

# Projet Niche - Guerlédan

LEGOUALLEC Jules - SERY Mathys - PITAUD Mathieu - BELIER Titouan

December 2023

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Abstract . . . . .	3
1.2	Objectif du rapport . . . . .	3
1.3	Contexte . . . . .	3
1.4	Matériel et outils . . . . .	3
<b>2</b>	<b>Partie Mécanique</b>	<b>5</b>
2.1	Modification des travaux précédents . . . . .	5
2.2	Travail effectué . . . . .	5
<b>3</b>	<b>Contrôle</b>	<b>12</b>
3.1	Travaux précédents . . . . .	12
3.2	Travaux effectués . . . . .	12
3.3	Expérience et résultats . . . . .	13
<b>4</b>	<b>USBL</b>	<b>14</b>
4.1	Récupération des données . . . . .	14
4.2	Calibration magnétique . . . . .	14
4.3	Guidage . . . . .	16
4.4	Expériences supplémentaires et résultats . . . . .	17
4.4.1	Tests de distance et position relative . . . . .	17
4.4.2	Tests d'azimut . . . . .	18
<b>5</b>	<b>Vision caméra</b>	<b>20</b>
5.1	Travaux précédents . . . . .	20
5.2	Travail effectué . . . . .	20
5.2.1	Obtenir les images caméras du BlueRov Réaliser un algorithme de calibration intrinsèque . . . . .	20
5.2.2	Déterminer la pose du BlueROV par rapport à la cage . . . . .	21
5.2.3	Traiter l'image pour détecter sous l'eau . . . . .	21
5.2.4	Simuler le robot pour débuter le guidage . . . . .	22
5.3	Résultats . . . . .	24
<b>6</b>	<b>Conclusion</b>	<b>25</b>
6.1	Avancé du projet . . . . .	25
6.2	Suite du projet . . . . .	25

# 1 Introduction

## 1.1 Abstract

This report provides an overview of the progress of the "Niche" project. This 6-month project is undertaken by four third-year students specializing in autonomous robotics at ENSTA Bretagne. The project is supervised by Thomas Le Mézo, a teaching researcher at ENSTA Bretagne and a member of the ROBEX team. This project has been proposed to third-year robotics students for several years, building upon previous work to enhance the robot's positioning.

## 1.2 Objectif du rapport

Ce rapport nous sert de point de contrôle sur notre avancée. Il permet aussi à ceux voulant continuer nos travaux pour comprendre les enjeux du projet qui nous ont poussé à raisonner de cette façon et mettre en évidence les points d'attention auxquels nous avons fait face.

## 1.3 Contexte

Ce rapport fait l'état de l'avancé du projet "Niche". Ce projet de six mois est réalisé par quatre élèves de 3ème année de l'ENSTA Bretagne spécialisés en robotique autonome. Il est encadré par Thomas Le Mézo, enseignant-chercheur à l'ENSTA Bretagne, membre de l'équipe ROBEX. Ce projet est proposé depuis plusieurs années aux élèves de 3ème année de robotique. Il convient donc de partir des travaux effectués pour améliorer le positionnement du robot.

A l'origine, le projet consistait à asservir un ROV (Remotely Operated underwater Vehicle) de l'entreprise BlueRobotics pour qu'il soit en mesure de rentrer dans une "niche" sous l'eau. Cette niche prenait la forme d'une cage métallique placée à environ trois mètres de profondeur. Pour ce faire, le guidage du robot se fera en deux parties. Tout d'abord, il utilisera la communication entre deux USBLs (Ultra-Short BaseLine acoustic positioning system). Le premier situé sur le ROV et le second sur la cage. Cette communication sera utilisée pour la guidage à moyenne distance (d'environ 50m à 2m). A plus courte distance, le robot se positionnera grâce aux informations données par sa caméra qui détecte des codes fiduciaires (ArUco) placés sur la cage. Cette détection permet de déterminer sa position et son orientation relatives à la cage.

Cette année, le projet a quelque peu changé puisque le but final de la mission n'est plus de se placer dans la cage métallique mais de pouvoir "docker" précisément un appendice du robot dans une prise placée sur la cage. A terme, ce projet permettra de vérifier s'il est possible de docker deux robots sous marins entre eux pendant leur déplacement.

## 1.4 Matériel et outils

Pour réaliser ce projet, nous avons à disposition :

- Un ROV de l'entreprise BlueRobotics appelé BlueROV,
- Deux USBLs Subsonus de la marque Advanced Navigation.

Pendant la réalisation de ce projet, notre équipe a utilisé GitLab pour le versionnement des codes. Vous pouvez trouver le lien vers le dépôt à cette adresse : <https://gitlab.ensta-bretagne.fr/niche/niche>

## 2 Partie Mécanique

### 2.1 Modification des travaux précédents

Pour cette partie préliminaire du projet, le but était avant tout de remettre la cage en état et de se familiariser avec la structure.

Sur site, nous avions plusieurs objectifs:

- Comprendre comment la structure s'installe et se manipule (ce qui peut être une piste d'amélioration),
- Observer et ressentir quelle était la "synergie" entre le robot et la structure (Est-ce que la structure peine à accueillir le robot? Est-ce que le robot a des problèmes pour approcher la structure? etc),
- Et enfin, en déduire des pistes d'améliorations sur l'aspect "Docking" du projet. Autrement dit, qu'est-ce qui peut être simplifié, ajouté, enlevé ou amélioré sur la structure d'accueil du robot, voire sur le robot lui-même.



Figure 1: Photo de la cage complète et montée

### 2.2 Travail effectué

Le projet "Niche" est en réalité un projet "Docking" : l'objectif de ce projet est de réaliser une structure permettant de docker l'appendice d'un robot à un appendice sur la structure. La réflexion derrière ce projet est, comme pour les avions de chasses en air, de permettre au robot de se ravitailler et/ou d'échanger des données sans avoir à revenir à la base et arrêter sa mission.

Cette manière de relier un véhicule à une structure ou à un autre véhicule, ainsi que les différents projets de docking trouvables sur internet nous ont encouragé à repenser la conception pour partir sur une idée de réception "en entonnoir".

L'avantage de cette méthode c'est que la forme en entonnoir va accompagner le bras, l'attache, la prise du ROV (voire le BlueROV lui-même) jusqu'à la sortie de data et d'énergie



Figure 2: Méthode d'attache pour ravitaillement en carburant

de la structure de réception. La commande peut donc se permettre de ne pas être très précise (jusqu'à un certain point) dans ce cas. Cela offre une certaine marge d'erreur.

Cependant, dans le cas de notre projet nous avons 2 choses contraignantes pouvant dissuader de passer sur une réception entonnoir prenant le robot au complet comme les idées ci-dessus:

1. La première c'est la forme très cubique de notre robot. La réception entonnoir s'associe bien avec des AUV ou ROV long, cylindriques et profilés mais dans le cas d'un BlueROV (avec Heavy Configuration), on risque d'avoir un robot qui se coince et/ou qui ne peut pas forcément arriver au bout (donc là où se trouve la potentielle entrée-sortie data/énergie).
2. La deuxième c'est la présence de l'USBL sur le dessus du Blue ROV. Si le robot rentre et s'enfonce de plus en plus dans un entonnoir fermé, ce dernier risque de percuter l'USBL, l'endommager et/ou le décrocher. Une solution serait de faire une rainure au niveau de l'entonnoir pour que l'USBL puisse s'y mettre lorsque que le ROV s'insère dans l'entonnoir. Le risque de cette méthode serait de perdre l'avantage offert par l'entonnoir : cela demanderait d'être précis sur l'insertion de l'USBL dans la rainure et d'être prudent lors de l'avancée du ROV pour que l'USBL ne tape les bords de la rainure et l'endommage.

La solution qui a été adoptée pour l'instant est de relier le BlueROV à la structure à l'aide de deux appendices : un appendice sur la structure et un autre sur la structure faite de profilés. Ces deux pièces seront usinées en POM-C.

La forme en entonnoir a bien été conservée. Sur les deux pièces, nous avons différentes petites extrusions afin de laisser l'eau circuler pendant leur utilisation et ainsi ne pas créer d'effet ventouse ou autre effet indésirable.

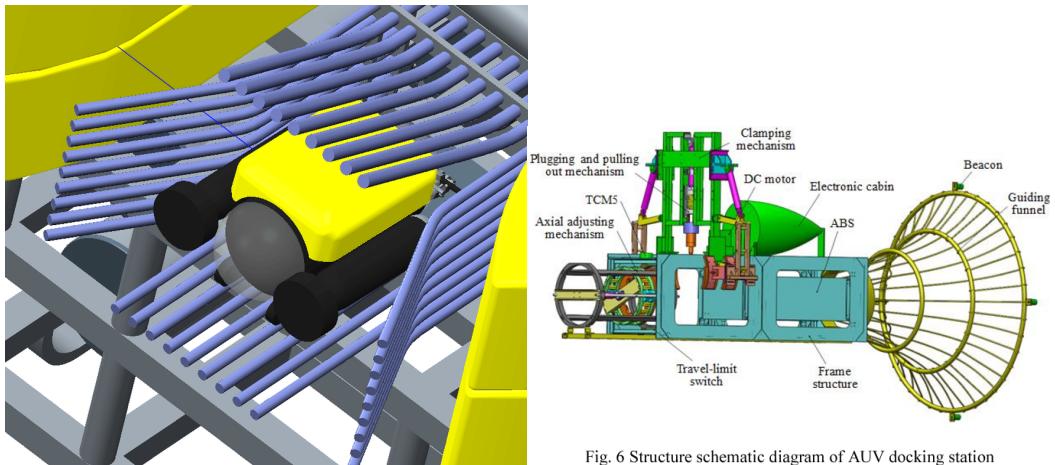


Fig. 6 Structure schematic diagram of AUV docking station

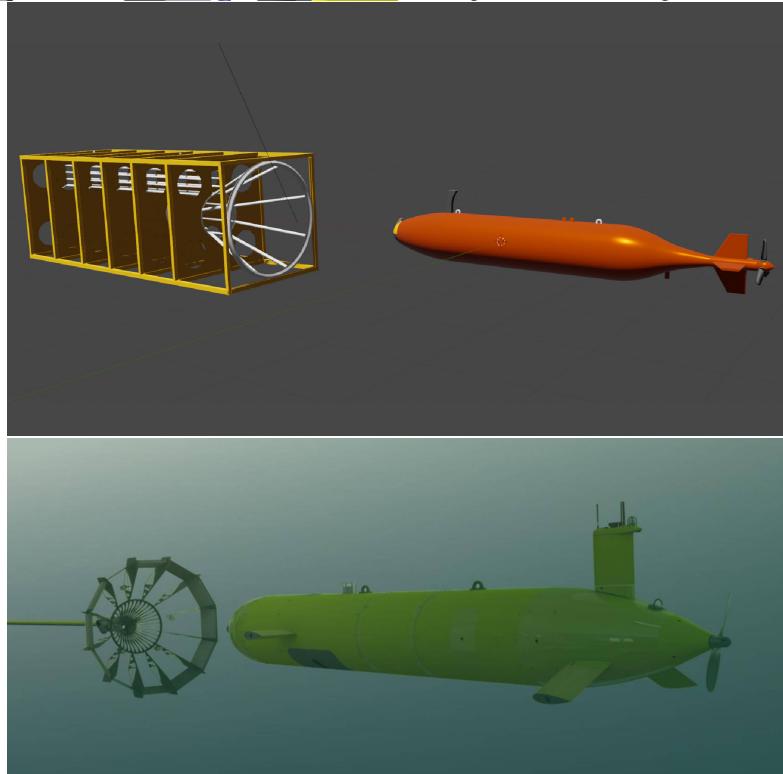


Figure 3: Idées de référence pour la reconception

La pièce à mettre sur le BlueROV sera fixée sur la Tape avant en utilisant les vis de cette dernière (un peu plus grande pour que cela traverse à la fois la pièce et la tape).

Afin d'accueillir le plug récepteur, la structure de profilés a été modifiée, simplifiée et renforcée suite aux observations faites à Guerlédan et pour pouvoir supporter la charge du robot au moment de l'accroche. Il faut un support de fixation stable et résistant.

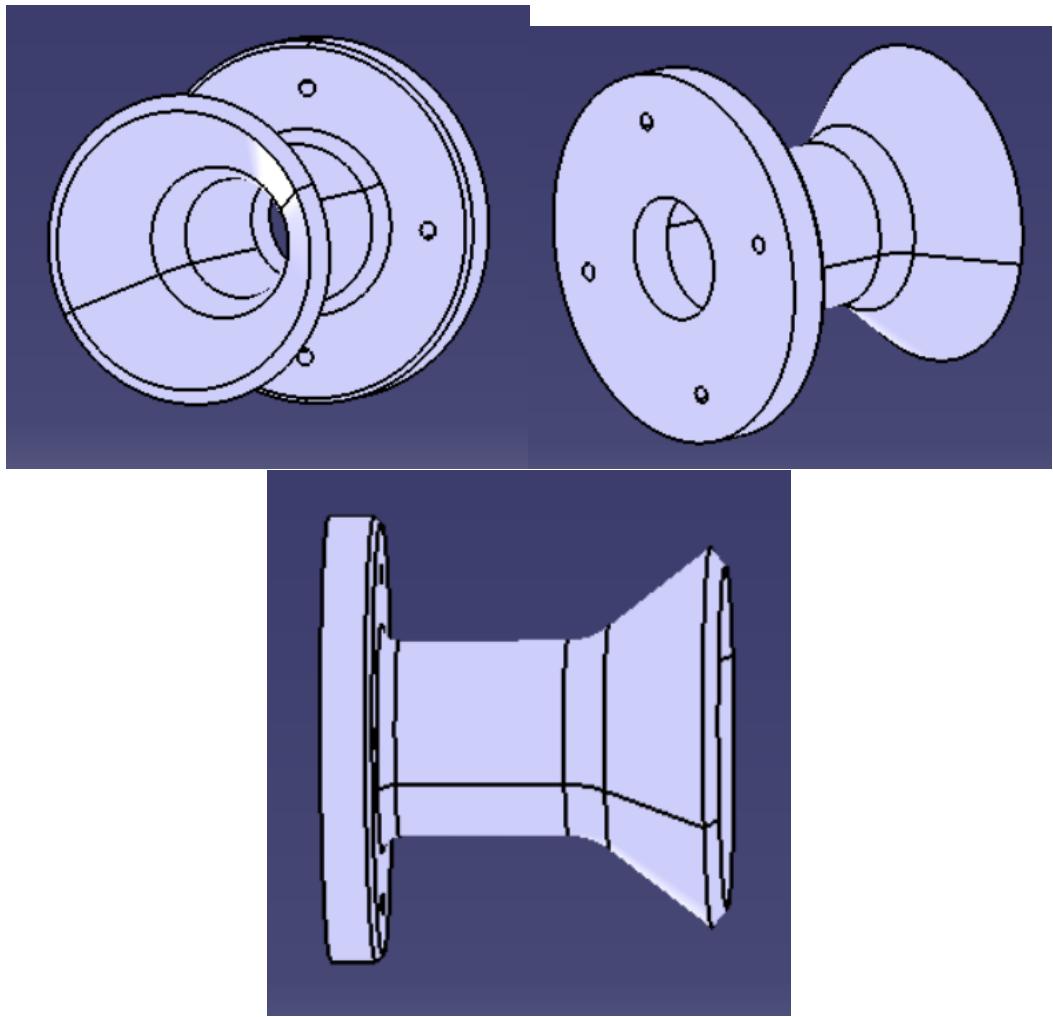


Figure 4: Appendice pour structure profilé

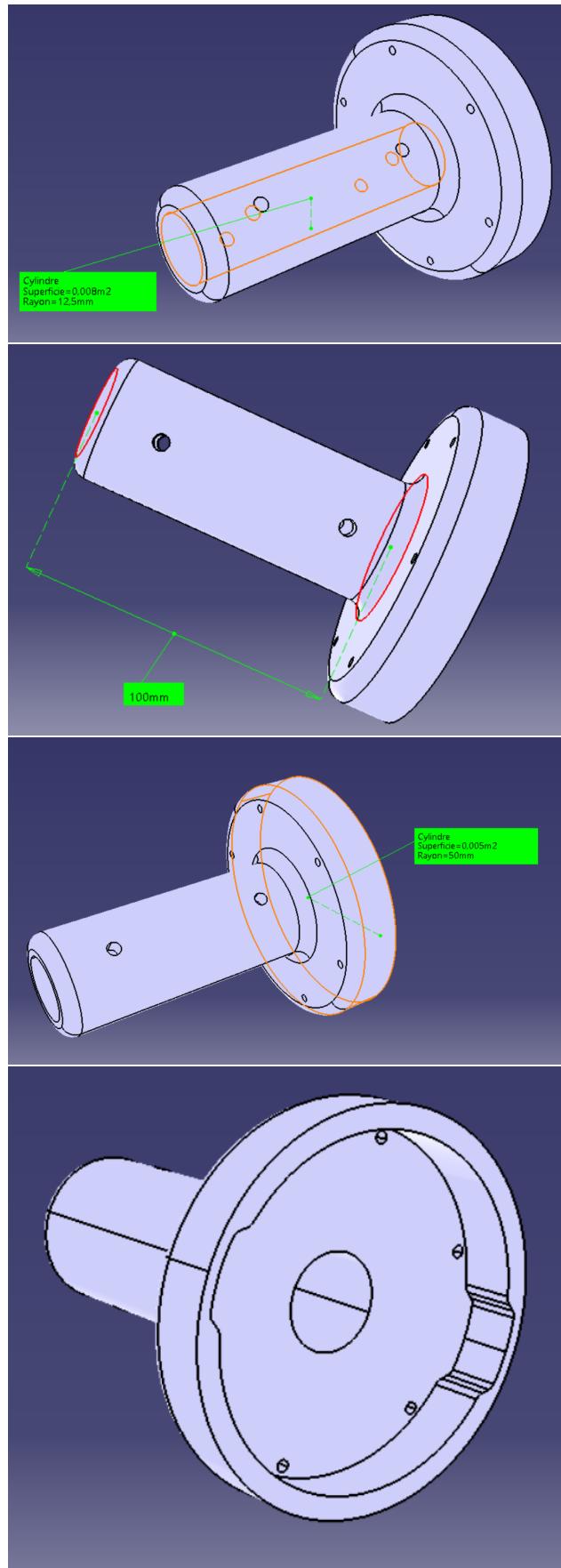


Figure 5: Appendice pour BlueROV

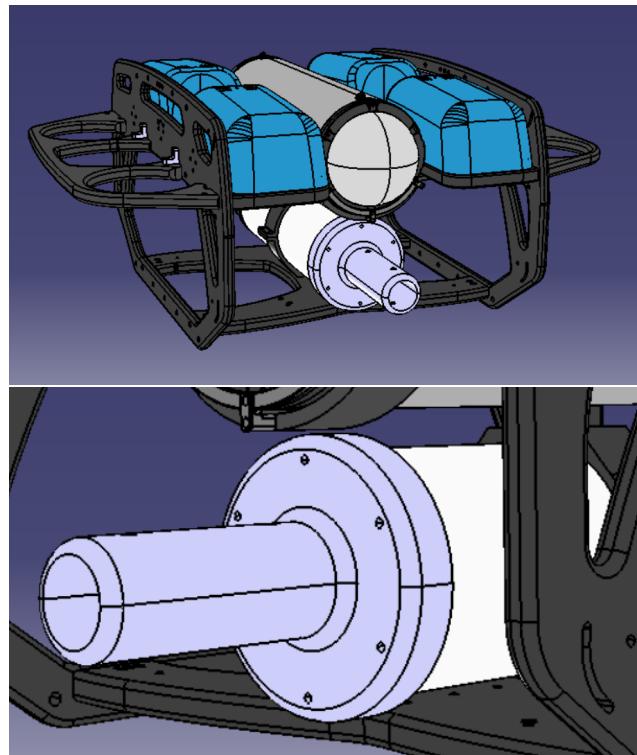


Figure 6: BlueROV avec son plug (fichier CAD fourni pour simulation)

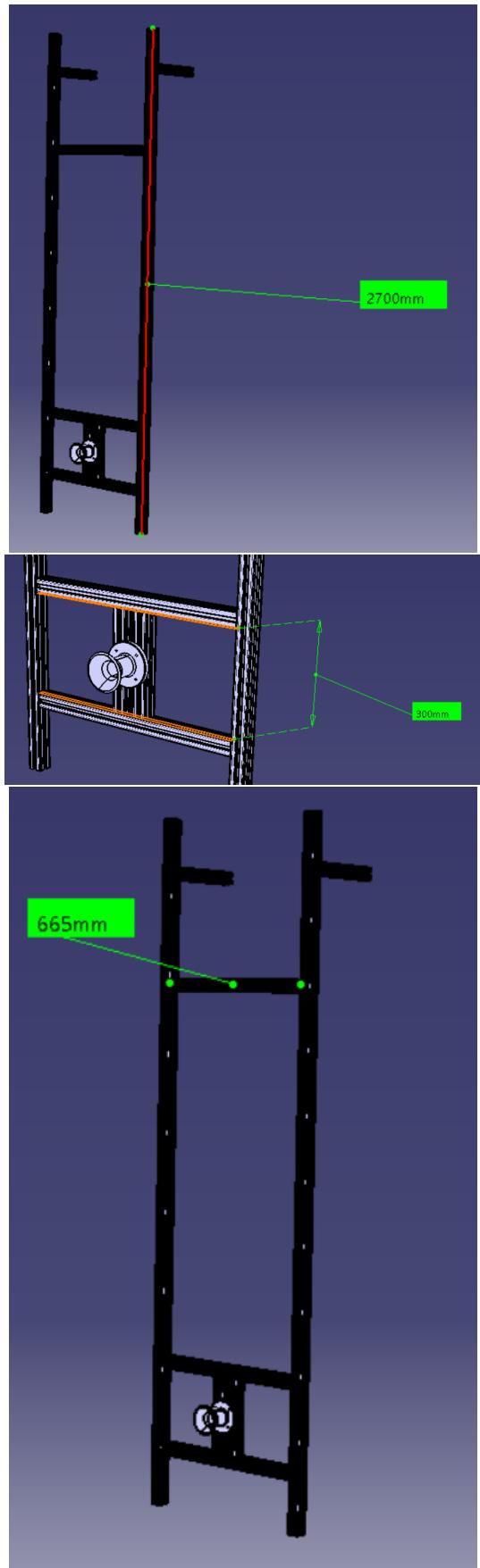


Figure 7: Structure de profilés avec son plug (fichier CAD fourni pour simulation)

## 3 Contrôle

### 3.1 Travaux précédents

Les étudiants précédents ont codé le contrôle du robot en ROS en se basant sur le code de télécommande de Christophe Viel. Ils envoyait les commandes au robot en utilisant des messages Mavros de type *RC\_Override*.

### 3.2 Travaux effectués

Pour coder le contrôle du ROV, nous nous sommes aussi inspiré du code de Christophe VIEL, chargé de recherche du CNRS, membre de l'équipe ROBEX. Des difficultés rencontrées en essayant d'utiliser Mavros avec ROS2 nous poussent à utiliser directement des messages Mavlink pour envoyer les commandes au robot. Le node récupère les valeurs des boutons de la manette et envoie des messages Mavlink au robot.

Le plugin *Mavlink inspector* de QGroundControl permet de vérifier que les paramètres du message sont bien les mêmes qu'avec Mavros.

De même que pour le groupe précédent, il est possible de commander le drone en lacet, en tangage, en roulis. Il est aussi possible d'ordonner au drone de se déplacer de haut en bas, d'avant en arrière et latéralement. Les valeurs envoyées sont des entiers entre 1300 et 1700. Elles représentent des vitesses mais ce n'est pas possible de commander directement une vitesse précise.

Il est désormais possible de contrôler le robot à la télécommande ou en envoyant un message de type *Twist* sur le topic */real/cmd\_vel*. Le passage du mode Manuel au mode Automatique se fait en appuyant sur le bouton Start, ce qui facilitera les expérimentations lors de la prochaine semaine au lac de Guerlédan.

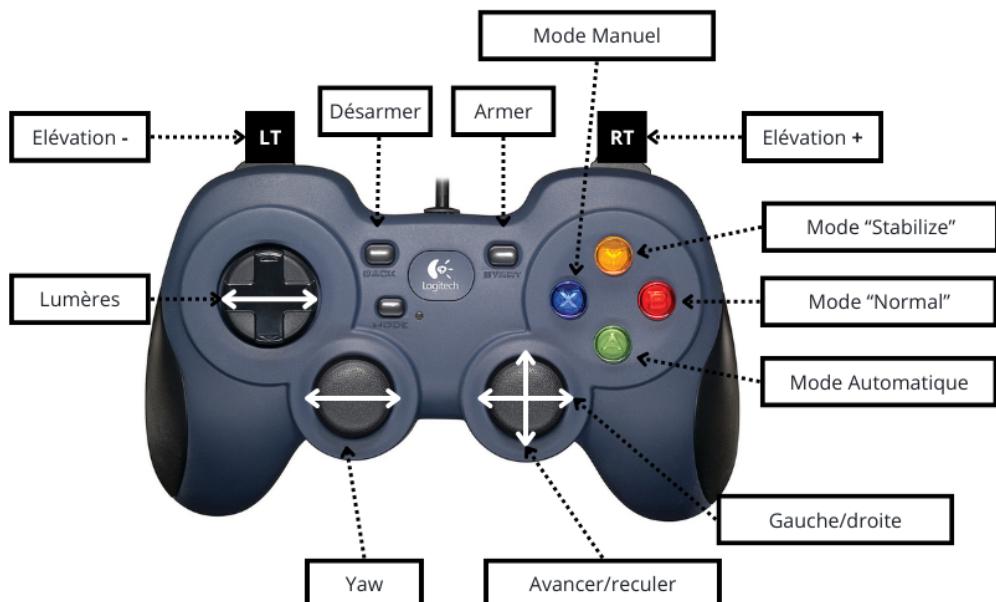


Figure 8: Inputs de la télécommande

### 3.3 Expérience et résultats

Nous avons testé le contrôle du ROV dans la bassin Hydro avec la télécommande. Le micro-contrôleur déjà présent sur le robot est efficace, le ROV est très maniable. Cependant en mode "Manual" le robot ne reste pas droit lorsqu'il avance, ou lorsqu'aucune commande ne lui est envoyée.

Nous avons donc testé plusieurs modes implémentés dans le micro-contrôleur. Le mode "Depth hold" permet de garder une profondeur constante mais le robot oscille en tangage et roulis, ce qui est problématique car le robot doit être stable pour pouvoir se docker à l'appendice de la niche. Le mode "Stabilize" permet de rester parfaitement droit lorsque le ROV avance et lorsqu'il est immobile. Nous avons donc choisi d'utiliser ce mode pour le futur guidage. Le seul désavantage est que le robot ne maîtrise pas directement sa profondeur, donc il reste à le coder nous-même.

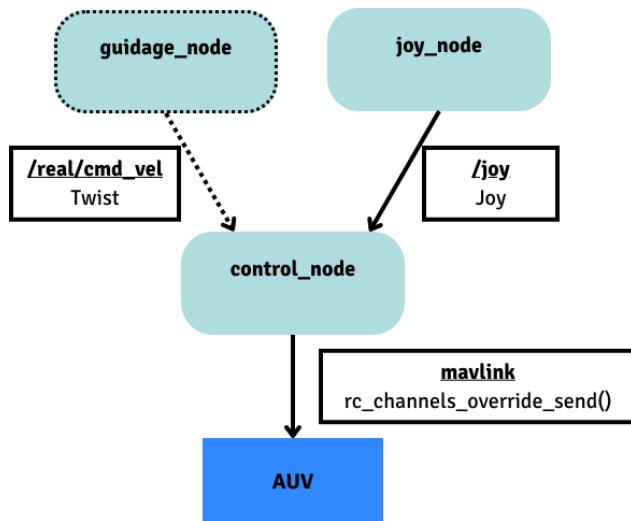


Figure 9: Architecture ROS2 du contrôle

## 4 USBL

### 4.1 Récupération des données

Pour la récupération des données, nous sommes repartis des codes faits par Maël GODARD, doctorant en robotique à l'ENSTA Bretagne, qui avait lui même récupéré certains drivers dans la documentation des UBSLs.

Nous avions un code d'exemple permettant de récupérer les valeurs d'un packet envoyé par l'USBL. Ensuite, nous avons créé nos propres nodes pour récupérer tous les packets nécessaires au guidage du ROV (un node par packet).

Pour s'assurer que nos nodes soient correctement fonctionnels, on vérifie sur Wireshark le nombre de packets reçus et on compare au nombre de données publiées avec ROS.

On réalise deux tests avec les packets de données brutes de la centrale inertuelle de l'UBSL (SubsonusRawSensorsPacket), un test avec l'UBSL de surface et un test avec l'USBL du ROV.

#### USBL surface :

Messages ROS : 4538

Paquets reçus : 4733 (filtre "*ip.src == 192.168.2.100 & tcp.port == 20000*" sur Wireshark)

Durée : 90s

#### USBL ROV :

Messages ROS : 2185

Paquets reçus : 2178 (filtre "*ip.src == 192.168.2.200 & tcp.port == 20000*" sur Wireshark)

Durée : 110s

### 4.2 Calibration magnétique

Afin d'avoir des valeurs de cap précises, qui seront nécessaires pour le guidage, nous devons calibrer le magnétomètre de l'USBL. En effet, avec tout l'électronique présent dans le ROV, il est probable que le magnétomètre subisse des perturbations qui peuvent fausser les mesures si elles ne sont pas prises en compte. Tout d'abord, on veut s'assurer que chaque axe ait la même sensibilité, et qu'il n'y a pas de bruit constant dans les mesures. On va donc prendre une série de mesures en faisant tourner l'USBL dans tous les sens (de préférence à l'extérieur dans un endroit dégagé), de manière à avoir des mesures selon chaque zone de l'espace. Si le magnétomètre est bien calibré : l'affichage des points de mesure doit être une sphère. En pratique c'est plutôt une ellipse, à l'aide d'un package de calibration on trouve les matrices  $A$  et  $b$  de qui définissent respectivement la transformation de l'ellipse et son centre. En appliquant ces transformations dans le sens inverse, on peut ramener toutes nos mesures (notées  $X$ ) dans une sphère centrée en zéro. On a donc une normalisation selon chaque axe et un centrage en zéro.

$$X_{corrected} = A(X - b) \quad (1)$$

Ensuite, on veut que notre valeur de cap ne soit pas impactée par le tangage et le roulis du robot. Ce qui revient à projeter nos mesures magnétiques dans le plan horizontal (le plan

orthogonal à l'axe de la pesanteur). Ainsi, on retire de nos mesures la composante selon l'axe de  $\vec{g}$  :

$$X_{proj} = X_{corrected} - (X_{corrected} \cdot \vec{g}) * \vec{g} \quad (2)$$

Nous avons testé l'accéléromètre en le posant sur chacun de ses axes pour vérifier qu'il renvoie bien environ  $g$  sur son axe soumis à la gravité, et 0 sur les deux autres. Si ce n'est pas le cas, une calibration est nécessaire.

Ensuite, on prend une mesure magnétique de référence, notée  $\vec{N}$  en dirigeant l'USBL vers le Nord. Cette mesure permet de compenser un léger décalage sur l'axe de la centrale inertie. Pour connaître notre cap, il suffit alors de calculer l'angle entre le vecteur de référence et notre mesure projetée.

$$cap = \arccos(\vec{N} \cdot \vec{X}_{proj}) \quad (3)$$

On teste ensuite nos valeurs de cap. Protocole : on oriente le ROV en direction de  $0^\circ$ ,  $90^\circ$ ,  $180^\circ$  et  $-90^\circ$ , en restant quelques secondes à chaque fois. On utilise notre téléphone comme valeur de référence de cap.

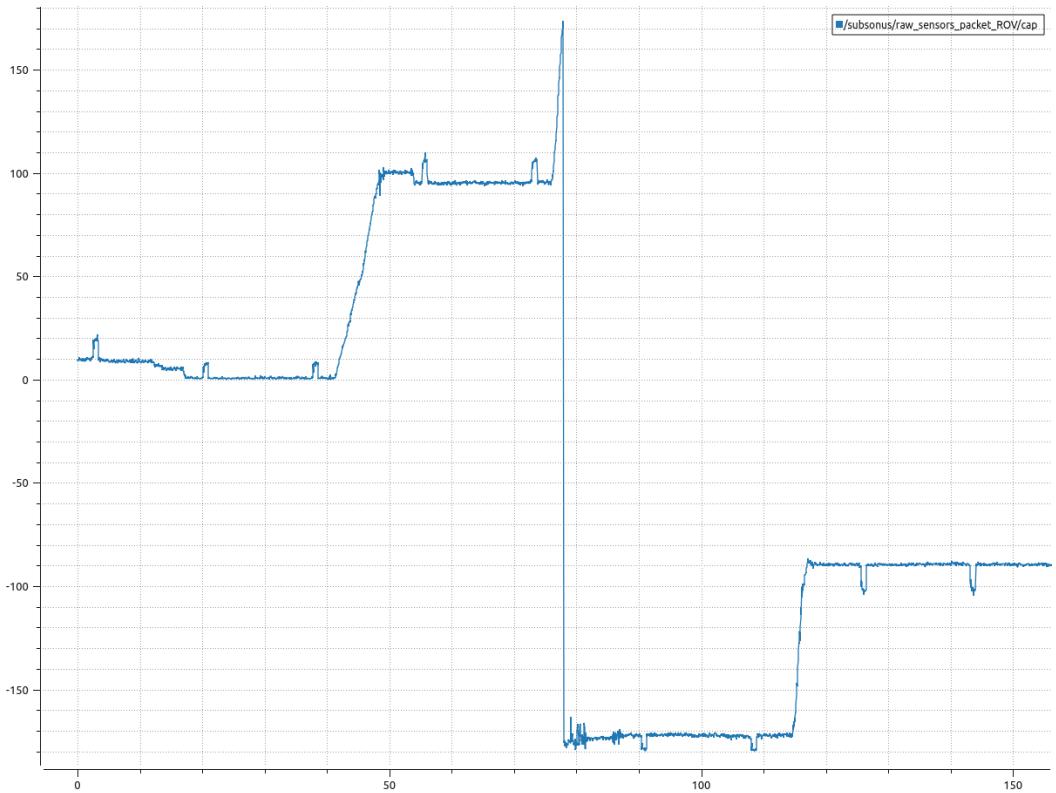


Figure 10: Résultats des tests de cap

On observe que les valeurs sont assez stables et quasiment égales à celles données par notre téléphone ( $+/- 5^\circ$ ).

#### **Limites et améliorations :**

Pour tester la calibration, on a appliqué les transformations inverses de l'ellipse sur un jeu de données, pour avoir un jeu de données bien calibré. Ensuite, on affiche les matrices qui définissent l'ellipse de ce nouveau jeu de données, on est sensé trouver des matrices  $A = I_3$  et  $b = 0_{(3,1)}$ . Cependant, nous n'obtenons pas ces résultats, peut-être parce qu'on ne prend

qu'un petit échantillon de données pour calculer les matrices de l'ellipse. Cette imprécision participe à l'erreur sur le cap (erreur qui reste raisonnable).

Egalement, malgré la projection des mesures, les valeurs de cap changent drastiquement lorsqu'on applique beaucoup de tangage proche d'un cap de  $0^\circ$  ou  $180^\circ$  ( $+/- 40^\circ$  pour un tangage d'environ  $60/70^\circ$ ). Autrement, les valeurs de cap restent assez stable ( $+/- 10^\circ$ ) lorsqu'on applique du roulis, ou du tangage avec un cap loin de l'axe Nord-Sud.

### 4.3 Guidage

La stratégie de guidage n'est pas encore définie, mais on récupère déjà toutes les données nécessaires pour guider le ROV vers la cage :

#### Stratégie 1 :

- Distance entre les USBLs (en m)
- Azimut (entre  $-180^\circ$  et  $180^\circ$ )
- Cap de chaque USBL (entre  $-180^\circ$  et  $180^\circ$ )
- Différence de hauteur (en m)
  - $z < 0$  si le ROV est moins profond que l'USBL surface
  - $z > 0$  si le ROV est plus profond que l'USBL surface

Il faudrait ensuite déterminer le cap désiré du ROV, noté  $\hat{\theta}_{ROV}$ , en fonction du cap des deux UBSLs,  $\theta_{ROV}$  et  $\theta_{surface}$ , et de l'azimut  $\alpha$  :

$$\hat{\theta}_{ROV} = \theta_{surface} + \alpha - \pi \mod 2\pi \quad (4)$$

Puis on commanderait le robot en vitesse de rotation pour le mettre suivant la bonne direction, avant de le faire avancer tout droit :

$$w = \text{sawtooth}(\hat{\theta}_{ROV} - \theta_{ROV}) \quad (5)$$

#### Stratégie 2 :

- Position relative dans le repère de l'UBSL de surface : x, y
- Différence de hauteur (en m)
  - $z < 0$  si le ROV est moins profond que l'USBL surface
  - $z > 0$  si le ROV est plus profond que l'USBL surface
- Cap de chaque USBL

Il faudrait ensuite passer de la position relative dans le repère de l'UBSL de surface à la position relative dans le repère du ROV. Afin de pouvoir contrôler le ROV en vitesse dans son repère.

## 4.4 Expériences supplémentaires et résultats

Afin de faire le guidage, il est nécessaire de vérifier la fiabilité des données envoyées par les UBSLs. Pendant la semaine à Guerlédan nous avons étudié la cohérence des valeurs de distance, de position relative et d'azimut. Nous avons mis en place des filtres médians sur ces données car nous recevions régulièrement des valeurs aberrantes.

### 4.4.1 Tests de distance et position relative

Protocole : on a tiré un mètre entre le ROV et la cage attachée au ponton. Le ROV était maintenu sous l'eau depuis un kayak, avec le bout du mètre maintenu sur l'UBSL. L'autre USBL était fixé sur la cage mais à deux mètres de profondeur (ce qui rajoute de la distance par rapport à la mesure du mètre). Il était difficile de maintenir une distance constante avec le kayak, ce qui rajoute de l'incertitude.

On place les UBSLs à 10m, 20m, 30m et 40m l'un de l'autre, on vérifie les valeurs de distance renvoyées ainsi que la cohérence des valeurs de positions relative ( $x, y, z$ )



Figure 11: Exemple de résultats pour  $d=40m$

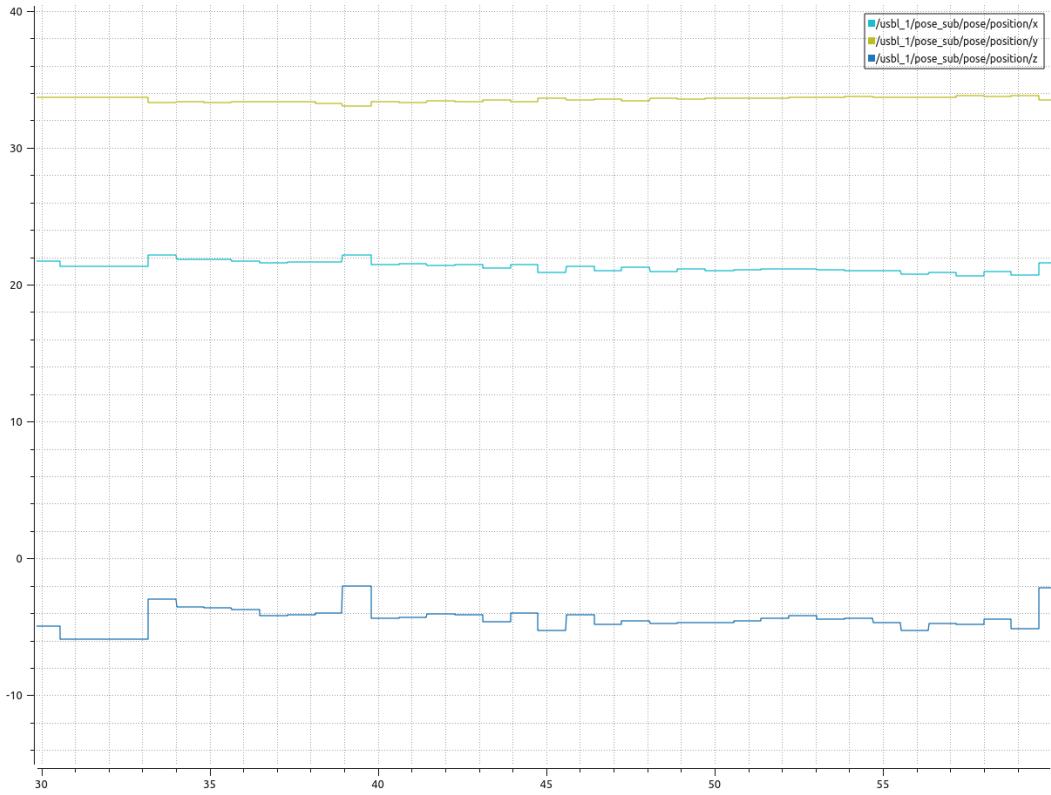


Figure 12: Position relative

Les valeurs de positions relatives sont cohérentes, la norme vaut environ 40m et on retrouve bien les 2m de profondeur de la cage. Avec beaucoup d'incertitudes de protocole et de mesures, on ne peut pas conclure sur la précision exacte du capteur.

#### 4.4.2 Tests d'azimut

L'azimut est l'angle entre l'axe de l'USBL de surface et l'axe  $USBL_{Surface} - USBL_{ROV}$  (entre  $-180$  et  $180^\circ$ ). Protocole : on réalise un arc de cercle de  $-90^\circ$  à  $90^\circ$  avec le ROV contrôlé à la manette autour de la cage (sur laquelle est placé l'USBL de surface), et on mesure l'azimut. On marque une pause à  $-90^\circ$ ,  $0^\circ$  et  $90^\circ$ . On obtient des résultats cohérents mais avec les incertitudes de mesure, dûes au protocole compliqué à mettre en place sur le lac, on ne peut pas conclure sur la précision.

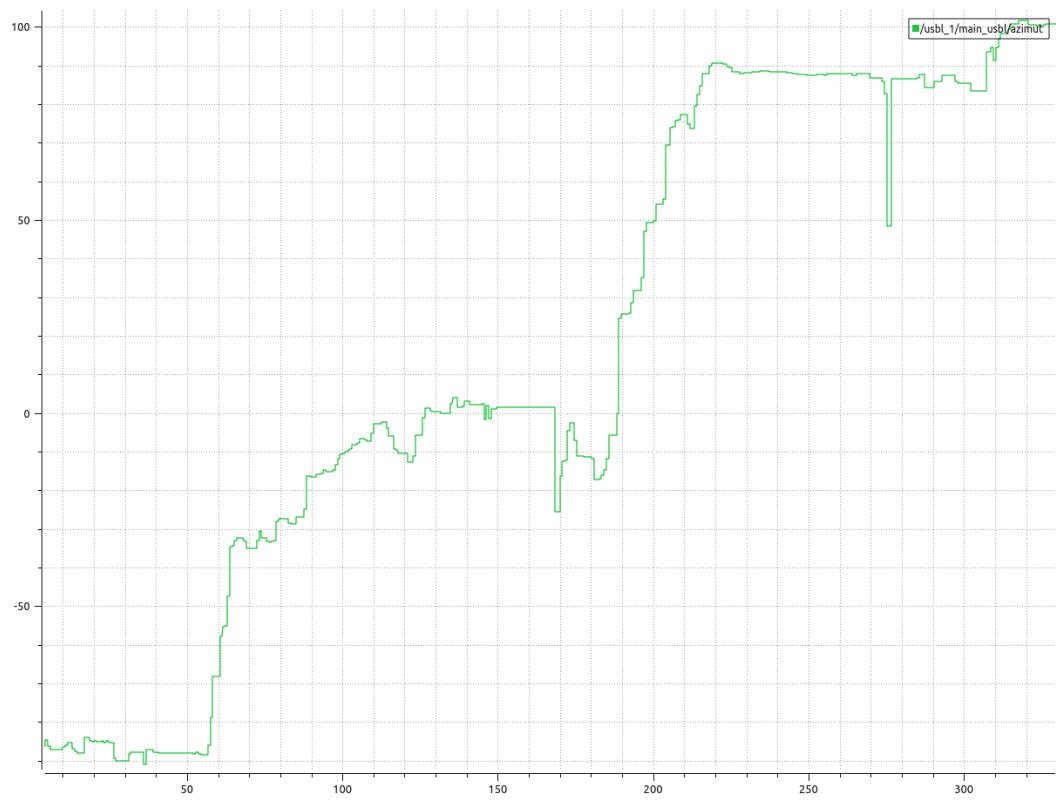


Figure 13: Mesures d'azimut

## 5 Vision caméra

Lorsque le robot arrive proche de la cage, à environ 3 mètres. Le guidage par USBL laisse place au guidage par vision. En effet, si cette deuxième méthode est inutilisable à longues distances puisque la vision par caméra est obstruée par la turbidité de l'eau, elle est au contraire précise pour de courtes distances.

### 5.1 Travaux précédents

Le groupe de l'année dernière avait réalisé un guidage par vision qui leur permettait d'entrer dans la cage métallique. Il nous faudra donc reprendre ce guidage à notre façon pour assurer un guidage précis. Dans notre cas, nous n'essayons pas de rentrer le robot dans la cage mais bien de docker un appendice à un autre présent sur cette cage, ce qui rend la tâche plus complexe.

Il convient également d'améliorer le traitement d'image et de réaliser une calibration de la caméra sous l'eau permettant de déterminer la position de notre robot avec précision par rapport aux ArUco. Nous utilisons la librairie OpenCV et plus précisément l'algorithme RANSAC pour la calibration.

### 5.2 Travail effectué

#### 5.2.1 Obtenir les images caméras du BlueRov Réaliser un algorithme de calibration intrinsèque

La première tâche à faire pour faire du guidage par vision est d'être en mesure de récupérer les images de la caméra du BlueROV puis de calibrer cette caméra. Récupérer la vidéo du BlueROV n'est pas trivial. Il faut créer un objet Video() qui prend le port de communication entre la caméra et votre ordinateur. Nous vous conseillons de consulter le code ViewSaveDetect.py présent sur notre Gitlab : [lien](#).

Maintenant que nous avons récupéré la vidéo, nous pouvons réaliser la calibration intrinsèque. Pour réaliser cette calibration intrinsèque nous utilisons un damier de taille (5,8), il faut faire attention à prendre le nombre d'intersections (pas de carrés) en compte lorsqu'on entre la taille du damier dans le code. Dans la Figure 14 ci-dessous, vous pouvez trouver la calibration que nous avons faite dans l'air, par la suite, nous ferons cette calibration dans l'eau puisque l'interface entre le hublot du ROV et l'eau déforme légèrement l'image (créant une léger effet fisheye).



Figure 14: Calibration intrinsèque grâce au damier

Avec cette calibration on récupère les coefficients paramètres intrinsèques :

- les coordonnées du centre optique  $cx$  et  $cy$ ,
- les distances focales selon  $x$  et  $y$ , elles sont égales dans notre cas,
- les paramètres de distorsions.

Sans entrer dans le détail car c'est l'objet de la sous-partie suivante, nous avons pu quantifier la validité de notre calibration grâce à notre algorithme qui détermine la distance et l'orientation entre la caméra et la cage. On place des marqueurs à une distance connue et on vérifie que cette valeur est cohérente avec ce que nous renvoie l'algorithme. Si c'est le cas, la calibration est bonne. Dans nos expériences, nous avons une erreur sur la distance de l'ordre de 5 à 10 centimètres lorsque le marqueur est placé à 4 mètres. Si cet écart est négligeable à de longue distance, il devient important lorsque l'on désire se docker avec précision (à courte distance, nous avions encore une erreur de quelques centimètres). Nous prendrons plus d'image pour notre calibration sous l'eau pour assurer une calibration performante.

### 5.2.2 Déterminer la pose du BlueROV par rapport à la cage

Pour réaliser le guidage par la vision, nous avons besoin de connaître la position et l'orientation de la caméra du BlueROV par rapport aux marqueurs. Ces informations nous permettant de déterminer la position de l'appendice du robot par rapport à l'appendice de la cage.

Notre code, trouvable sur ce lien nous permet de détecter les marqueurs et d'afficher sur l'image la distance du marqueur avec la caméra selon les 3 axes ainsi que son orientation dans les 3 axes. Le traitement de l'image appliqué avant cette détection est présenté dans la sous-partie suivante. Finalement, nous utilisons un filtre médian sur les valeurs pour supprimer les valeurs aberrantes.

### 5.2.3 Traiter l'image pour détecter sous l'eau

Lors de sa mission sous l'eau, le caméra doit détecter les marqueurs ArUco présents sur la cage pour déterminer sa position et son orientation relative. Cependant le milieu aquatique est très

contraignant pour la vision, et ce, pour plusieurs raisons. Tout d'abord à cause de la turbidité et les bulles d'airs qui peuvent s'intercaler entre la caméra et le capteur, ce qui obstrue la vision mais surtout parce qu'un gradient de luminosité apparaît et fausse la détection. En effet, le fond du lac (ou de la mer selon la mission) est bien plus sombre que la surface de l'eau. Ce gradient de luminosité rend les parties inférieures de l'ArUco bien plus sombres que celles supérieures, faussant l'algorithme de détection, le ROV ne sait alors plus se situer dans son environnement, il avance à l'aveugle.

Pour pallier ces désavantages du fond marin, on traite l'image en appliquant une amélioration de contraste de l'image. Plus précisément, on utilise l'histogramme d'égalisation du contraste limité (CLAHE) dans l'espace colorimétrique LAB. Ensuite, on applique une correction gamma non linéaire pour ajuster la luminosité de l'image.

La figure 15 représente la vision de la caméra lorsque le robot est en mission à environ 2 mètres sous l'eau. L'image de gauche représente la vision avant traitement de l'image, le ROV ne détecte pas l'AruCo. Tandis que celle de droite représente la vision après avoir traité l'image, et dans ce cas la détection est bien possible. Le BlueROV peut alors se positionner dans son environnement comme on peut le voir dans le coin supérieur gauche de cette image.

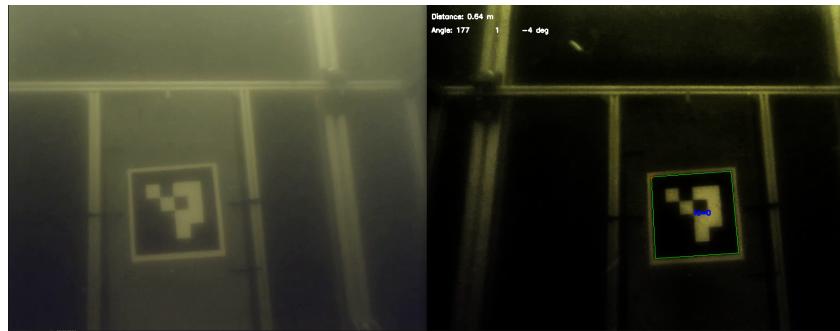


Figure 15: Résultats de la détection après traitement d'image

#### 5.2.4 Simuler le robot pour débuter le guidage

Dans le but de pouvoir commencer le guidage du robot grâce à la vision, nous avons créé une simulation sous Gazebo du robot en parallèle de la partie contrôle. En effet, si nous ne pouvions pas encore tester le guidage sur le robot réel, nous pouvons désormais faire nos tests sur la simulation simplifiée. Nous pouvons faire ces tests quand nous voulons, où nous voulons. Sans avoir à faire tous les branchements du robot à chaque fois tant que le guidage n'est pas pertinent en simulé.

Le simulateur retenu est Gazebo puisque nous avons déjà réalisé un projet sur ce simulateur. Attention ce simulateur est très laborieux à utiliser à cause de la documentation lacunaire voire inexistante, en particulier sur ROS2. On conseille de réaliser soi-même son propre plugin pour déplacer le robot, soit en partant du package `my_custom_plugin` que vous trouverez sur le GitLab du projet, soit en partant du code source d'un plugin ROS2-gazebo comme `differential-diff-drive`.

La figure 16 est un diagramme des nodes donné par `rqt` pour comprendre l'architecture actuelle de la simulation avec notre plugin.

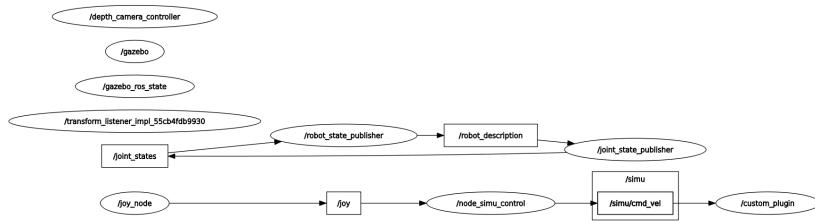


Figure 16: Diagramme des nodes ROS2

Grâce à notre plugin et nos codes, nous avons une simulation sur Gazebo qui nous permet soit de déplacer notre robot à la manette avec les mêmes contrôles que ceux sur le robot réel (cf. partie contrôle). Sinon, en appuyant sur START, on change de mode pour passer au mode autonome, c'est à dire en utilisant le commandes données par le node de guidage.

La figure 17 est une vue du BlueROV sur gazebo face à un ArUco, par la suite, le BlueROV ne sera pas tout bleu mais avec ses vraies couleurs (nous utilisons des objets en .stl alors que ce format ne prend pas en compte les couleurs, nous allons passer sur des fichiers .obj). De même l'ArUco sera en noir et blanc pour faire la détection.

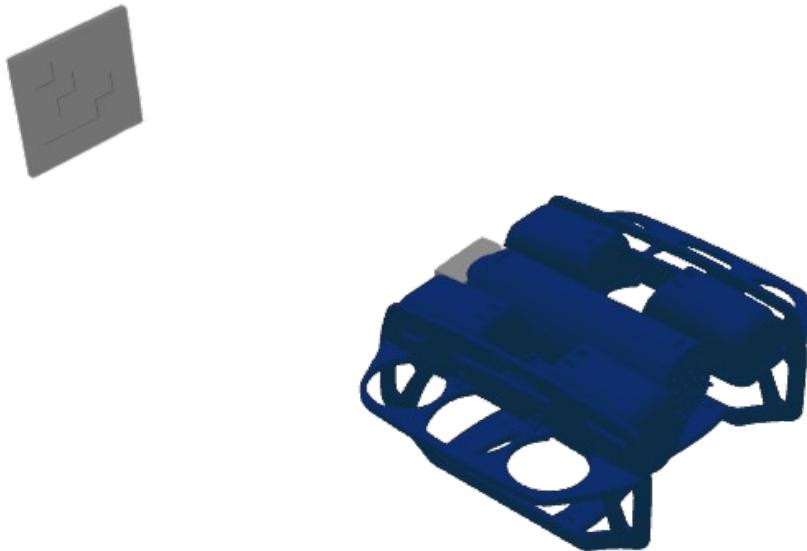


Figure 17: Simulation sous gazebo du BlueROV et un ArUco

Sur la figure 18, on voit qu'on récupère directement la vue de la caméra du robot grâce au plugin. Nous pourrons donc appliquer la détection de l'AruCo grâce à ce topic.

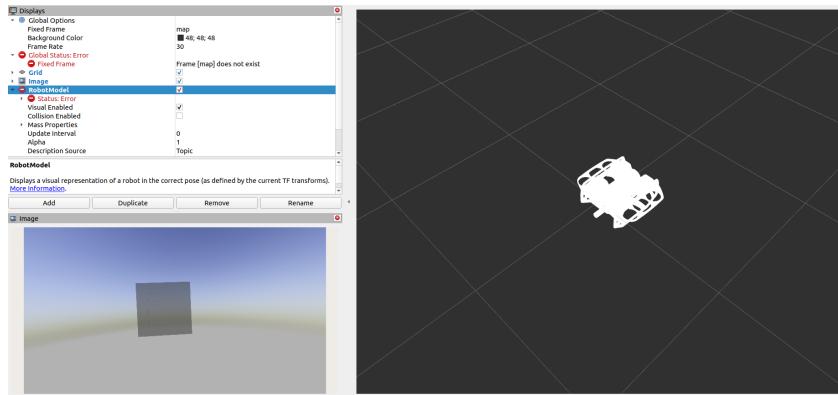


Figure 18: Représentation du BlueROV et le retour caméra sous Rviz

Le node de guidage est la suite directe de ce travail. Cette simulation nous permet de faire des tests d'où l'on veut pour créer une stratégie d'approche de l'appendice robuste. En effet, l'angle d'incidence du robot est un point critique qu'il faut pallier. Le robot doit d'abord se trouver face à la cage avant de tenter une approche s'il veut se docker efficacement.

Cette simulation nous permet également de tester plusieurs structures et agencements des marqueurs sur la cage sans avoir à refaire toute une structure mécanique à chaque fois. Ce qui nous fera gagner beaucoup de temps.

### 5.3 Résultats

Finalement , notre travail nous a permis :

- d'obtenir les images caméras du BlueROV,
- de réaliser la calibration intrinsèque de la caméra,
- de traiter l'image pour amélioration la détection sous-marine de AruCo,
- de simuler le robot sous gazebo pour réaliser le guidage.

La prochaines étapes du guidage par vision seront donc :

- de refaire la calibration intrinsèque sous l'eau,
- de réaliser les noeuds ROS2 de guidage via la vision,
- de modifier la structure de la cage pour que le robot puisse la trouver qu'importe son angle d'incidence,
- de lier le guidage par USBL et par vision à l'aide d'une machine à états finis performante.

## 6 Conclusion

### 6.1 Avancé du projet

Actuellement, nous avons une structure mécanique qui est adaptée à notre nouvelle mission, docker un appendice du robot dans l'appendice d'une structure sous l'eau. Les plans ont été réalisés et il nous reste à les tester dans des conditions réelles avec notre robot dans la piscine du LabSTICC.

Nous avons également créé les nodes ROS2 de contrôle du robot via télécommande avec MavLink. Nous avons réalisé les tests dans la piscine. Notre robot peut donc se déplacer verticalement et horizontalement tout en maintenant une parfaite stabilité en roulis et tangage. Le passage au mode autonome est prêt, lorsque le guidage autonome sera réalisé, il pourra être utilisé en appuyant sur le bouton Start de la manette.

De plus nous avons amélioré les nodes ROS2 de l'USBL créés dans des projets précédents pour qu'ils soient adaptés à notre situation. Nous les avons testés en conditions réelles dans le lac de Guerlédan et nous avons réalisé la calibration magnétique du robot.

Enfin, nous avons créé les codes permettant l'obtention de la vision du BlueROV, la calibration de la caméra et le traitement des images pour détecter la position et l'orientation du robot par rapport à des marqueurs Aruco. Des nodes ROS2 et un plugin ont aussi été créés pour réaliser une simulation du robot dans son environnement. Ce simulateur sous Gazebo nous permet de réaliser le guidage dans un monde simulé pour tester notre stratégie de guidage avant de l'utiliser dans le monde réel.

### 6.2 Suite du projet

Outre le fait que nous devons peaufinier nos travaux sur chacuns des aspects mentionnés précédemment, nous devons aussi :

- Réaliser le guidage du BlueROV par USBL,
- Réaliser le guidage du BlueROV par vision,
- Créer la stratégie de docking grâce à une machine à états finis,
- Tester la structure mécanique en condition réelles et la modifier pour s'adapter à notre stratégie d'approche.