mètre en ligne droite, le robot effectue bien la distance. La confirmation a été faite avec la caméra extérieur, l'odométrie et physiquement avec un marquage au sol. Il y a léger décalage lorsque le déplacement implique des rotations, lorsque l'on demande d'effectuer un cercle ou un carré par exemple, le robot s'arrête systématiquement quelques centimètres avant son point de départ. On suppose que ce décalage est dû à un temps de latence entre l'envoie et la réception de ordre, il reste néanmoins négligeable.

Un scénario a été mis en place dans le but d'obtenir des données à exploiter. Le lidar recevant des données dans un rayon de trois mètres, il a été décidé de le réduire et de ne pas mettre d'autre obstacle que ceux que l'on veut détecter dans le nouveau rayon d'acquisition. Afin de reproduire au mieux l'expérience simulée dans le document [4], le robot devait évoluer dans un rectangle définit de largeur délimitée par deux balises éloignées d'une distance choisie mais connue et de longueur choisie grâce au rayon d'acquisition du lidar. Les mouvements du robot sont aléatoires, il est commandé grâce aux joysticks. Pour éviter des problèmes qui pourraient se retrouver dans le calcul de la position du robot, celui-ci ne doit pas être sur la ligne formée par les deux balises ni sur les perpendiculaires à cette ligne passant par les balises. De plus, aucun obstacle ne doit se trouver en dehors de ce rectangle afin que seules les balises ne soient

captées par le lidar comme montré Figure 15. Les données qui sont enregistrées sont celles renvoyées par la manette, la commande envoyée aux moteurs, celles du lidar, de l'IMU et de l'odométrie. Les données sont ensuite traitées afin d'être réunies dans un seul fichier et organisées par ordre chronologique, les informations inutiles sont également supprimées puis sont transmises à Mme A. Feddaoui pour être exploitées et valider sa méthode avec des données réelles.

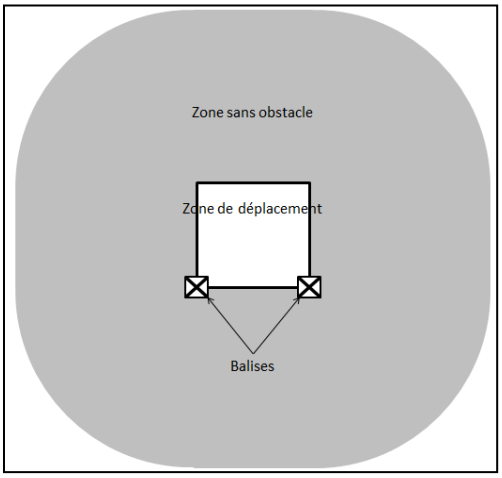

Figure 15 : Zone de déplacement pour l'acquisition de données

4. Poursuite des travaux

Pour continuer d'accompagner Mme A. Feddaoui et poursuivre mes travaux, un autre stagiaire, M. A. Mojika, a été recruté. Il est en quatrième années de l'Institut supérieur de l'électronique et du numérique (ISEN, [12]) ce qui fait que nous avons des compétences assez complémentaires.

Nous avons pu travailler ensemble quelques temps, ce qui m'a permis d'informer M. A. Mojika des avancements du projet et de ce qu'il restait à faire. Je l'ai également informé des protocoles de mise en route et d'arrêt des différents robots. Je lui ai également enseigné ce que j'avais appris sur l'utilisation des outils ROS.

Au cours de son stage, il devra effectuer les tests de l'algorithme sur les dernières données enregistrées, afin de confirmer le bon fonctionnement de cet algorithme. Lorsque le Mini-Rov sera opérationnel, il devra effectuer une première acquisition de données en suivant les scénarios que nous avons établis ensemble. C'est-à-dire, dans un premier temps, acquérir les données du sous-marin face à deux bouées et faire un mouvement de rotation sur lui-même afin de se retrouver successivement face aux deux balises. Puis dans un second temps, effectuer un mouvement défini et connu (comme

un cercle) et dans un troisième temps, effectuer un mouvement aléatoire, toujours en suivant les contraintes définies dans la Figure 15.

Il devra également continuer à implémenter l'algorithme dans le programme de comportement des robots : faire une fonction qui, à partir des anciennes et des nouvelles données connues d'une balise, corrige sa position dans l'espace. Il transmettra également ces connaissances sur ROS et les robots aux membres des laboratoires LIS et COSMER afin que ceux-ci puissent utiliser les robots. Je lui ai également fournit les modes d'emplois que j'ai mis au point sur les différents robots où j'ai répertorié la liste des problèmes rencontrés et les méthodes mises en place pour régler ces problèmes, afin qu'il puisse les compléter.

5. Conclusion

La mise en place du matériel et le développement de solutions pour une utilisation simple lors des séances de travaux pratiques a été entièrement réalisée. L'utilisation des robots et l'acquisition de leur données a été simplifiée afin que ceux-ci puissent être utilisés après mon départ.

Bien que le robot Mini-Rov, plateforme robotique centrale du stage tel qu'il a été initialement pensé, n'ait pas pu être utilisé, nous avons pu mettre en place des solutions alternatives afin de pouvoir réaliser les expériences souhaitées.

La validation et l'implémentation de l'algorithme de fusion de données asynchrones se poursuivra avec les travaux de M. A.Mojika.

Ce stage m'a permis de rendre fonctionnel un ensemble de robots tous différents en toute autonomie et de constater les différences d'utilisations entre les robots sous-marins et terrestres pour arriver au même objectif. J'ai également pu découvrir l'intérêt d'utiliser ROS, un outil très pratique et facile d'utilisation qu'il serait intéressant d'utiliser au cours de notre formation sachant que cet outil est énormément employé dans notre domaine.

Bibliographie

[1] «Université de Toulon,» [En ligne]. Available: http://www.univ-tln.fr/.

[2] «Laboratoire LIS,» [En ligne]. Available: http://www.lis-lab.fr/.

[3] «Laboratoire COSMER,» [En ligne]. Available: http://cosmer.univ-tln.fr/.

[4] A. Feddaoui, N. Boizot, E. Busvelle et V. Hugel, «High-gain extended Kalman for continuous-discrete systems with asynchronous measurements.,» mai 2018. [En ligne]. Available: http://hal.archives-ouvertes.fr/hal-01785473/.

[5] «Institut de recherche IFREMER,» [En ligne]. Available: https://wwz.ifremer.fr/mediterranee/implantations/La-Seyne.

[6] «SubSeaTech,» [En ligne]. Available: https://www.subsea-tech.com/?lang=fr.

[7] «Gazebo,» [En ligne]. Available: http://Gazebosim.org/.

[8] J. M. O'Kane, A Gentle Introduction to ROS, 2014.

[9] «Assembly BlueRov2,» [En ligne]. Available: http://docs.bluerobotics.com/brov2/assembly/.

[10] «Documentation Turtlebot,» [En ligne]. Available: http://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/#specifications.

[11] F. Novotny, «ROS : Visp-auto-tracker,» 2016. [En ligne]. Available: http://wiki.ROS.org/visp_auto_tracker.

[12] «Ecole d'ingénieur ISEN,» [En ligne]. Available: https://www.isen.fr/campus/ecole-ingenieurs-toulon/.