Stage 2A

Rapport

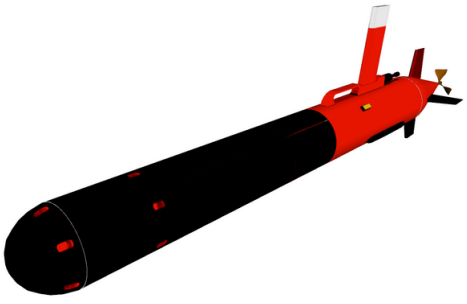
# Simulation d'un Suivi d'isobathes par un AUV

Stagiaire : Gwendal Priser

Tuteur : Luc Jaulin

Période de stage : 1 Juillet 2020 - 1 Septembre 2020

ENSTA BRETAGNE



# Contents



# List of Figures



# List of Tables

# 1 Introduction

## 1.1 Contexte du stage et objectifs

Ce stage a été effectué durant la crise du COVID-19 pendant deux mois en télétravail. Il avait pour objectifs de me former sur ROS / Gazebo afin de faire de la simulation et de la commande. C'est un domaine important de la robotique qui m'intéresse. J'ai apprécié disposer de beaucoup de libertés et d'autonomie dans la conduite de ce projet.

## 1.2 Présentation du problème

Contrairement à la partie terrestre, l'océan et le milieu aquatique demeurent mal connus. Cette méconnaissance est due à la difficulté des missions sous-marines, l'environnement marin est hostile pour l'humain. Mieux connaître l'océan est un atout dans pleins de secteurs : la pêche, la construction de ports, la connaissance de l'écosystème sous-marin, les énergies, dans les domaines de la recherche pour les études sous-marines, dans le domaine militaire pour le déminage ou la lutte anti-sous-marine.

Les premières solutions technologiques proposées consistaient à immerger des hommes avec des scaphandres puis plus tard des sous-marins. Ces solutions demandent du temps, sont risquées et coûteuses. Afin de supprimer le risque, les ROV sont apparus (Remotely Operated Underwater Vehicle), ainsi le risque humain disparaît mais cela demande du temps pour piloter le ROV, cela limite le champs d'exploration. Aujourd'hui afin de palier à ces problèmes, les AUV (Autonomous Underwater Vehicle) sont la solution. Les AUV sont contrôlés par des capteurs et des algorithmes, l'environnement marin ne permettant pas la communication sans fil ou le contrôle à distance. En effet, en autonomie, ie sans liaison avec la surface, ces robots sont capable d'effectuer une mission d'explorations. Ces AUV peuvent donc effectuer des tâches de manière plus sûre et plus efficace à un coût moindre. Les seules contraintes sont la capacité de ses batteries qui limite le temps de la mission et la robustesse des algorithmes embarqués afin d'assurer l'exécution de la mission. Ces véhicules doivent être totalement autonomes et capables de s'adapter à un environnement changeant et complexe. Avant de lancer un UUV pour effectuer une mission, il est donc nécessaire de tester son comportement dans les conditions rencontrées en mer.

Ainsi, nous nous proposons de construire un démonstrateur simulant un AUV suivant un relief sous marin. Le robot pourrait être utilisé pour obtenir des données dans une certaine zone. Nous utiliserons Gazebo que nous pourrons facilement interfacer avec le Middleware ROS qui constitue une brique amovible pouvant être associée soit au simulateur, soit au réel AUV.

L'objectif de la mission est de faire suive à un AUV une isobathe. Sur une carte marine, une isobathe est une ligne joignant des points d'égale profondeur ; c'est donc une courbe de

niveau, indiquant la profondeur d'une surface au-dessous du niveau de l'eau (figure 1). Dans notre cas, la surface est un relief sous marin. Pour ce faire l'AUV est équipé d'un sonar capable de lui donner la profondeur à laquelle est la surface en dessous lui, mais il est aussi capable de déterminer le gradient de la surface. L'AUV est aussi équipé d'un baromètre qui lui indique la profondeur à laquelle il est immergé ainsi que d'une centrale inertielle. L'AUV doit être capable de suivre l'isobathe et donc d'effectuer un cycle. Cela nous garantit de ne pas perdre l'AUV et ainsi de récupérer les données mesurées, de recharger les batteries et de l'envoyer sur une autre mission.

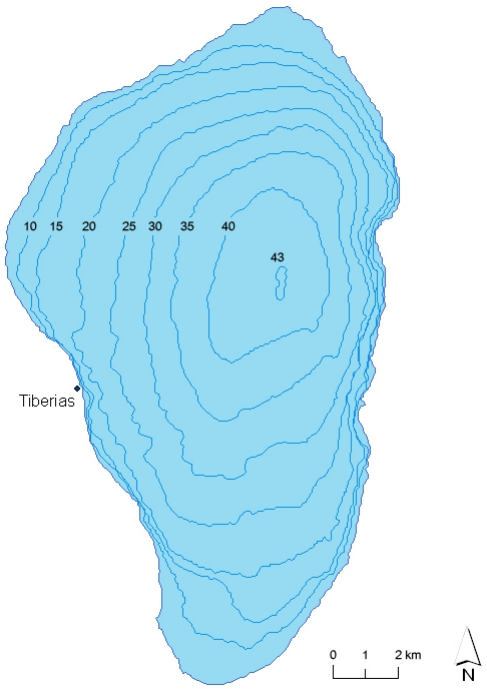



Figure 1: Représentation d'une isobathe

# 2 Modélisation cinématique de l'AUV

Afin de faciliter, nous considérons un AUV de type torpille. Cette simulation est inspirée des travaux de Luc Jaulin [1]. Pour décrire l'AUV nous considérons le vecteur d'état : $X(x, y, z, \psi)$. Le triplé $(x, y, z)$ correspond à la position dans le référentiel lié à l'océan. $\psi$ est l'angle de lacet de l'AUV autour de l'axe z.

L'équation d'état de l'AUV, $f(X, u)$ est la suivante :

$$\begin{cases} \dot{x} & = \cos(\psi) \\ \dot{y} & = \sin(\psi) \\ \dot{z} & = u_1 \\ \dot{\psi} & = u_2 \end{cases}$$

$u_1$ et $u_2$ sont les commandes qu'il faut déterminer.

Les capteurs nous donne accès à la fonction d'observation $Y = g(X)$ suivante :

$$\begin{cases} y_1 & = z - h(x, y) \\ y_2 & = \nabla h - \psi \\ y_3 & = -z \end{cases}$$

$y_1$ est donné par le sonar, $y_2$ est obtenu à la fois par l'IMU et par le sonar enfin nous obtenons $y_3$ grâce au baromètre.

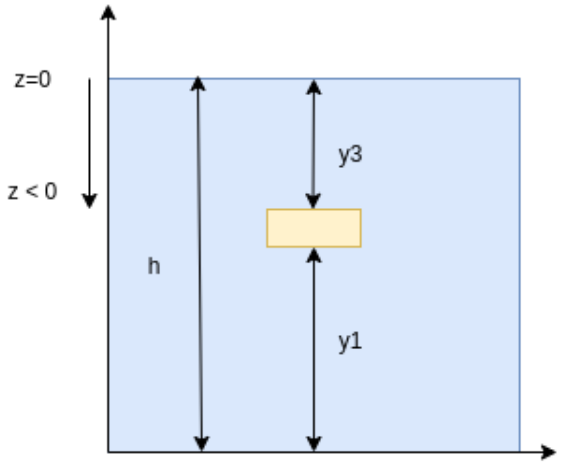



Figure 2: Définition des grandeurs

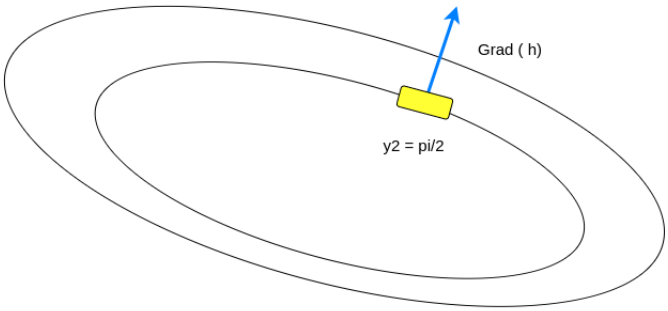



Figure 3: Définition de $y_2$

# 3 Contrôleur

Afin de piloter l'AUV, d'après l'article [1] nous pouvons proposer les commandes $u_1$ et $u_2$ suivantes :

- $u_1 = y_3 - \bar{y_3}$ de cette manière l'AUV reste toujours à la même profondeur.
- $u_2 = -\tanh(h_0 + y_3 + y_1) + \text{sawtooth}(y_2 + \frac{\pi}{2})$ L'AUV va suivre l'isobathe $h_0$, le deuxième terme permet au robot de suivre l'isobathe perpendiculairement au gradient.

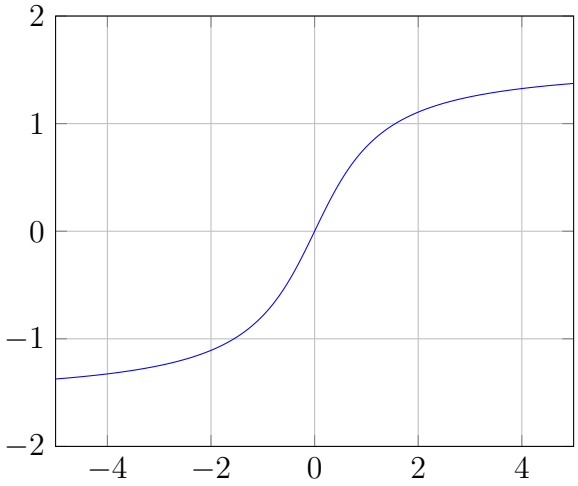



Figure 4: Contrôle de la saturation grâce à la tangente hyperbolique

Ainsi nous avons proposé des commandes proportionnelles simples qui devraient fonctionner avec le modèle cinématique. Toutefois dans le cas réel ou dans le simulateur Gazebo avec la présence potentielle de courant, nous pourrions ajouter un intégrateur afin de diminuer l'erreur statique.

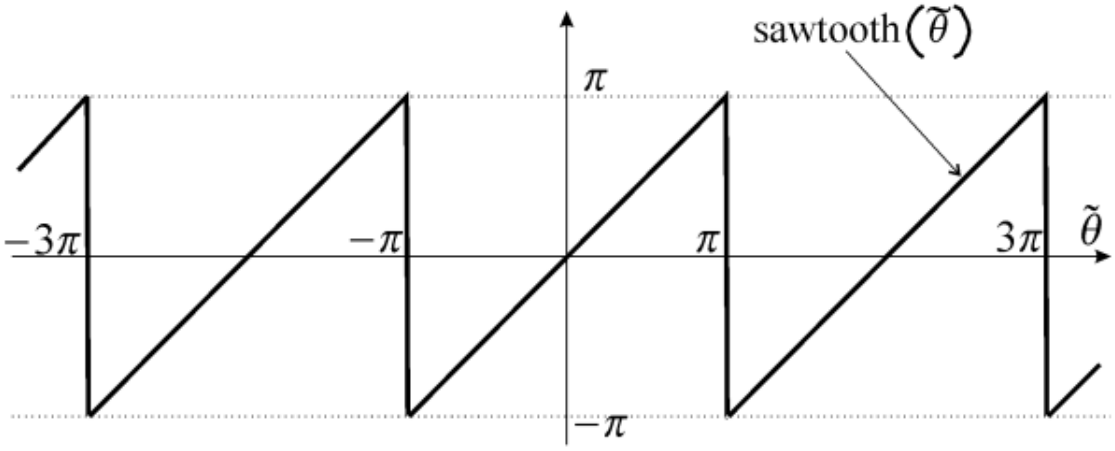



Figure 5: Fonction sawtooth qui permet de borner la commande entre $-\pi$ et $\pi$

# 4 Test du modèle

Afin de valider le modèle en cinématique ainsi que la loi de commande, nous simulons l'AUV au dessus d'un relief sous marin. Sur la figure 6, le robot se déplace à une profondeur de 3m et décrit l'isobathe des -9m , la commande semble fonctionner car le robot effectue des cycles.

La trajectoire bleue correspond à l'AUV effectuant son cycle à $z = -3m$ de profondeur, la trajectoire grise correspond à la projection sur la surface de la mer enfin la trajectoire noire correspond à la projection sur le plan contenant l'isobathe.

## 4.1 Robustesse de la commande

Le robot effectue des cycles, nous souhaitons vérifier que ses cycles sont stables. En effet, les incertitudes des capteurs peuvent éloigner le robot de sa trajectoire. Nous devons être certain que les cycles sont stables afin d'avoir une chance de récupérer le robot qui est en autonomie. En effet, dans l'eau les ondes électromagnétique sont atténuées. Il est impossible de repérer l'AUV par GPS.

Nous définissons donc un plan $P(x = 0, y, z)$ ainsi que le point $x_b$ correspondant à l'état du robot à l'intersection entre la trajectoire du robot et le plan $P$. Appelons $\phi$ la fonction, prenant en entrée la position du robot et retournant le point $x_b$. Le robot effectue des cycles stables si et seulement si le module des valeurs propres de la jacobienne de $\phi$ est strictement inférieur à 1.

En l'occurrence, la simulation nous donne 2 valeurs propres de module inférieur à 1 (0.8

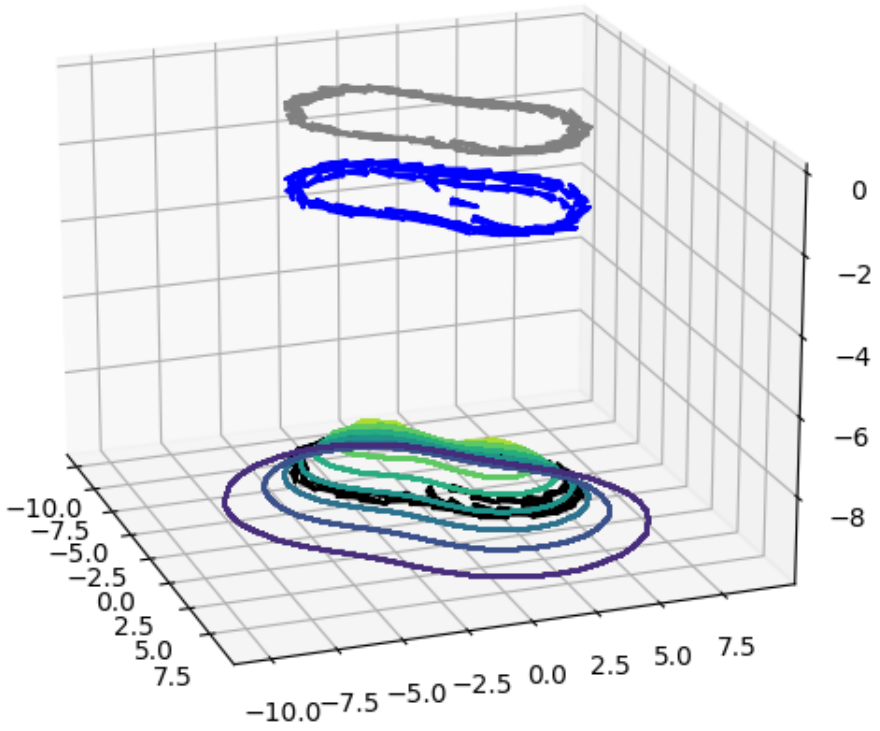



Figure 6: Simulation Python

et 0.2), autrement dit le robot revient sur sa trajectoire même s'il en est écarté : les cycles sont stables. La commande ne diverge pas.

# 5 Modèle dynamique et simulation dans Gazebo

Maintenant que le modèle a été vérifié en cinématique, on peut le tester en dynamique avec le simulateur Gazebo. Pour ce faire, nous utilisons le plugin Gazebo UUV Simulator, il propose un monde aquatique réaliste (possibilité d'ajouter du courant, viscosité de l'eau prise en compte, force de frottement). Nous choisissons comme AUV, le robot LAUV (Light Autonomous Underwater Vehicle) designé par la société portuguaise OceanScan-MST (figure 8). Vous trouverez son électronique et ses actionneurs dans cet article [2] et vous trouverez sa modélisation mécanique dans cet article [3] Nous équipons l'AUV d'une IMU, d'un baromètre ainsi que d'un sonar.

L'architecture logicielle que nous proposons est présentée sur la figure 7.

Le sonar utilisé par l'AUV LAUV, n'est pas assez performant pour pouvoir déterminer le gradient du fond sous marin. Nous devons donc l'estimer via un observateur de Kalman comme décrit dans l'article [1]. Il faut donc implémenter dans le node d'observation un

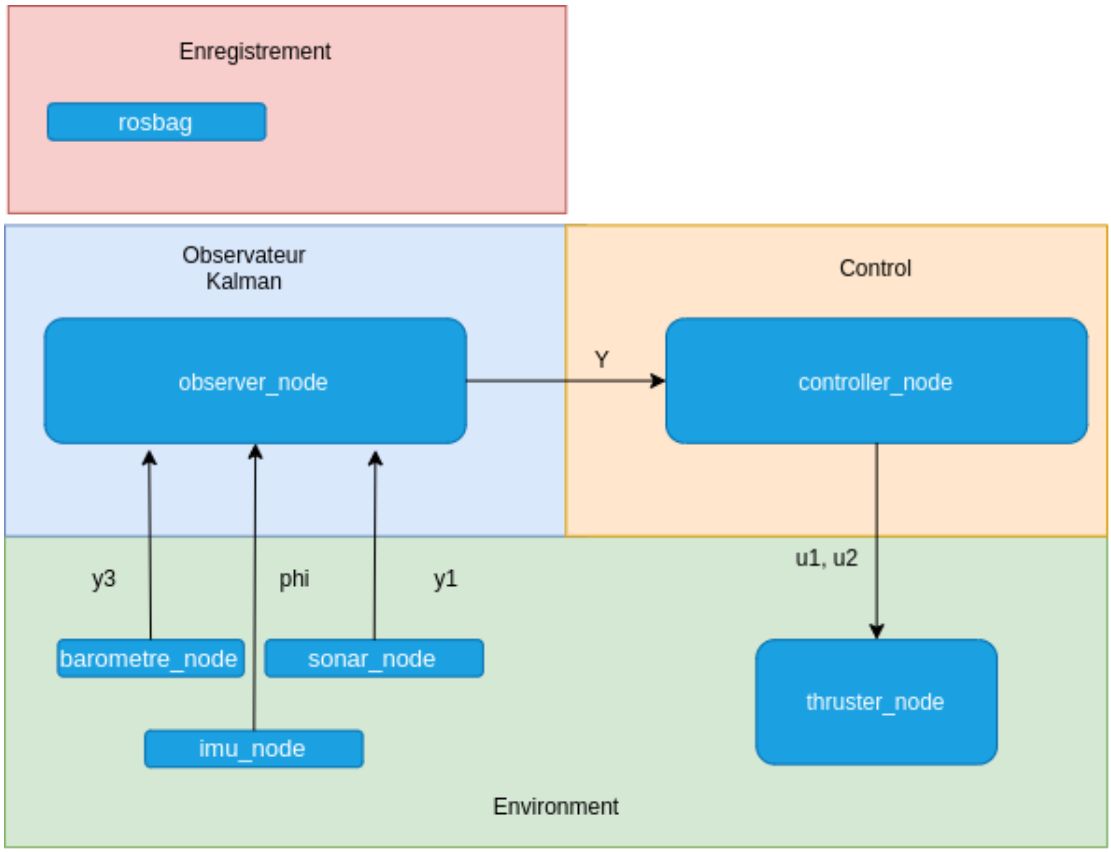



Figure 7: ROS Nodes

observateur de Kalman. Nous supposons qu'en dessous l'AUV le sol est localement un plan d'équation : $z = p_1x + p_2y + p_3$, il faut estimer le vecteur $p$ via le filtre de Kalman. $p_1$ et $p_2$ correspondes aux coordonnées du gradient. Les équations sont décrites dans l'article de Luc Jaulin [1]. Nous pouvons donc l'implémenter d'abord sur le modèle cinématique construit en Python.

```
    p = np.array([[0], [0], [0]])
    Gp = 10*np.eye(3, 3)
    Ga = np.eye(3, 3)
    Gb = 0*np.eye(1, 1)

    u = np.array([[0], [0]])

    for t in arange(0,80,dt):
        A = np.array([[1, dt*u[1, 0], 0], [-dt*u[1, 0], 1, 0], [dt, 0,
1]])
        C = np.array([[0, 0, 1]])
        y = np.array([- g(x)[0] - g(x)[2]])
        p, Gp = rl.kalman(p, Gp, np.array([0, 0, 0]).reshape(3, 1), y, Ga,
```

```
    Gb, A, C)
```

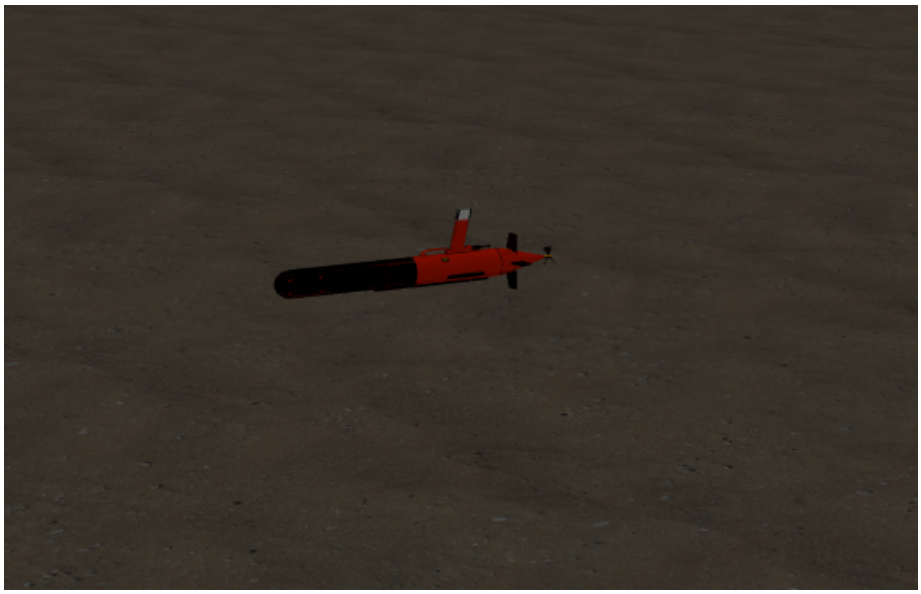


Figure 8: LAUV dans Gazebo

Ainsi nous obtenons un environnement réaliste avec un robot effectuant des cycles en suivant une isobathe. On peut voir les hélices tourner et les ailerons s'orienter. Nous pouvons perturber la mission en augmentant la vitesse du courant ou en dessinant un relief erratique.

# 6 Idées de poursuites du projet

Le repérage de l'AUV dans l'espace pour effectuer son cycle fonctionne avec les isobathes mais cela suppose des hypothèses pas toujours vérifiées. D'abord l'AUV repose sur 3 capteurs devant nécessairement fonctionner simultanément. Le relief sous-marin n'est pas nécessairement dans le champs du sonar. L'AUV est donc perdu. On pourrait donc construire une machine à état qui dans le cas d'une défaillance d'un capteur, l'AUV remonte à la surface pour capter un signal GPS et se diriger classiquement vers le bateau des opérateurs.

Autrement, nous pourrions définir une méthode de secours de parcours de cycles. Par exemple, nous pourrions disposer autour du périmètre de la zone à explorer plusieurs bouée. L'AUV les repère avec son sonar et les considère comme des points de passage à valider.

# 7 Conclusion

La mission était ambitieuse, ma méthode était de diviser le projet en plusieurs parties pour ensuite les valider les unes à la suite des autres. Il y a d'abord eu une phase d'état de l'art où j'ai analysé comment fonctionnait un AUV de type torpille. J'ai étudié l'article de Luc Jaulin [1] qui m'a donné l'ensemble des pistes nécessaires à la réalisation de la mission et au design d'un prototype en Python. Sur ce prototype, j'ai pu valider la loi de commande et m'assurer qu'elle conduisait l'AUV à effectuer des cycles stables. Une fois ces étapes terminées, j'ai pu commencer à réaliser mon environnement aquatique dans Gazebo. Après avoir essayé de construire à partir de rien la simulation avec beaucoup de difficulté, j'ai découvert le package UUV Simulator et l'ensemble des possibilités qu'il offre. Finalement ce stage fût instructif et bénéfique pour mon parcours en robotique, j'ai apprécié la variété des tâches du projet. Toutefois certains moments ont été plus difficiles, la documentation des différents plugins utilisés était souvent absentes. Mon ordinateur était parfois pas assez performant pour pouvoir pleinement utiliser le simulateur, le milieu aquatique et les capteurs simulés sont gourmands en ressources.

# 8 Annexes

## 8.1 Acronymes

ROV : Remotely Operated underwater Vehicles

AUV : Autonomous Underwater Vehicles

ROS : Robot Operating SystemaColor

IMU : Inertial Measurement Units

UUV : Unmanned underwater vehicles

## 8.2 Listing

Robustesse de la commande:

```
p_xb = phi(x, 3)[-1]
print("pxb ",p_xb)
H= 0.2
xb_dy = np.copy(p_xb)

xb_dy[1, 0] += H
p_xb_dy = phi(xb_dy, 1)[-1]

xb_dth = np.copy(p_xb)
xb_dth[2, 0] += H
```

```
p_xb_dth = phi(xb_dth, 1)[-1]

p1 = (p_xb_dy - p_xb)/H
p2 = (p_xb_dth - p_xb)/H

Jacobienne = np.hstack((p1, p2))[1:3]

print("Jacobienne : ", Jacobienne)
valp, vp = np.linalg.eig(Jacobienne)
print("Valeurs propres : ", valp)
print("Module des vp : ", np.abs(valp))
```

# References


[1] Luc Jaulin. Isobath following using an altimeter as a unique exteroceptive sensor. In *International Robotic Sailing Conference 2018 (IRSC 2018)*, Southampton, United Kingdom, August 2018.

[2] Alexandre Sousa, Luis Madureira, Jorge Coelho, José Pinto, João Pereira, João Borges Sousa, and Paulo Dias. Lauv: The man-portable autonomous underwater vehicle. *IFAC Proceedings Volumes*, 45(5):268 – 274, 2012. 3rd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles.

[3] Jorge Estrela da Silva. Modeling and simulation of the lauv autonomous underwater vehicle. *IFAC Proceedings Volumes*, 2012. 3rd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles.

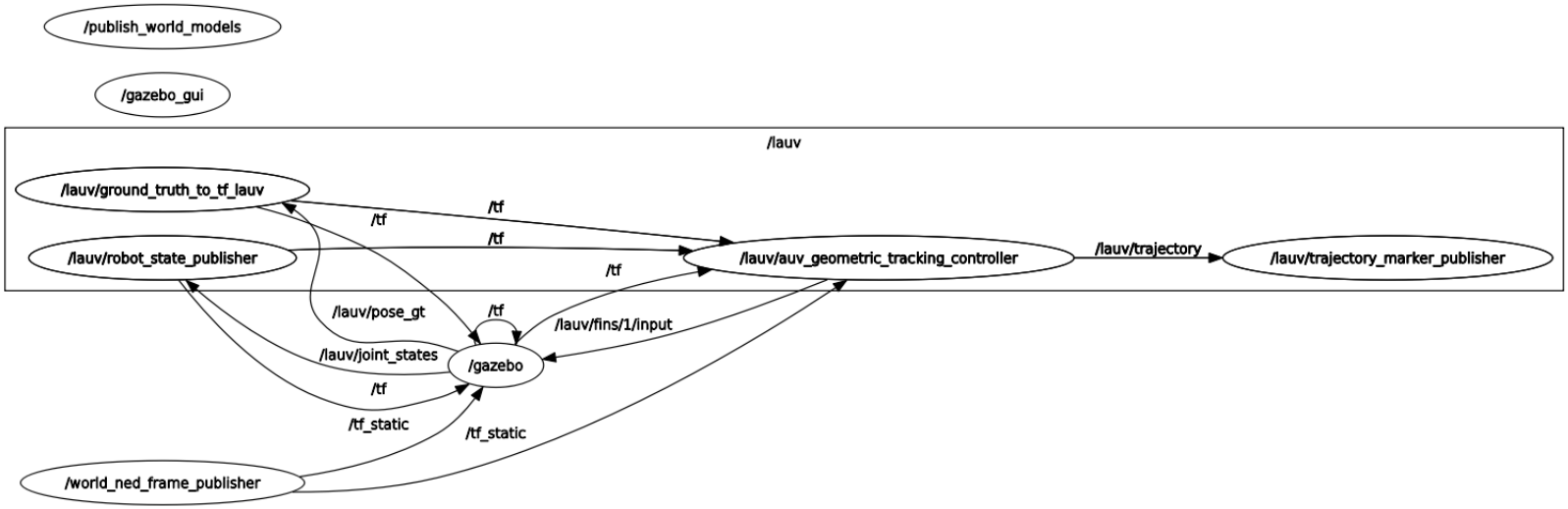



Figure 9: Node Graph