Localisation d'un ROV via son ombilical et métrologie en piscine

Louis Gillard, FISE 24

Maître de stage : Christophe Viel
Tuteur de stage : Luc Jaulin
Addresse : ENSTA Bretagne

August 29, 2024

Remerciements

Je tiens à exprimer ma profonde gratitude à Christophe Viel, mon maître de stage, pour son encadrement et ses précieux conseils, qui m'ont permis d'apprendre énormément. Je remercie également Luc, Fabrice, Erwan, Alain et Thomas de l'équipe labSTICC pour leur aide tout au long de cette expérience, dont l'expertise a été essentielle à la réussite de mon stage.

Chapter 1

Introduction

1.1 Contexte

Localiser un ROV (Remotely Operated Vehicle) sous l'eau présente des défis uniques en raison de l'incapacité des ondes radio GPS à pénétrer ce milieu. Les ROV sont des robots sous-marins conçus pour explorer des zones inaccessibles à l'homme et qui se différencient des AUV (Autonomous Underwater Vehicle) par la présence d'un câble qui les relie à la surface. Ce câble permet l'alimentation et la circulation d'informations capteurs et commandes, et même, en cas de panne, d'assurer un lien pour retrouver l'engin. Les ROV ont vu leurs usages et technologies évoluer rapidement au cours des dernières décennies. Cependant, pour accomplir des tâches telles que l'exploration, la reconstruction de cartes ou l'amarrage, un système de suivi précis est indispensable.

Un système parfois utilisé pour la localisation sous-marine est le réseau de capteurs sans fil sous-marin (Underwater Wireless Sensor Networks, UWSNs). Ces réseaux, composés de véhicules et de capteurs déployés dans une zone acoustique spécifique, permettent un suivi collaboratif et la collecte de données en temps réel. Par exemple, une méthode de localisation basée sur une infrastructure avec quatre nœuds d'ancrage a été proposée pour former un UWSN. Chaque nœud, situé à une position connue, émet des ondes électromagnétiques à des fréquences spécifiques, et la localisation des véhicules sous-marins se fait par la mesure de la force des signaux reçus (RSS).

Malgré leur utilité, les UWSNs présentent plusieurs inconvénients, notamment une bande passante limitée, un délai de propagation élevé, des contraintes de puissance, ainsi que des coûts et une complexité d'installation élevés. En conséquence, la méthode la plus répandue pour localiser un ROV est la Ultra Short Base Line (USBL). Ce système de positionnement acoustique sous-marin consiste en un émetteur-récepteur placé sous un navire et un transpondeur sur le véhicule. L'USBL calcule à la fois la distance et l'angle entre l'émetteur-récepteur et la balise sous-marine, fournissant ainsi la position du ROV. Toutefois, l'USBL offre une mise à jour de position à un faible taux et parfois avec une précision limitée. Il est donc souvent combiné avec plusieurs capteurs tels qu'une unité de mesure inertielle (IMU), une caméra, un baromètre ou

un sonar.

En plus des limitations techniques, l'utilisation d'ultrasons peut perturber la vie sous-marine en générant des nuisances acoustiques. Dans des environnements confinés comme les piscines, les ultrasons peuvent rebondir sur les parois, créant des interférences et des erreurs de mesure. Face à ces contraintes, il devient pertinent d'explorer des méthodes de localisation qui s'affranchissent des ultrasons et des systèmes acoustiques.

L'objectif de ce rapport est de détailler les travaux réalisés durant mon stage pour localiser un ROV via son ombilical et effectuer des mesures de métrologie en piscine.

1.2 Problématique

Une alternative viable pourrait reposer sur l'exploitation de la forme de l'ombilical, le câble qui permet la communication entre le ROV et l'ordinateur de contrôle en surface. La question centrale de ce projet est donc : comment peut-on utiliser la forme de l'ombilical pour déterminer la position exacte du ROV sous l'eau ?

1.3 Résumé de la solution

En capturant la forme du câble à l'aide de capteurs, nous pouvons modéliser sa forme tridimensionnelle sous l'eau. Simultanément, un capteur de profondeur embarqué sur le ROV enregistre la profondeur à laquelle se trouve le véhicule. Par ailleurs, une masse glissante se déplace le long du câble afin de lui donner une forme de "V", simplifiant énormément la modélisation mathématique. En combinant ces données sur la forme de l'ombilical et la profondeur du ROV, des calculs géométriques permettent de reconstituer la position tridimensionnelle du ROV par rapport à son point d'attache en surface. Cette approche présente plusieurs avantages par rapport aux méthodes traditionnelles, réduisant les coûts en éliminant le besoin de systèmes acoustiques complexes et minimisant l'impact sur l'environnement sous-marin. De plus, elle permet une localisation continue et précise du ROV grâce aux données en temps réel fournies par les capteurs, essentielle pour des opérations sous-marines efficaces et sécurisées.

Chapter 2

Théorie

2.1 Modèle géométrique

Le modèle géométrique est entièrement expliqué dans l'article [2].
Nous reprenons ici les démonstrations qui permettent d'obtenir la position du ROV :

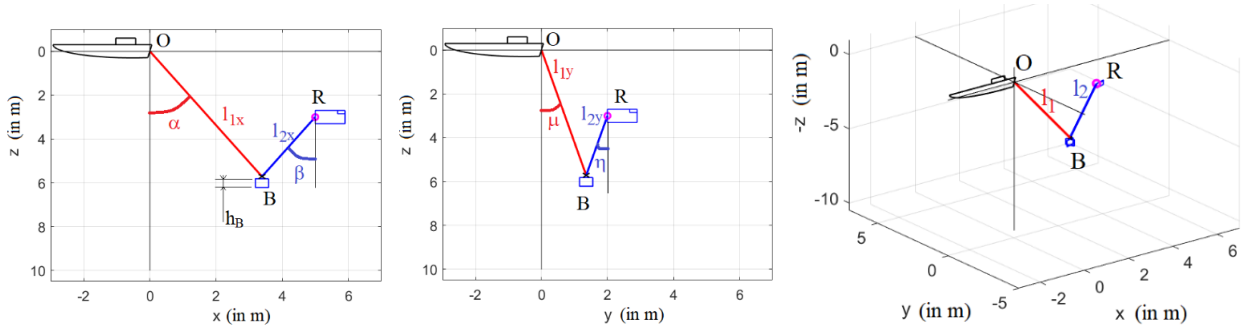


Figure 2.1: Définitions des grandeurs du problème comme présenté dans l'article [2]

Comme représenté selon plusieurs angles sur la Figure 2.1, le système se compose du câble qui relie le ROV à un point d'attache à la surface et d'une masse glissante qui est placée sur le câble. Celle-ci peut se déplacer librement le long du câble grâce à un système de poulie. Grâce à elle, la modélisation mathématique de la forme du câble est beaucoup plus simple puisqu'on obtient une forme de "V" i.e. deux lignes droites.

On note O le point d'attache, B la position de la masse glissante et R la position du ROV.
$R = [x, y, z]$, $O = [0, 0, 0]$

L est la longueur du cable et on a $L = l_1 + l_2$, avec l_1 et l_2 les distances respectives OB et BR. l_{1x} (respectivement l_{2x}, l_{1y}, l_{2y}) est la projection de l_1 (resp. l_2, l_1, l_2) dans le plan Oxz (resp. Oxz, Oyz, Oyz).

L'angle α (resp. β, μ, η) correspond à l'angle entre la projection de $\overrightarrow{OB}$ (resp. $\overrightarrow{RB}$, $\overrightarrow{OB}$, $\overrightarrow{RB}$) sur le plan Oxz (resp. Oxz, Oyz, Oyz) et l'axe vertical Oz.

Les grandeurs α, β, μ, η, z et L sont, dans tout ce rapport, considérées comme l'entrée du

système ; a_1, a_2, a_3, a_4, l_1, l_2, l_{1x}, l_{1y}, l_{2x} et l_{2y} comme les valeurs intermédiaires du système ; et x et y comme les sorties du systèmes.

On cherche donc à exprimer x et y en fonction de l'entrée du système (données capteurs).
Le système peut être modélisé de la façon suivante :

$$\begin{cases} x = l_{1x}\sin\alpha + l_{2x}\sin\beta \\ y = l_{1y}\sin\mu + l_{2y}\sin\eta \end{cases} \tag{2.1}$$

et

$$\begin{cases} z = l_{1x}\cos(\alpha) - l_{2x}\cos(\beta) \\ z = l_{1y}\cos(\mu) - l_{2y}\cos(\eta) \end{cases} \tag{2.2}$$

On a besoin d'exprimer l_{1x}, l_{1y}, l_{2x} et l_{2y} en fonction de l'entrée.
On peut constater que,

$$\begin{cases} l_1 = \sqrt{l_{1x}^2 + \sin(\mu)^2 l_{1y}^2} \\ l_2 = \sqrt{l_{2x}^2 + \sin(\eta)^2 l_{2y}^2} \end{cases} \tag{2.3}$$

Par ailleurs, $l_{1x}\cos(\alpha) = z_B = l_{1y}\cos(\mu)$ donc $l_{1x} = l_{1y}\frac{\cos(\mu)}{\cos(\alpha)}$
De même, $l_{2x}\cos(\beta) = l_{2y}\cos(\eta)$ donc $l_{2x} = l_{2y}\frac{\cos(\eta)}{\cos(\beta)}$
En injectant ces expressions de l_{1x} et l_{2x} dans (2.3), on obtient :

$$\begin{cases} l_{1x} = \frac{l_1}{\sqrt{1+\tan(\mu)^2\cos(\alpha)^2}} \\ l_{1y} = \frac{l_1}{\sqrt{(\sin\mu)^2+\left(\frac{\cos\mu}{\cos\alpha}\right)^2}} \\ l_{2x} = \frac{l_2}{\sqrt{1+(\tan\eta)^2(\cos\beta)^2}} \\ l_{2y} = \frac{l_2}{\sqrt{(\sin\eta)^2+\left(\frac{\cos\eta}{\cos\beta}\right)^2}} \end{cases}$$

D'où, en notant :

$$\begin{cases} a_1 = \sqrt{1+(\tan\mu)^2(\cos\alpha)^2} \\ a_2 = \sqrt{1+(\tan\eta)^2(\cos\beta)^2} \end{cases} \quad \begin{cases} a_3 = \sqrt{(\sin\mu)^2+\left(\frac{\cos\mu}{\cos\alpha}\right)^2} \\ a_4 = \sqrt{(\sin\eta)^2+\left(\frac{\cos\eta}{\cos\beta}\right)^2} \end{cases} \tag{2.4}$$

on obtient :

$$\begin{cases} l_{1x} = \frac{l_1}{a_1} \\ l_{1y} = \frac{l_1}{a_3} \\ l_{2x} = \frac{l_2}{a_2} \\ l_{2y} = \frac{l_2}{a_4} \end{cases} \tag{2.5}$$

Puis, en utilisant la seconde équation de (2.2) :
$z = \frac{l_1 \cos(\mu)}{a_3} - \frac{l_2 \cos(\eta)}{a_4}$

Or,
$L = l_1 + l_2$

donc
$z = \frac{L \cos(\mu)}{a_3} - l_2 \left(\frac{\cos(\mu)}{a_3} + \frac{\cos(\eta)}{a_4} \right)$

Ainsi,

$$l_2 = \frac{\frac{L \cos \mu}{a_3} - z}{\frac{\cos \mu}{a_3} + \frac{\cos \eta}{a_4}} \tag{2.6}$$

et

$$l_1 = L - l_2 = \frac{\frac{L \cos \eta}{a_4} + z}{\frac{\cos \mu}{a_3} + \frac{\cos \eta}{a_4}} \tag{2.7}$$

Ou encore, en reprenant les calculs mais en choisissant la première équation de (2.2) :

$$\begin{cases} l_1 = \frac{\frac{L \cos \beta}{a_2} + z}{\frac{\cos \alpha}{a_1} + \frac{\cos \beta}{a_2}} \\ l_2 = \frac{\frac{L \cos \alpha}{a_1} - z}{\frac{\cos \alpha}{a_1} + \frac{\cos \beta}{a_2}} \end{cases} \tag{2.8}$$

De cette manière, on a exprimé l_1 et l_2 en fonction de l'entrée. Ce qui termine les calculs puisque le système d'équations (2.5) s'exprime maintenant entièrement en fonction de l'entrée et de même pour (2.1). On a exprimé x et y en fonction de l'entrée i.e. des données capteurs.

2.2 Modèle par intervalles

Mon apport théorique se fait essentiellement au travers de cette approche par intervalles. L'utilisation de la méthode par intervalles se justifie par le fait qu'on souhaite avoir une garantie de la position déduite pour pouvoir assurer des manoeuvres plus ou moins périlleuses maglré le manque de précision des capteurs. Cependant, on ne sait pas a priori si cette méthode aboutira à des résultats suffisamment précis pour être utilisable en pratique.

2.2.1 Analyse par intervalle

Il s'agit d'une méthode qui fait des calculs avec des intervalles au lieu de nombres réels afin d'obtenir des résultats plus rigoureux. Ainsi, toutes les erreurs d'arrondis, de précision de capteurs, etc... sont prises en compte tout au long du calcul.

A partir des intervalles on peut définir une boîte qui est un vecteur d'intervalles de dimension quelconque.

Sur les domaines en question, on peut appliquer des contracteurs qui sont des opérateurs mathématiques qui permettent de réduire la taille de ces domaines sans pour autant perdre de solution potentielle.

En pratique, on associe à toutes nos variables un intervalle ou une boîte dont la taille dépend de la précision des capteurs et des conditions initiales du système. Puis, les contracteurs définis grâce aux équations qui régissent le système, viennent retirer des solutions au fur et à mesure, compte tenu des relations entre les variables du système. On obtient ainsi pour chaque variable un ensemble restreint de valeurs possibles.

L'intérêt principal de cette méthode est la garantie de conserver toutes les solutions possibles au cours du calcul, quitte à en conserver trop (par pessimisme).

Un exemple de contraction est illustré sur la Figure 2.2.

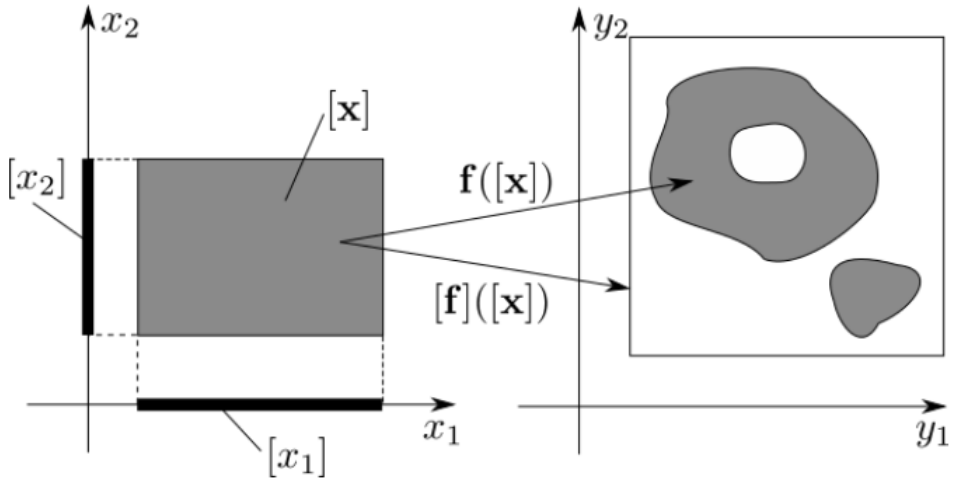


Figure 2.2: Illustration de contraction où $[x]$ est une boîte de dimension 2, $f([x])$ est l'ensemble image du domaine $[x]$ par la fonction f et $[f]([x])$ est une boîte contenant toutes les solutions de $f([x])$

2.2.2 Contracteurs

Un contracteur est une fonction f de la forme $f(x) = 0$ où x est un vecteur de dimension quelconque. On obtient donc des contracteurs à partir des formules définies en section 2.1 en soustrayant le membre de droite pour obtenir par exemple : $x - l_{1x} \sin \alpha - l_{2x} \sin \beta = 0$.

Les grandeurs l_1 et l_2 peuvent être calculées de deux manières différentes. Il est important de construire le contracteur de l_1 et l_2 pour chaque méthode puisque les contracteurs ne sont pas redondants, augmentant ainsi la précision du résultat (ou permettant au moins de converger plus vite vers la solution fixe à chaque instant).

Par ailleurs, on est en mesure de déterminer la position de la masse glissante grâce au système d'équations suivant :

$$\begin{cases} x_{masse} = l_{1x} \sin(\alpha) \\ y_{masse} = l_{1y} \sin(\mu) \\ z_{masse} = l_{1x} \cos(\alpha) = l_{1y} \cos(\mu) \end{cases}$$

On en déduit ainsi, de la même manière que précédemment, un nouveau contracteur pour la masse glissante.

Enfin, par définition d'une ellipse et la longueur du câble étant constante, on sait que la position de la masse glissante se trouve sur une ellipse dont les foyers sont le point d'attache du câble et la position courante du ROV, comme représenté sur la Figure 2.3. Cette ellipse est définie comme suit :

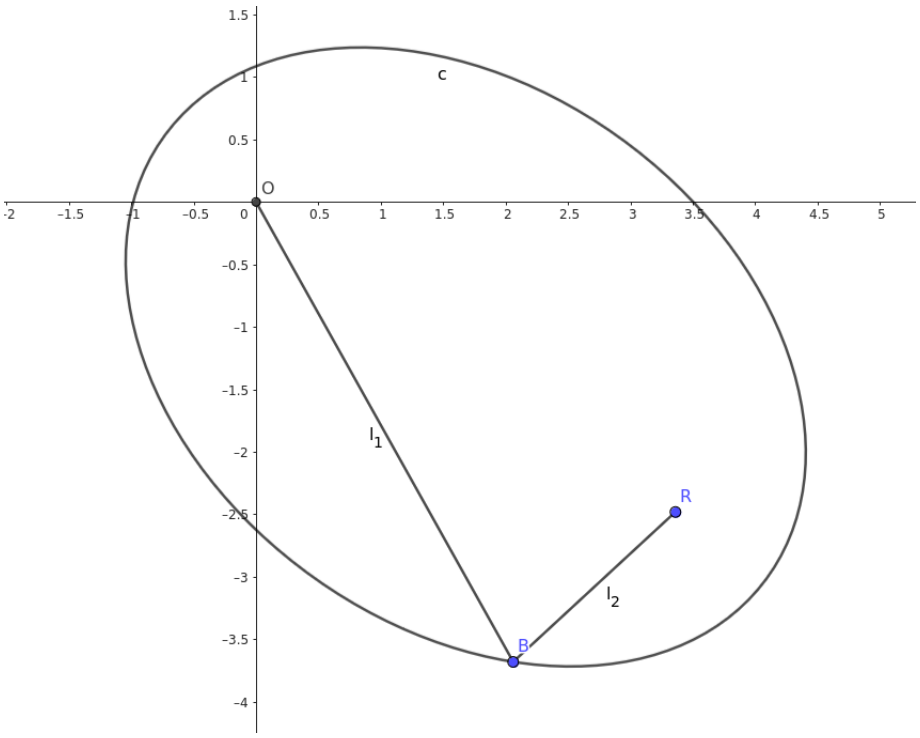


Figure 2.3: Représentation de l'ellipse des positions possibles de la masse glissante B

$$\begin{cases} \frac{4\left(\left(x_{\text{masse}}-\frac{x}{2}\right)\cos(\theta_x)+\left(z_{\text{masse}}-\frac{z}{2}\right)\sin(\theta_x)\right)^2}{L^2} + \frac{4\left(\left(x_{\text{masse}}-\frac{x}{2}\right)\sin(\theta_x)-\left(z_{\text{masse}}-\frac{z}{2}\right)\cos(\theta_x)\right)^2}{L^2-(x^2+z^2)} = 1 \\ \frac{4\left(\left(y_{\text{masse}}-\frac{y}{2}\right)\cos(\theta_y)+\left(z_{\text{masse}}-\frac{z}{2}\right)\sin(\theta_y)\right)^2}{L^2} + \frac{4\left(\left(y_{\text{masse}}-\frac{y}{2}\right)\sin(\theta_y)-\left(z_{\text{masse}}-\frac{z}{2}\right)\cos(\theta_y)\right)^2}{L^2-(y^2+z^2)} = 1 \end{cases}$$

avec $\theta_x = \arctan(x/z)$ et $\theta_y = \arctan(y/z)$

Ces équations définissent ainsi le dernier contracteur utilisé.

Tous les contracteurs sont par la suite ajoutés les uns à la suite des autres dans le contractor network qui a pour fonction d'appliquer chaque contracteur en boucle jusqu'à obtenir une solution fixe.

La Figure 2.4 donne comme exemple le contracteur pour x et y qui correspond donc au système d'équations 2.1. Celui-ci est codé en python grâce à la bibliothèque codac qui implémente le calcul par intervalles avec l'ensemble des outils de contraction nécessaires. On peut voir de la ligne 5 à la ligne 14 l'ensemble des variables d'entrée et de sortie confondues. La ligne 15 défini la fonction f à images dans $\mathbb{R}^2$, telle que $f(X) = 0_2$ avec X les variables précédentes.

```python
import codac as cdc

ctc_xy = cdc.CtcFunction(
    cdc.Function(
        "alpha", # Ensemble des variables d'entrée et de sortie
        "beta",
        "mu",
        "eta",
        "l1x",
        "l1y",
        "l2x",
        "l2y",
        "x",
        "y",
        "(l1x * sin(alpha) + l2x * sin(beta) - x ; l1y * sin(mu) + l2y * sin(eta) - y)", # Relations entre les variables
    )
)
```

Figure 2.4: Exemple de contracteur codé en python

2.2.3 SIVIA

Les contracteurs définis précédemment ne sont malheureusement pas optimaux dans le sens où la solution trouvée est très pessimiste. Pour résoudre ce problème, on peut appliquer l'algorithme SIVIA (Set Inverter via Interval Analaysis) qui consiste à paver l'espace de départ (c.f. Figure 2.5) et à appliquer les contracteurs sur chacune des boîtes issue du pavage.

De ce fait, on affine le résultat et on est capable d'éliminer certaines des solutions produites par ce pessimisme superflux.

L'algorithme SIVIA de la bibliothèque codac ne permet que d'appliquer un seul contracteur dans la boucle principale. J'ai donc adapté l'algorithme pour qu'il puisse également prendre en paramètre un contractor network.

La condition d'arrêt de l'algorithme est déterminée par un paramètre ε que l'on va comparer à la taille de la boîte considérée à chaque appel récursif de l'algorithme. Si la boîte est plus grande que ε, on la coupe en deux selon sa dimension la plus grande et on répète l'algorithme sur chaque morceau. Sinon, on s'arrête. Ainsi, l'algorithme se termine, lorsque la taille de la plus grande boîte tend vers 0 (divisions successives par 2).

L'inconvénient principal de cet algorithme est le coût temporel qui est en $O(x^n)$ où n est la dimension de l'espace de départ. Le choix de ε est donc important puisqu'un compromis entre le temps d'exécution et l'affinage du résultat apparaît.

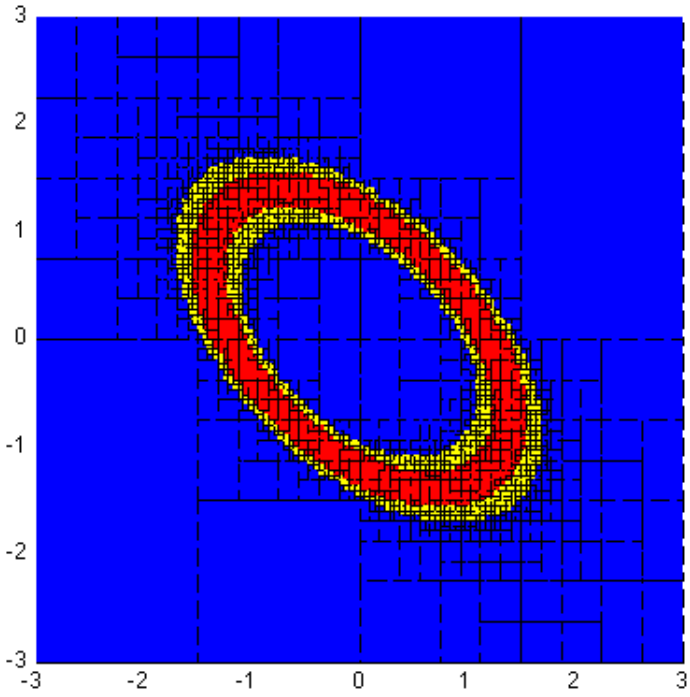


Figure 2.5: Exemple de pavage pour l'algorithme SIVIA provenant de [1]

2.2.4 Résultats avec jeu de données

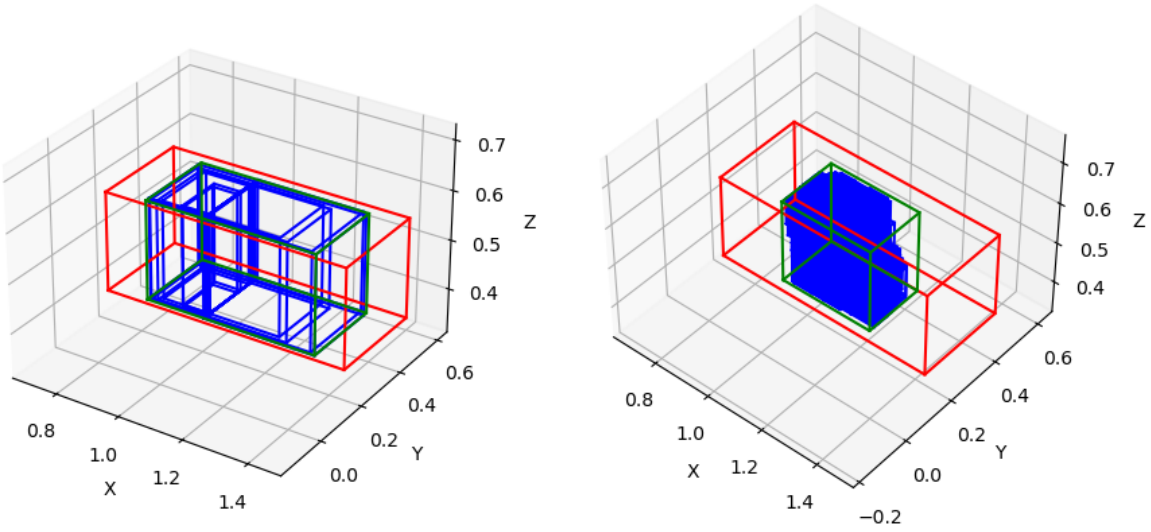


Figure 2.6: Algorithme SIVIA avec paramètre ε égal à 0.35 à gauche et égal à 0.1 à droite

Un jeu de données m'a été transmis, provenant d'un setup équivalent et contenant également des données sur la position x et y. Ce jeu de données permet donc de réaliser le calcul de x et y par le biais des intervalles, tout en permettant de vérifier le résultat obtenu.

La Figure 2.6 montre deux exemples de la fonction de calcul d'intervalles prise à un instant quelconque du jeu de données. Les mêmes variables d'entrée sont utilisées pour les deux exemples, la seule différence étant le paramètre ε choisi pour l'algorithme SIVIA. Pour plus de détail on peut se référer à la Figure 2.7. Le pavé rouge correspond au résultat sans l'algorithme SIVIA, les pavés bleus sont les résultats de chaque élément issu du pavage et le pavé vert correspond au plus petit pavé qui englobe l'ensemble des pavés bleus. L'union des pavés bleus s'approche donc de la solution réelle pour un ε qui tend vers 0. On fait le choix pessimiste de se contenter du pavé vert englobant par simplicité. On constate que le volume économisé grâce au SIVIA n'est pas négligeable et c'est d'autant plus vrai pour l'exemple de droite. Cependant, l'exécution de l'algorithme prend beaucoup plus de temps pour l'exemple de droite que pour l'exemple de gauche.

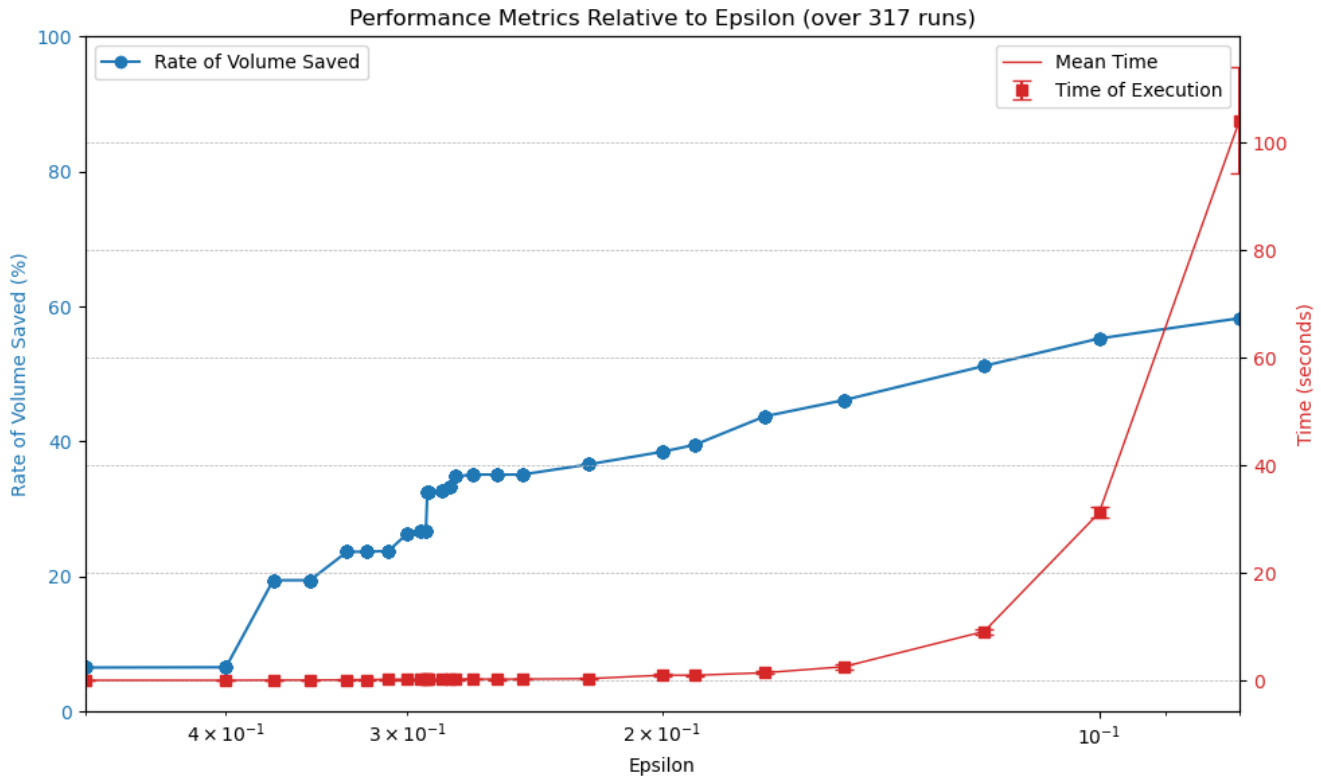


Figure 2.7: Graphe du pourcentage de volume économisé et du temps d'exécution en fonction du paramètre ε sur une même entrée

La Figure 2.7 présente le volume économisé en pourcentage et le temps d'exécution du programme pour les mêmes variables d'entrée mais pour des ε différents. En particulier, pour les deux exemples de la Figure 2.6, on peut lire que l'exemple de gauche a fait économiser

19,4% de volume grâce au SIVIA pour un temps d'exécution de 0,11 sec, tandis que l'exemple de droite a fait économiser 55,3% de volume mais pour un temps d'exécution de plus de 31 sec.

Ce graphe révèle que l'algorithme SIVIA est pertinent dans notre cas puisqu'on arrive à économiser autour de 30% de volume pour une exécution de moins de 1 sec de l'algorithme. Pour le programme principale, on choisit donc un ε à 0.278 qui conduit à un temps d'exécution d'environ 0.3 sec et une économie de 34,8% de volume. Avec ce choix, on arrive à avoir une actualisation de la position du ROV toutes les secondes en prenant en compte l'outil graphique qui tourne en parallèle du programme principal pour afficher la position en 3D.

Remarque : il se trouve que le contracteur de l'ellipse conduit parfois à une boîte resultante vide c'est à dire qu'aucune solution n'est trouvée. Pour autant, le contracteur ne réduit pas la taille de la boîte résultante, suggérant une redondance parmi les contracteurs. J'ai donc choisi de le retirer de la liste des contracteurs.

Chapter 3

Réalisation expérimentale

3.1 Description de la solution

On tente ici de réaliser un dispositif pour reproduire l'expérience théorique, comme montré sur la Figure 3.1.

Pour réaliser ce système, le choix s'est porté sur un caisson étanche contenant toute l'électronique nécessaire que l'on peut par la suite attacher au ROV afin de déterminer sa position. De cette manière, on obtient un système indépendant qui fonctionne avec n'importe quel dispositif.

Du côté ROV on a besoin d'un capteur de pression pour déterminer la profondeur, et d'une IMU à fixer sur le câble et à étanchéifier pour mesurer la direction du segment de câble correspondant (segment $[BR]$ de la Figure 2.1). Pour récupérer ces données on utilise une raspberry PI 3 modèle B V1.2 qui communiquera avec l'ordinateur en surface par le biais du câble ombilical. Une seconde IMU est nécessaire du côté PC pour mesurer la direction du second segment de câble (segment $[OB]$ de la Figure 2.1) de la même manière.

On utilise le firmware ROS 2 pour gérer les communications entre les différents blocs du programme donc Ubuntu est nécessairement installé sur la carte SD de la Raspberry PI.

Afin de fixer l'IMU sur le câble, un modèle 3D de boîtier a été imaginé comme décrit par la suite.

Enfin, lorsque l'ordinateur en surface calcule la position, il affiche également une simulation du setup en temps réel pour avoir un retour visuel direct de l'expérience en cours.

3.2 Les IMUs

Pour mesurer les quatre angles de notre problème, on utilise deux IMUs placées sur chaque extrémité du câble.

Les IMUs choisies sont les SparkFun OpenLog Artemis qui ont un accéléromètre, un gyromètre et un magnétomètre avec une précision de 1° en cap d'après le constructeur (précision qui définira la taille des boîtes correspondantes en entrée).

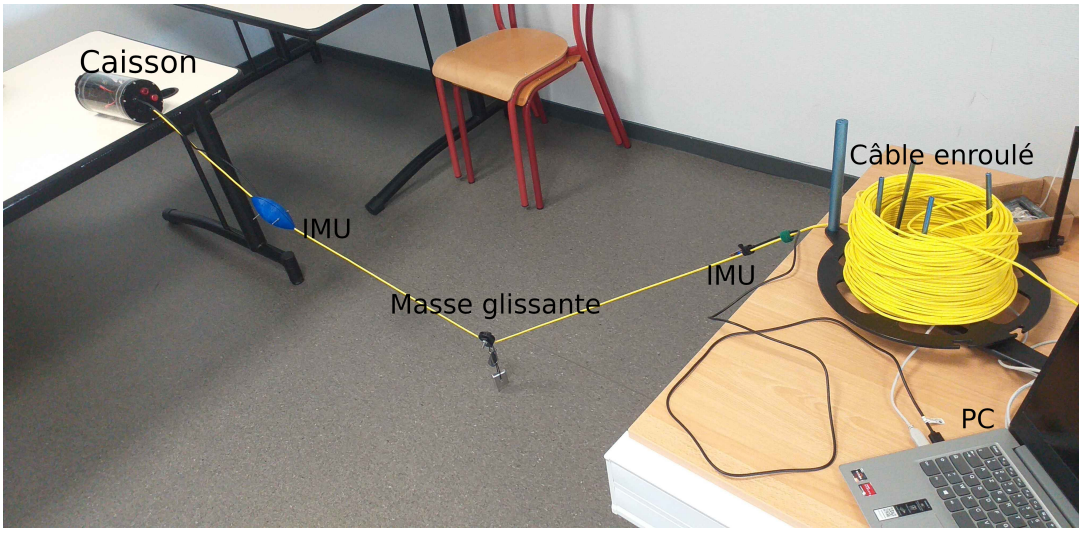


Figure 3.1: Setup complet du système de localisation

Figure 3.2: IMU SparkFun OpenLog Artemis

3.2.1 Driver et calibration

L'IMU prévoit un petit logiciel qui permet de flasher le firmware avec les bons paramètres de calibration du magnétomètre. On peut trouver ces paramètres avec n'importe quelle méthode classique de calibration. Cependant, pour pouvoir flasher le firmware, il faut assurer une connexion USB. Or, dans notre cas, on a choisi de couler l'IMU dans de la résine époxy (voir Figure 3.7). Ainsi, on se retrouve obligé d'étanchéifier un câble USB, ce qui n'est pas mince affaire. Au lieu de cela, on peut se connecter à l'IMU via les ports séries et envoyer les paramètres de calibration par le connecteur RX de l'IMU. Le problème étant qu'à chaque redémarrage de l'IMU, elle se réinitialise et utilise les paramètres inscrits sur le firmware. L'idée est donc d'utiliser un driver qui, à l'exécution, va récupérer les paramètres de calibration que l'on aura stockés dans un fichier texte sur l'ordinateur (ou sur la raspberry pi) et de les envoyer à l'IMU.

Pour réaliser la calibration, j'ai implémenté deux noeuds ROS. Le premier permet d'enregistrer en continu les données brutes de l'IMU dans une liste. Puis, lorsque suffisamment

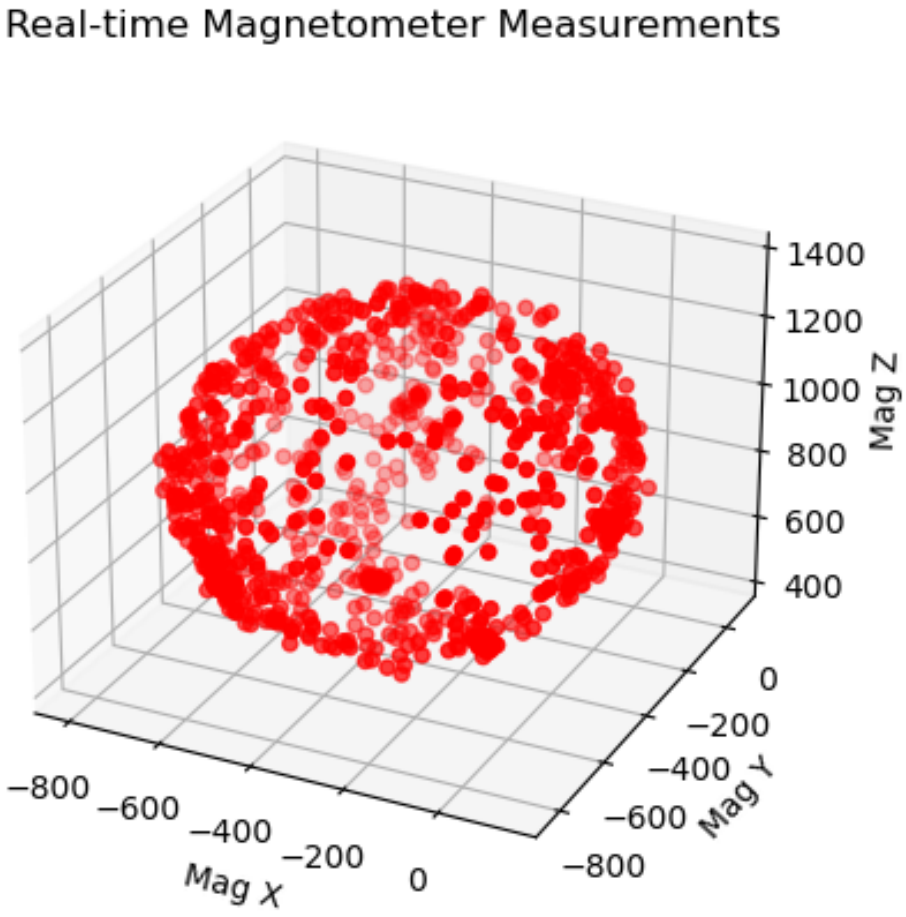


Figure 3.3: Sphère produite pendant la calibration de l'IMU

de points ont été enregistrés, un calcul est effectué pour translater et réorienter l'ensemble des points afin d'obtenir une sphère unitaire. On obtient donc un vecteur de translation et une matrice de rotations qui définissent les 12 paramètres de calibration que l'on vient enregistrer dans un fichier texte.

Le deuxième noeud ROS permet, lui, de visualiser les points en 3D au fur et à mesure qu'on les enregistre. Ce dernier noeud est optionnel mais il permet à l'utilisateur de se rendre compte d'où il se trouve dans le processus de calibration (si par exemple la "sphère" a une zone dans laquelle peu de points ont été enregistrés) comme illustré sur la Figure 3.3.

Pour calibrer l'IMU connectée à la raspberry pi, on peut lancer le premier noeud sur la raspberry pi puis le deuxième noeud sur le PC.

3.3 Architecture ROS

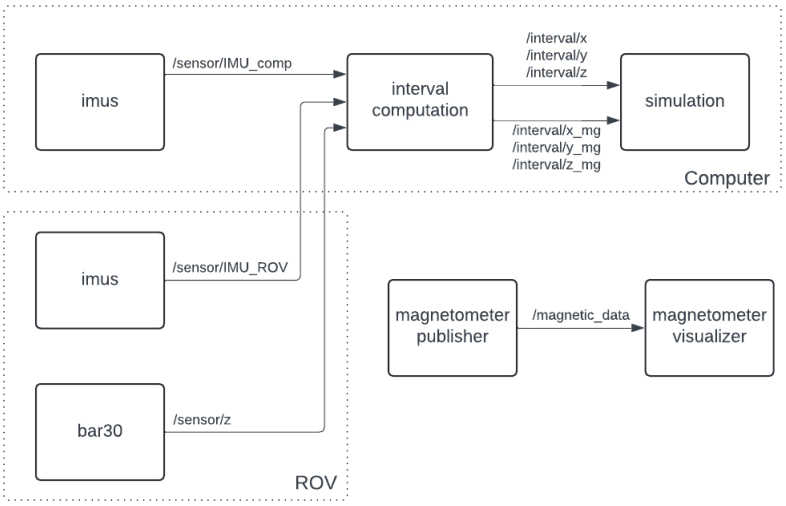


Figure 3.4: Diagramme de l'architecture ROS

La Figure 3.4 illustre les relations publisher → subscriber entre les noeuds ROS réalisés.

La raspberry pi embarquée n'est pas puissante donc on se contente d'exécuter seulement les noeuds des capteurs embarqués, à savoir, le capteur de pression et l'IMU sur celle-ci.

Du côté ordinateur, on a également une IMU donc on exécute une seconde fois le même noeud IMU.

Ensuite, le noeud ROS responsable du lourd calcul d'intervalles est interval_computation. Celui-ci récupère les messages envoyés par les 3 noeuds des capteurs et en déduit un IntervalVector de dimension 3 pour la position du ROV et un second pour la position de la masse glissante.

Enfin, ces intervalles sont envoyés sur 6 topics différents (un pour chaque dimension des deux vecteurs) au noeud simulation. Ce dernier permet de visualiser sur un outil graphique 3D, les positions calculées précédemment des éléments du système.

Enfin, un couple de noeuds responsables de la calibration des IMUs communiquent ensemble. Le publisher récupère les données brutes de l'IMU sous la forme d'un vecteur de dimension 3, publie ce vecteur sur /magnetic_data, puis à la terminaison du programme, calcul les paramètres de calibration.

Le subscriber s'occupe d'afficher les vecteurs reçus sur un outil de visualisation 3D.

3.4 Architecture électronique

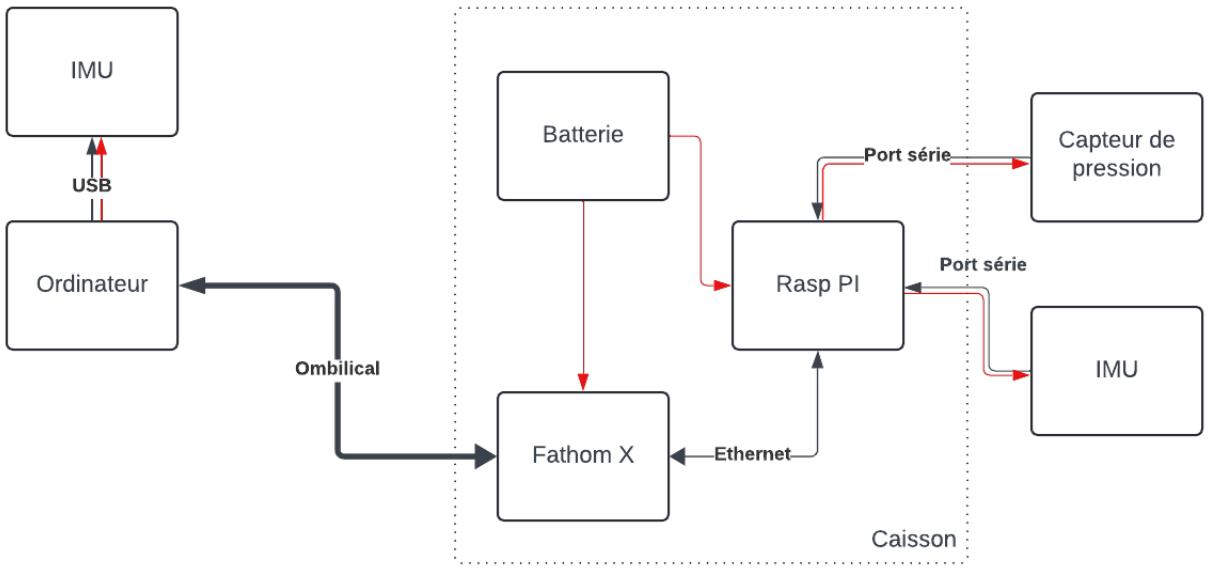


Figure 3.5: Diagramme de l'architecture électronique

Dans la Figure 3.5, les flèches rouges représentent l'alimentation tandis que les flèches noires correspondent aux transferts de données.

On a deux ensembles qui ont chacun leur propre alimentation.

D'abord, l'ordinateur alimente et communique avec son IMU par USB.

Puis, du côté caisson, une batterie 3S à 2,2A est la source d'énergie de l'ensemble des composants. Celle-ci alimente directement la raspberry pi et la carthe Fathom X qui permet, elle, de convertir les données de l'ombilical en données ethernet (et vice versa). La raspberry pi alimente ensuite les deux capteurs qui lui renvoient des données.

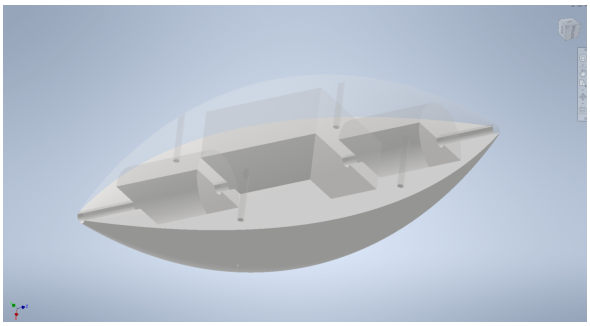
3.5 Le boîtier

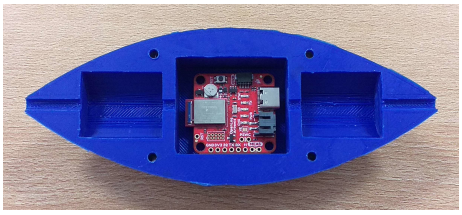
Pour fixer les IMUs le long du câble, plusieurs contraintes étaient à prendre en compte :

- être capable de déplacer l'IMU d'une mission à l'autre pour pouvoir changer la longueur du câble
- ne pas réduire l'hydrodynamisme du câble
- ne pas gêner le déplacement de la masse glissante le long du câble
- être neutre pour éviter de modifier la forme que décrit le câble

- étanchéifier l'IMU

La solution retenue pour respecter toutes ces contraintes est le boîtier suivant, réalisé par impression 3D :





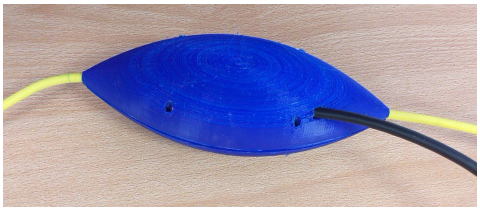


Figure 3.6: Modèle 3D imprimé au-dessus, demi-coque à gauche et coque entière fixée sur le câble à droite

Il se compose de deux demi-coques symétriques qui viennent se fixer autour du câble (le câble jaune sur la Figure 3.6) à l'aide d'écrous.

La cavité centrale vient accueillir l'IMU tandis que les deux autres servent à ajouter des masses ou de la mousse pour pouvoir équilibrer le boîtier.

On peut facilement déplacer le boîtier le long du câble en l'ouvrant.

La forme d'ogive permet d'augmenter l'hydrodynamisme de la coque afin de ne pas perturber les mouvements du câble.

En ce qui concerne l'étanchéité, l'IMU est coulée dans de la résine époxy comme le montre la Figure 3.7, ce qui la condamne malheureusement à cet usage. En revanche, cela évite la conception d'un nouveau caisson étanche qui aurait ajouté un travail conséquent et imposerait une reconsidération des précédentes contraintes.

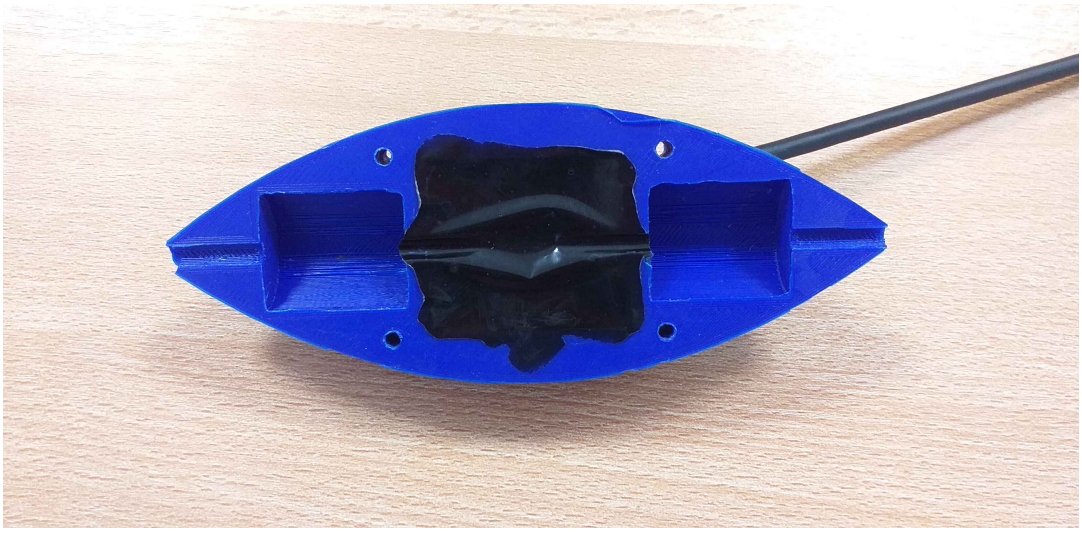


Figure 3.7: Coque avec IMU piégée dans de la résine époxy

Enfin, l'IMU communique avec la raspberry pi au moyen d'un câble étanche (le câble noir sur la Figure 3.7) qui traverse une des coques.

3.6 Simulation 3D

Le rôle du simulateur 3D est de donner un rendu visuel des calculs effectués afin de voir une correspondance en temps réel avec l'expérience en cours. Celui-ci devait afficher les éléments important du setup, tout en assurant un rendu le plus fluide possible.

Initialement, l'outil choisi était VIBes Viewer puisque la bibliothèque codac est pensée pour fonctionner avec VIBes Viewer. Cependant, l'outil ne propose qu'un rendu 2D, alors que dans notre problème les 3 dimensions de l'espace sont importantes. Il était donc plus adapté de transitionner vers un outil comme matplotlib qui permet de créer des plots 3D.

On repère plusieurs éléments sur le simulateur capturé sur la Figure 3.8:

- Le câble en forme de "V" qui se compose simplement de deux lignes
- La boîte rouge qui représente l'ensemble des positions possibles du ROV après SIVIA
- La boîte bleue qui représente l'ensemble des positions possibles de la masse glissante

Deux paramètres sont à ajuster en fonction de l'expérience réalisée :

- La longueur totale de câble entre le point d'attache et le ROV
- L'offset en hauteur du point d'attache si l'on ne peut pas mettre le point d'attache directement au niveau de l'eau

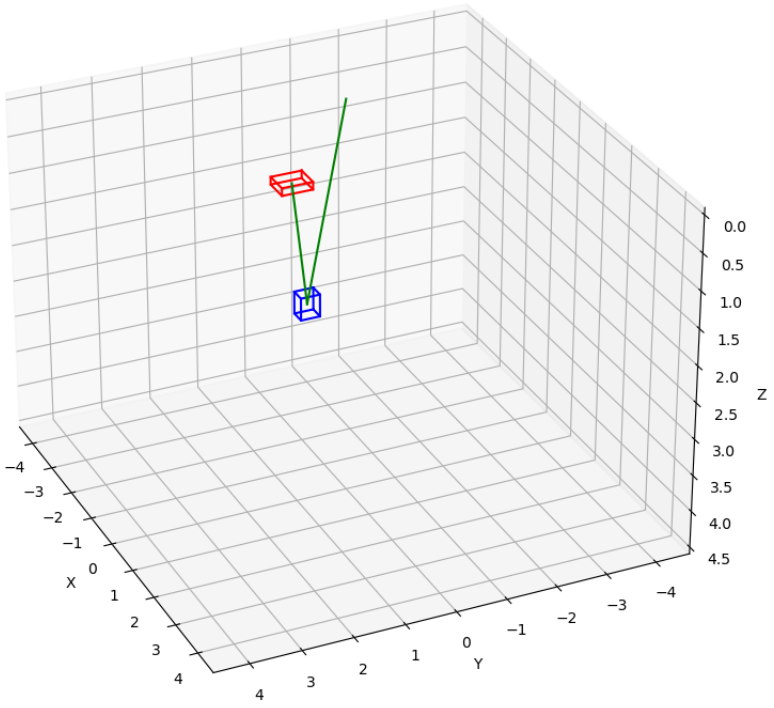


Figure 3.8: Simulateur affichant les boîtes des positions du ROV et de la masse glissante

3.7 Produit

Sur la Figure 3.9 on peut voir le produit final. La raspberry PI et la carte Fanthom X sont fixées sur une plaque grâce à une pièce imprimée en 3D. La batterie vient s'installer de l'autre côté de la plaque. Cette plaque s'imbrique facilement dans un support à l'intérieur de la capsule, ce qui permet d'accéder sans problème à tous les éléments de la capsule. Les câbles passent par une tap étanche de chez BlueRobotics et sur laquelle est également placé le capteur de pression. La coque se visse relativement rapidement autour du câble.

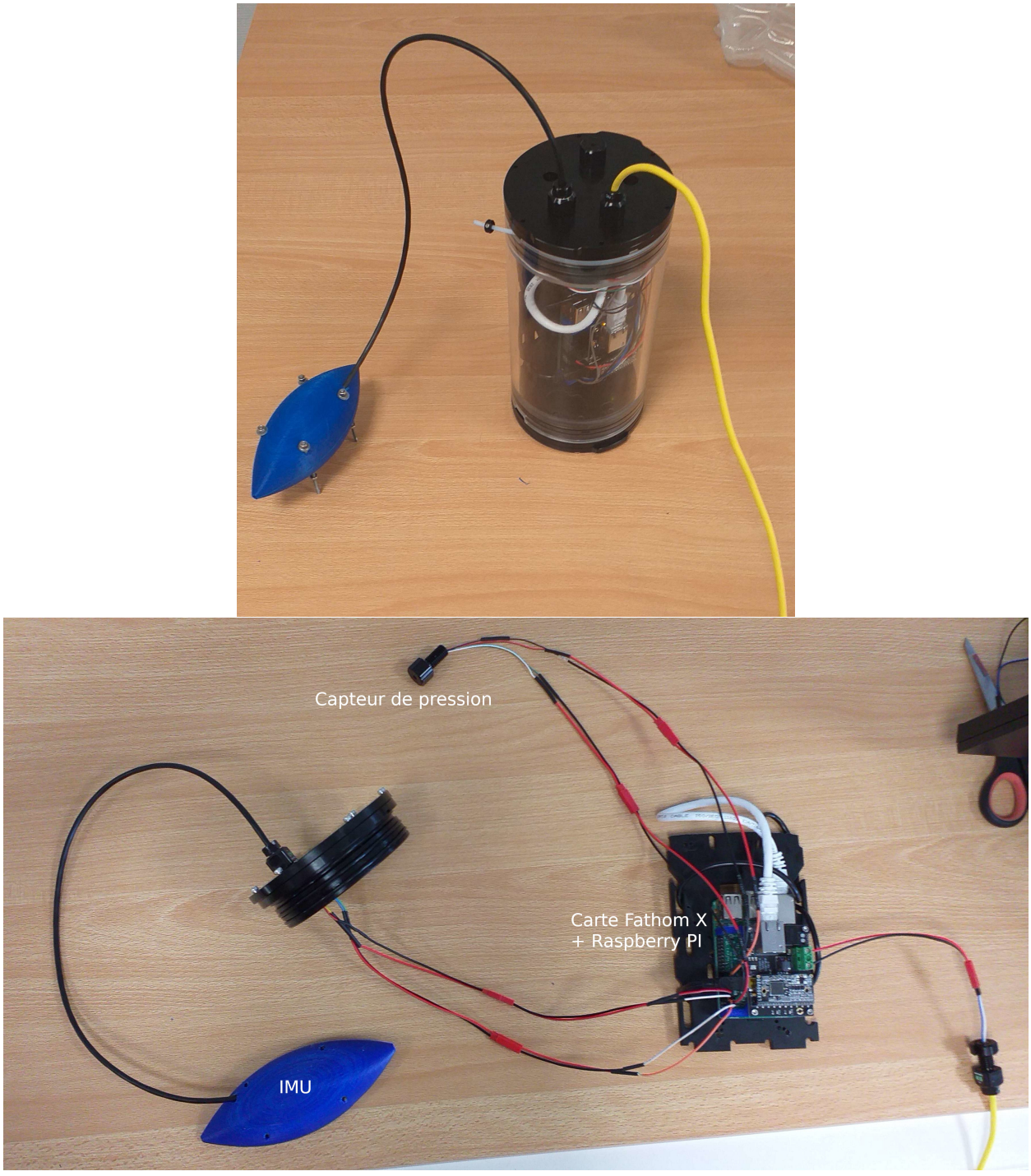


Figure 3.9: Produit final assemblé et désassemblé

Chapter 4

Expérimentation

L'expérimentation s'est faite dans deux environnements différents : dans l'air et dans un bassin d'eau.

Nous allons décrire les résultats obtenus dans chaque cas.

4.1 Dans l'air

4.1.1 Description

Les tests ont d'abord été faits dans l'air pour plusieurs raisons : permettre d'effectuer des premiers tests avant même d'avoir assuré l'étanchéité du système ; avoir plus de place pour permettre d'évaluer les résultats pour des longueurs de câble importantes ; et avoir facilement à disposition une vérité terrain.

Cependant, pour se passer du capteur de pression qui ne donne des valeurs cohérentes que dans l'eau, il fallait néanmoins fournir une profondeur au programme pour obtenir des résultats. L'idée était donc de déplacer le système à une distance verticale (entre le point d'attache et le ROV et qu'on appelle par la suite profondeur) constante et mesurée au préalable. Ainsi, il suffit de publier sur le topic adéquat la profondeur choisie.

Pour une même profondeur on déplace le système à des positions connues (voir Figure 4.1) que l'on compare au résultat du calcul.

4.1.2 Résultat

La Figure 4.2 présente les résultats d'une expérience avec 5m de câble. Durant cette expérience le système a été déplacé selon un seul axe et à 3 positions différentes : -2m, -3m et -4m (représentées par les traits noir horizontaux sur le graphe). Sur la figure, les courbes bleue et rouge correspondent aux deux valeurs extrêmes de la boîte résultante, et la courbe verte correspond au même calcul mais avec des boîtes d'entrée de taille nulle.

L'intérêt de cette dernière courbe est de se rendre compte de la valeur que l'on obtiendrait sans analyse par intervalles mais également la valeur vers laquelle on convergerait en faisant

Figure 4.1: Setup expérimental dans l'air

tendre les tailles des boîtes d'entrée vers 0. On peut remarquer dans un premier temps que cette courbe ne correspond pas au milieu des boîtes. Par ailleurs, les valeurs vraies sont systématiquement à l'intérieur des boîtes correspondantes, ce qui était l'objectif. La différence entre les deux graphes réside dans le choix du diamètre (la taille) des boîtes correspondant aux IMUs. Il s'agit d'un diamètre de 0,106 rad en haut et de 0,080 rad en bas. Il en résulte un diamètre pour la boîte de x d'environ 70 cm en haut contre environ 60 cm en bas. La valeur vraie de -4m est tout juste comprise dans l'intervalle résultant sur le graphe du bas ce qui suggère qu'on ne peut pas choisir un diamètre pour les boîtes d'entrée inférieur. Cependant, on remarque qu'il y a une erreur statique de 10 cm environ sur l'expérience. Celle-ci peut s'expliquer par plusieurs facteurs :

- La largeur de la poulie de la masse glissante est négligée dans la modélisation mathématique
- L'environnement était peut-être perturbé magnétiquement, faussant légèrement les mesures des IMUs

Si on arrivait à corriger cette erreur statique, on pourrait certainement réduire le diamètre en entrée pour affiner d'autant plus le résultat.

Le graphe de la Figure 4.3 rend compte de la distance moyenne entre l'erreur moyenne de la courbe verte et la valeur ciblée pour chaque expérience réalisée (au total 19, dont les résultats sont présentés en 4.1) en fonction de la longueur de câble L utilisée. Il s'agit ici de regarder l'évolution de l'erreur relative pour des valeurs de L de plus en plus grandes. En effet, si cette erreur est trop importante, l'intérêt du système pour des applications concrètes est réduit. Ici, la relation semble être affine entre les deux grandeurs, bien qu'il soit dur d'avoir la moindre conclusion étant donné que l'intervalle des valeurs de L n'est pas suffisamment grand et qu'il semble y avoir une valeur aberrante pour L = 5.

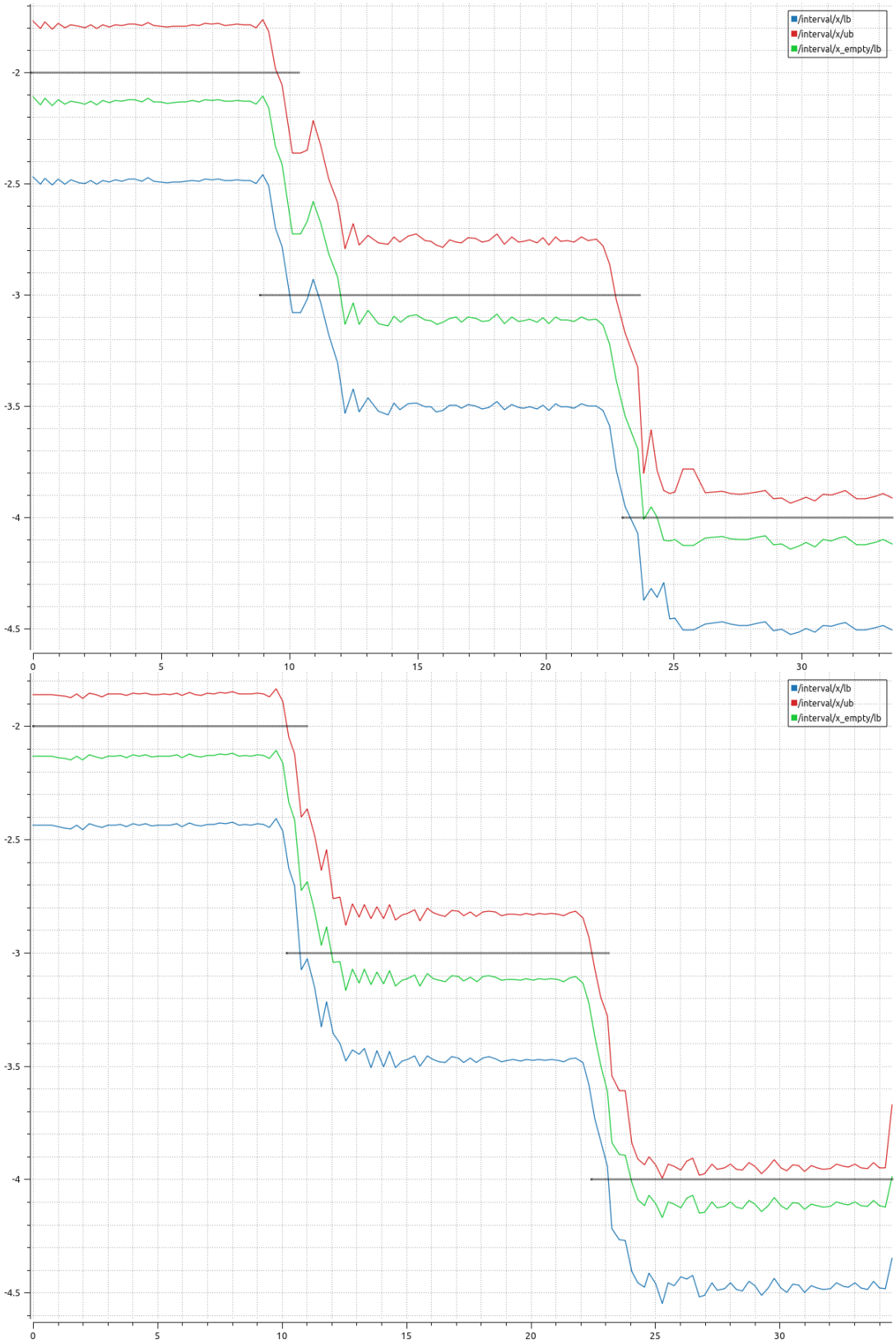


Figure 4.2: Graphe de x (en m) en fonction du temps pour une expérience avec une longeur de câble de 5 m et à une profondeur de 156 cm pour un diamètre d'entrée de de 0,106 rad en haut et de 0,080 rad en bas. Les traits noirs horizontaux sont la vérité terrain.

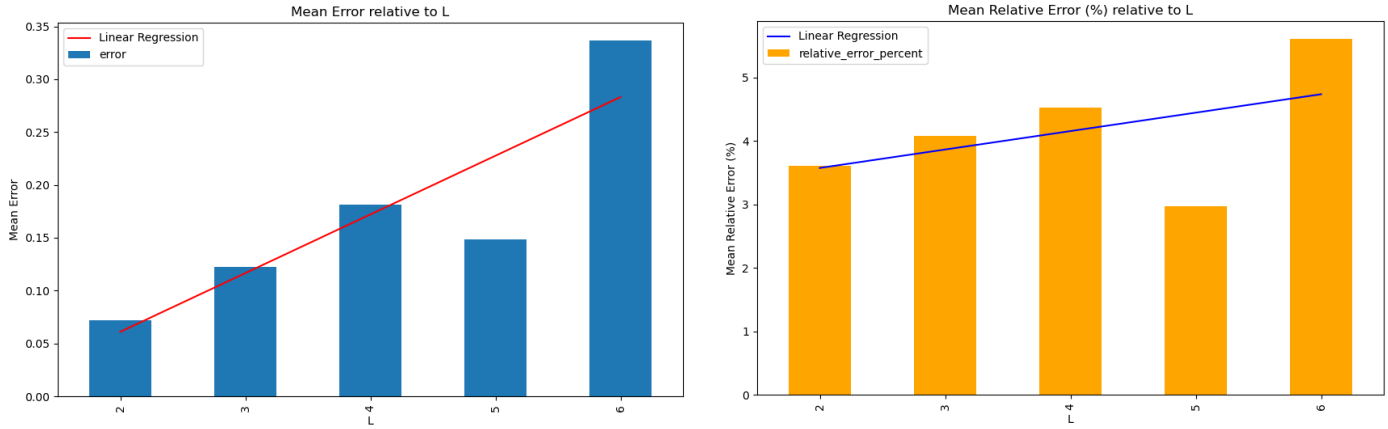


Figure 4.3: Erreur (en m) en fonction de la longueur L à gauche et erreur relative à droite. Les régressions linéaires donnent des coefficients directeurs de 0,06 à gauche et 0,27% à droite.

Test	L en m	z en m	Points testés [x, y]	Erreur moyenne en m
1	2.0	0.2	[0.5, 0], [1.0, 0], [1.5, 0]	0.10
2	2.0	0.4	[0.5, 0], [1.0, 0], [1.5, 0]	0.05
3	2.0	0.6	[0.5, 0], [1.0, 0], [1.5, 0]	0.07
4	2.0	0.8	[0.5, 0], [1.0, 0], [1.5, 0]	0.07
5	3.0	0.2	[0.5, 0], [1.0, 0], [1.5, 0]	0.14
6	3.0	0.6	[0.5, 0], [1.0, 0], [1.5, 0]	0.12
7	3.0	0.8	[1.0, 0], [1.5, 0]	0.10
8	4.0	0.2	[0, -2], [0, -3]	0.19
9	4.0	0.5	[0, -2], [0, -3]	0.13
10	4.0	1.0	[0, -3]	0.27
11	5.0	1.06	[-2, 0], [-3, 0], [-4, 0]	0.16
12	5.0	1.56	[-2, 0], [-3, 0], [-4, 0]	0.15
13	5.0	2.56	[-1, 0], [-2, 0], [-3, 0], [-4, 0]	0.14
14	6.0	1.06	[-2, 0], [-3, 0], [-4, 0]	0.22
15	6.0	1.56	[-2, 0], [-3, 0], [-4, 0]	0.24
16	6.0	2.56	[-1, 0], [-4, 0]	0.35
17	6.0	1.06	[-4, 0], [-4, -1], [-4, -2], [-4, -2.5]	0.26
18	6.0	1.56	[-4, 0], [-4, -1], [-4, -2], [-4, -2.5]	0.38
19	6.0	2.56	[-4, 0], [-4, -1], [-4, -2], [-4, -2.5]	0.55

Table 4.1: Erreur moyenne pour chaque test

4.2 Bassin

4.2.1 Caractéristiques problématiques

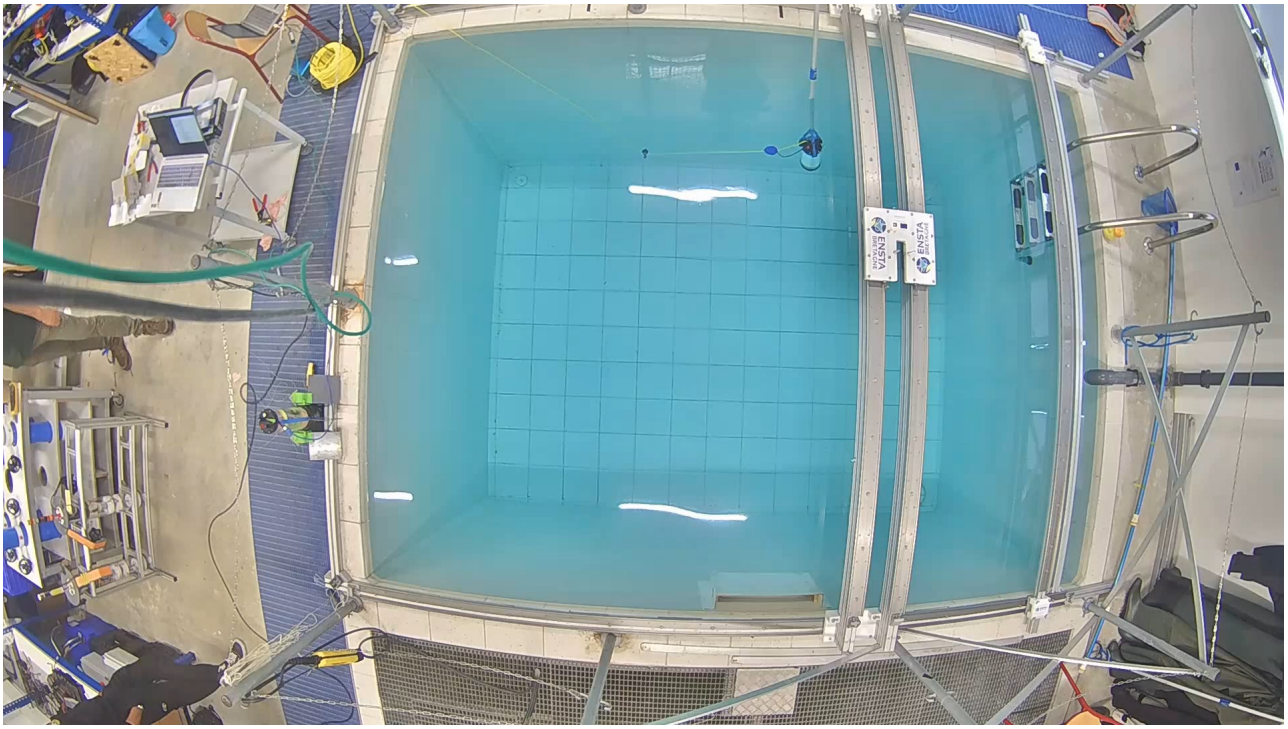


Figure 4.4: Setup expérimental en bassin (cliquer pour vidéo youtube)

Le bassin (c.f. Figure 4.4) qui était à ma disposition pour réaliser les essais était malheureusement soumis à des perturbations magnétiques. En particulier, lorsqu'on approche les IMUs des murs du bassins, le cap mesuré ne correspond plus au cap réel. Les IMUs étant l'élément le plus important du système, le bassin ne s'est pas révélé être un environnement adapté, bien qu'au centre du bassin on obtienne des résultats malgré tout corrects.

De plus, la taille de la piscine limite grandement la longueur de câble que l'on peut utiliser pour les expériences, ce qui est gênant pour étudier le comportement du système pour des longueurs L élevées.

Et par dessus tout, il n'y a aucune vérité terrain à disposition pour vérifier les résultats.

Comme illustré sur la Figure 4.4, le système a malgré tout été testé dans le bassin pour s'assurer de son bon fonctionnement (c.f. vidéo youtube).

4.2.2 Conclusion

Bien que le système fonctionne comme attendu, il est compliqué d'en tirer des conclusions, compte tenu des différentes caractéristiques discutées en partie 4.2.1. Cela n'est pas très gênant dans la mesure où le système ne nécessite pas d'être dans un milieu aquatique pour fonctionner,

sauf pour un élément qui est le capteur de pression. L'expérimentation en bassin a au moins permis de montrer que le capteur de pression fonctionne parfaitement et donne une très bonne précision. Les résultats de l'expérimentation dans l'air sont donc les seuls résultats qui ont été analysés durant ce stage.

Chapter 5

Conclusion

Comparaison des résultats

Il s'agit dans cette section de comparer les résultats obtenus durant ce stage avec ceux des systèmes que l'on trouve sur le marché.

Prenons l'exemple du système d'USBL décrit en introduction. D'après le site de l'entreprise iXblue, la précision du système est proportionnelle à la distance entre l'emetteur et le ROV avec un coefficient directeur typique de 1%. Dans notre cas, la régression linéaire donne un coefficient de l'ordre de 4,2% en moyenne pour les quelques longueurs de L testées. En considérant le fait que notre système est seulement un prototype, on peut se satisfaire de cet ordre de grandeur, cependant, pour des longueurs de L plus importantes cet ordre de grandeur est susceptible d'augmenter là où celui de l'USBL est constant.

Par ailleurs, une des limites de notre système est sa robustesse. En effet, il suffit que le câble se déforme légèrement en percutant un obstacle ou bien que la masse glissante vienne toucher le sol pour que le modèle mathématique, expliqué en section 2.1, ne corresponde plus à la réalité. Afin d'éviter cela, il est possible de remplacer et/ou d'ajouter à la masse glissante, un flotteur glissant, en revoyant légèrement les équations écrites. De cette manière, il est possible de s'adapter à l'environnement et éviter le problème décrit, à condition de connaître cet environnement un minimum.

Conclusion

L'objectif de réalisation du système de localisation est rempli. La capsule est prête à être fixée sur un ROV pour lui donner sa position en temps réel avec un affichage sur une simulation 3D en parallèle.

Concernant la précision de la méthode utilisée, les résultats sont mitigés. D'un côté, on arrive à bien cibler la valeur réelle dans toutes les expériences réalisées (4,2% d'erreur relative). Cependant, la taille des boîtes, bien qu'affinable, laisse à désirer (environ 70 cm de diamètre dans l'exemple présenté) et l'erreur relative augmente clairement avec la longueur du câble L.

On peut donc imaginer des missions où ce système pourrait être utile, compte tenu des avantages mentionnés en introduction, à condition de garder une longueur de câble raisonnable et où la marge d'erreur du ROV est de l'ordre du mètre.

Cependant, et comme décrit en section 5, notre système de localisation, dans son état actuel, n'est pas aussi précis que l'USBL qui est le système le plus couramment utilisé à ce jour.

Perspective

Dans un premier temps, il serait utile de tester des longueurs de câbles plus importantes pour voir la tendance de la courbe sur la Figure 4.3.

Il faudrait également varier les environnements de test en en choisissant des plus larges notamment.

En ce qui concerne le produit, la coque remplit parfaitement son rôle mais peut sûrement être améliorée pour que son installation soit plus rapide et plus pratique qu'actuellement.

Bibliography

[1] Pau Herrero, Pantelis Georgiou, Christofer Toumazou, BenoÃ®t Delaunay, and Luc Jaulin. An efficient implementation of sivia algorithm in a high-level numerical programming language. *Reliable Computing*, 16, 10 2012.

[2] Christophe Viel, Juliette Drupt, Claire Dune, and Vincent Hugel. Rov localization based on umbilical angle measurement. *Ocean Engineering*, 269:113570, 2023.