UE 3.2 - DDBOAT (Guerledan)
Promotion FISE 2027

ENSTA Campus de Brest
2 Rue François Verny
29200 Brest Cedex 9, France

Pedro ALVES DE ARAUJO SILVA
pedro.alves@ensta.fr
Luiz Felipe BARROS ALVES
luiz.barros@ensta.fr
Luíza DIESEL CHESANI
luiza.diesel_chesani@ensta.fr
Letícia Lie FURUTA
leticia.furuta@ensta.fr

Sommaire

Lien vers le dépôt : <https://gitlab.ensta-bretagne.fr/furutale/guerledan-2025>

1 Calibration IMU et déplacement vers l'ouest

L'objectif de la première journée à Guerlédan était de calibrer l'orientation du DDBOAT en utilisant des coordonnées. Cela a été réalisé en calibrant le bateau avec le champ magnétique terrestre au lac. Afin de vérifier le fonctionnement du code, nous avons réalisé un test : faire avancer le bateau vers l'ouest pendant 15 secondes.

1.1 Méthode de Moindres Carrés

Tout d'abord, un petit programme (implémenté dans `mycalibration.py`) crée un fichier texte (.txt) contenant les données lues par le magnétomètre du bateau. Grâce à ce fichier, il est possible de créer une matrice **M** composée des coordonnées ***x, y*** et ***z*** issues du magnétomètre. D'après notre méthode théorique (implémenté dans `MoindresCarrees.py`) :

Selon la Méthode de Moindres Carrées, la formule suivante est utilisée :

$$\vec{p} = (M^T.\ M)^{-1}.\ M^T.\ \begin{pmatrix} 1 \\ ... \\ 1 \end{pmatrix} \tag{1}$$

où $\boldsymbol{p}$ est le vecteur des paramètres ajustés. À partir de $\boldsymbol{p}$, nous pouvons obtenir la matrice **Q**, donnée par l'équation

$$Q = \begin{bmatrix} p_1 & \frac{1}{2}\,p_4 & \frac{1}{2}\,p_5 \\ \frac{1}{2}\,p_4 & p_2 & \frac{1}{2}\,p_6 \\ \frac{1}{2}\,p_5 & \frac{1}{2}\,p_6 & p_3 \end{bmatrix} \tag{2}$$

et le vecteur

$$\vec{b} = -\,\frac{1}{2}\,Q^{-1} \begin{pmatrix} p_7 \\ p_8 \\ p_9 \end{pmatrix} \tag{3}$$

Ces équations permettent de calculer la calibration $\boldsymbol{y}$ en utilisant la formule de linéarisation

$$y = \sqrt{Q}\ (x\ -\ b) \tag{4}$$

Et pour notre bateau, les valeures de la racine de Q et le vecteur b sont :

$$Q = \begin{bmatrix} 3.2447131810216e-04 & 9.45694949440714e-06 & -1.32311284804898e-06 \\ 9.45694949440714e-06 & 3.34176117235687e-04 & -8.13346622266005e-07 \\ -1.32311284804898e-06 & -8.13346622266019e-07 & 3.28722522151945e-04 \end{bmatrix} \tag{5}$$

et le vecteur

$$\vec{b} = \begin{pmatrix} -1.06212764559268e+03 \\ -4.60925804030264e+02 \\ -5.57279770380571e+02 \end{pmatrix} \tag{6}$$

Ces matrices sont appliquées en temps réel à chaque nouvelle mesure pour corriger la mesure. Enfin, nous avons simulé une sphère 3D afin de visualiser le résultat de la calibration et de vérifier que l'étalonnage a été effectué correctement.

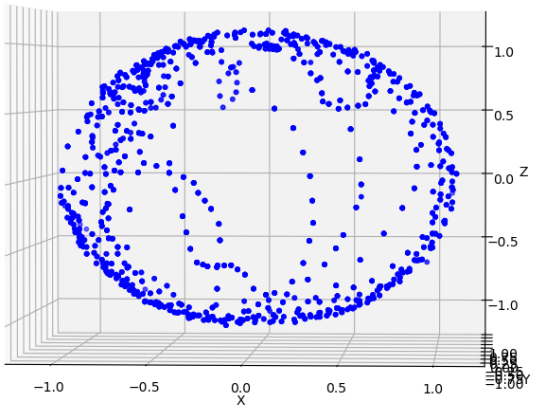


FIGURE 1 – Le sphère corrigée.

1.2 Parcours vers l'ouest

Pour commencer le premier test, nous avons développé un code qui se connecte à l'Arduino du bateau. Ce code a été légèrement modifié pour obtenir une série de données à chaque instant pendant une période déterminée.

Ces valeurs ont été utilisées pour obtenir notre matrice de rotation : la transposée de la matrice est alors formée par notre vecteur de coordonnées, ***x***, ***y*** et ***z***. Notre vecteur d'accélération, que nous définissons comme [0, 0, 1] pour simplifier le lac a un plan. La formule suivante était utilisée :

$$\mathbf{R} = \left(\frac{\mathbf{y}_1 - (\mathbf{y}_1^{\mathrm{T}}\mathbf{a}_1)\,\mathbf{a}_1}{\|\mathbf{y}_1 - (\mathbf{y}_1^{\mathrm{T}}\mathbf{a}_1)\,\mathbf{a}_1\|} \;\middle|\; \mathbf{a}_1 \wedge \frac{\mathbf{y}_1 - (\mathbf{y}_1^{\mathrm{T}}\mathbf{a}_1)\,\mathbf{a}_1}{\|\mathbf{y}_1 - (\mathbf{y}_1^{\mathrm{T}}\mathbf{a}_1)\,\mathbf{a}_1\|} \;\middle|\; \mathbf{a}_1 \right)^{\mathrm{T}}$$

Ensuite, avec cette matrice de rotation il est possible de déterminer les angles d'Euler (ϕ, θ, ψ) - roulis, tangage et lacet - par les formules :

$$\phi = \arctan 2(R_{32},\, R_{33}) \cdot \frac{180}{\pi},$$

$$\theta = -\arcsin(R_{31}) \cdot \frac{180}{\pi},$$

$$\psi = \arctan 2(R_{21},\, R_{11}) \cdot \frac{180}{\pi}.$$

En utilisant ces angles, on peut réaliser la première navigation sur le lac (implémentée dans `parcours_vers_ouest.py`). Comme expliqué précédemment, le bateau doit avancer vers l'ouest pendant 15 secondes. Ainsi, on utilise les angles d'Euler pour

calculer l'erreur en degrés. En fonction de cette erreur de direction, la vitesse des deux moteurs est ajustée : l'un est légèrement accéléré tandis que l'autre est ralenti, ce qui permet de corriger la trajectoire et de ramener le bateau vers la bonne orientation.

2 Parcours vers la bouée

2.1 Définition des matrices Z, Y et R_{imu}

Pour déterminer la matrice de correction R_{imu}, nous devons d'abord expliciter les matrices Z (valeurs théoriques) et Y (valeurs mesurées).

La matrice cible Z est construite à partir des vecteurs de champ magnétique attendus dans le repère local, en tenant compte d'une inclinaison magnétique $I \approx 64°$. Chaque colonne de Z correspond au vecteur théorique pour une orientation spécifique du bateau lors de l'étalonnage (respectivement : aligné au Nord, à l'Ouest, et à la Verticale).

D'après notre modèle théorique (implémenté dans `R_imu.py`) avec $C = \cos(64°)$ et $S = \sin(64°)$:

$$Z = \begin{pmatrix} C & 0 & -S \\ 0 & -C & 0 \\ -S & -S & C \end{pmatrix} \approx \begin{pmatrix} 0.438 & 0 & -0.899 \\ 0 & -0.438 & 0 \\ -0.899 & -0.899 & 0.438 \end{pmatrix}$$

La matrice Y est constituée des moyennes des mesures brutes (après calibration) obtenues expérimentalement pour ces trois mêmes positions. Ces valeurs reflètent la "vision" distordue du capteur en raison de son désalignement physique sur le bateau.

Les données expérimentales relevées sont les suivantes :

$$Y = \begin{pmatrix} -0.494 & -0.099 & 0.926 \\ -0.070 & -0.551 & 0.001 \\ 0.970 & 0.917 & -0.462 \end{pmatrix}$$

*(Note : Chaque colonne correspond au vecteur moyen mesuré dans l'une des trois positions d'étalonnage. Cette étape a été implémentée avec le code **captureDonne.py**)*

En appliquant la relation $R_{imu} = Z \cdot Y^{-1}$, nous obtenons la matrice de rotation qui compense le défaut de montage du capteur. Cette matrice assure que, lorsque le bateau est physiquement aligné vers le Nord, le vecteur magnétique corrigé sera mathématiquement égal au vecteur théorique du Nord.

Le résultat final, intégré dans notre système de navigation (**Vers_la_bouee.py**), est :

$$R_{imu} \approx \begin{pmatrix} -0.904 & 0.235 & 0.170 \\ 0.092 & 0.887 & -0.007 \\ -0.094 & 0.094 & -0.911 \end{pmatrix}$$

Cette matrice a été calculée dans **R_imu.py** est appliquée en temps réel à chaque nouvelle mesure pour "redresser" virtuellement le capteur avant le calcul du cap.

2.2 Calcul

Après avoir obtenu la matrice R_{imu}, la logique de navigation a été implémentée dans le code **Ver_la_bouee.py** pour que le DDBOAT suive le parcours correctement. Pour cela, le code peut être divisé en trois étapes principales.

2.2.1 Données via GPS

En utilisant les drivers déjà existants sur l'équipement, nous pouvons récupérer les données du GPS embarqué. Ensuite, nous calculons la distance (en x et y) du bateau jusqu'à l'objectif avec la fonction `Calc_dist`.

```
def Calc_dist(lon, lat, ref_lon, ref_lat):
    x = RAIO*np.cos(lat*np.pi/180)*(lon-ref_lon)*np.pi/180
    y = RAIO*(lat-ref_lat)*np.pi/180
    return x, y
```

Listing 1 – Calcul de la distance entre le bateau et la bouée.

À partir de cette distance, nous calculons l'angle désiré (la direction que le bateau *doit suivre* pour atteindre la bouée). Cet angle, converti en coordonnées géographiques, est stocké dans la variable `heading_geo`. Grâce à cet angle, le DDBOAT sait dans quelle direction il doit se déplacer.

2.2.2 Calcul de l'angle ψ

Cette étape calcule le cap actuel du bateau. Avec les données brutes de l'IMU, nous appliquons d'abord la matrice R_{imu} pour corriger la mesure magnétique (dans le code : `Y_cor = R_imu @ Y_mag_nor`).

La fonction `M_R` utilise ensuite ce vecteur corrigé pour calculer l'angle ψ. Cet angle ψ représente le **cap actuel** du bateau (la direction de son avant).

```
def M_R(Y_mag_nor, R_imu):
    a = np.array([[0],[0],[1]])
    Y_cor = R_imu @ Y_mag_nor
    _aux = Y_cor - (Y_cor.T@a)*a
    _mod = np.sqrt(_aux[0]**2 + _aux[1]**2 + _aux[2]**2)
    _col1 = _aux / _mod
    _col2 = np.cross(a.flatten(), _col1.flatten()).reshape(3,1)
    _col3 = a
    R_cor = np.c_[_col1, _col2, _col3]
    R_cor = R_cor.T

    _phi = np.arctan2(R_cor[2,1], R_cor[2,2]) * 180/np.pi
    _theta = np.arcsin(-R_cor[2,0]) * 180/np.pi
    _psi = np.arctan2(R_cor[1,0], R_cor[0,0]) * 180/np.pi

    return _psi

xmag,ymag,zmag = imu.read_mag_raw()
data_mag = np.array([[xmag], [ymag], [zmag]])
centered_data = data_mag - b_M
Y_mag = (Q_sqrt @ centered_data)
```

```
Y_mag_nor = Y_mag / np.sqrt(Y_mag[0]**2 + Y_mag[1]**2 + Y_mag
    [2]**2)

psi = (M_R(Y_mag_nor, R_imu))
```

Listing 2 – Calcul de l'angle ψ.

2.2.3 Système de contrôle des moteurs

En utilisant les données obtenues dans les étapes précédentes, le système de contrôle des moteurs a été implémenté.

Pour diriger le bateau, le système compare l'angle désiré (du GPS) avec le cap actuel (de l'IMU). Nous calculons l'erreur de direction : c'est la différence entre la direction de la bouée (`heading_geo`) et la direction actuelle du bateau (`psi`).

Un contrôleur proportionnel (Kp) simple est utilisé. Il multiplie cette erreur pour calculer un "ajustement" (`turn_adjustment`). Cet ajustement est ensuite ajouté à la vitesse de base d'un moteur et soustrait de la vitesse de l'autre moteur. Cela fait tourner le bateau dans la bonne direction.

Enfin, la fonction `clamp` assure que les commandes envoyées aux moteurs restent dans des limites de sécurité (entre 100 et 255).

```
def clamp(value, min_val, max_val):
    """Clamps a value between a min and max."""
    return max(min_val, min(value, max_val))

error = sawtooth(heading_geo - psi) + CORRECTION
turn_adjustment = int(Kp * error)

motor_right = BASE_SPEED + turn_adjustment
motor_left = BASE_SPEED - turn_adjustment
motor_left_cmd = clamp(motor_left, 100, 255)
motor_right_cmd = clamp(motor_right, 100, 255)
DDBoat7.send_arduino_cmd_motor(motor_left_cmd, motor_right_cmd)
```

Listing 3 – Système de contrôle.

Nous pouvons voir le système de contrôle du bateau comme le montre la figure ci-dessous :

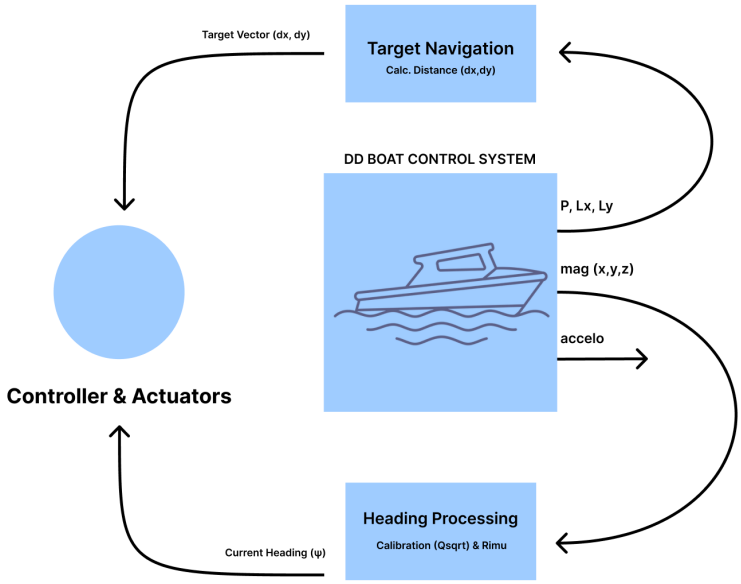


FIGURE 2 – Système DDBOAT.

2.3 Résultat

Pour valider l'expérience, le code **Ver_la_bouee.py** sauvegarde les données GPS capturées à chaque instant. Ces données sont enregistrées dans un fichier `.kml`, ce qui permet de visualiser le parcours exact réalisé par le bateau.

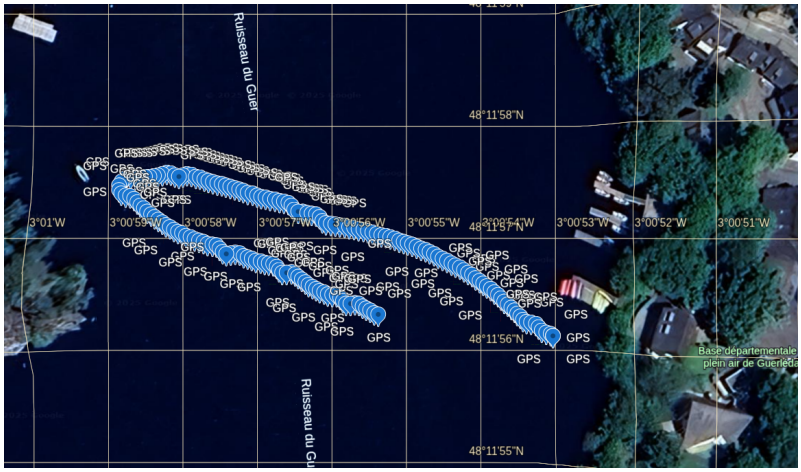


FIGURE 3 – Parcours DDBOAT.

La figure 3 montre le parcours complet du DDBOAT. On peut clairement identifier les deux phases de la mission, comme défini dans le code :

— **Phase 1 (Aller) :** Le bateau part du point de départ et navigue vers la bouée (la première cible GPS).

— **Phase 2 (Retour) :** Une fois la cible atteinte (avec une tolérance de 15 mètres), le bateau reçoit un nouvel objectif (le point de départ) et commence

son parcours de retour.

On note que le bateau suit bien le chemin désiré. La trajectoire montre comment le système de contrôle (décrit avant) corrige la direction du bateau pour atteindre la cible. Cette figure montre donc le succès de la navigation autonome.

Une autre analyse intéressante à présenter est le graphique de la distance entre le bateau et la cible (Figure 4), avec des valeurs obtenues par la fonction présentée dans la section 2.2.1. Au début, l'erreur est grande. Quand le bateau avance, le système de contrôle corrige sa trajectoire et l'erreur diminue progressivement jusqu'à atteindre la tolérance définie dans le code. Quand cette tolérance est atteinte, c'est-à-dire quand le bateau est assez proche de la bouée, sa cible change. Ainsi, l'erreur augmente de nouveau et le processus se répète jusqu'à ce que le DDBOAT arrive à sa nouvelle cible.

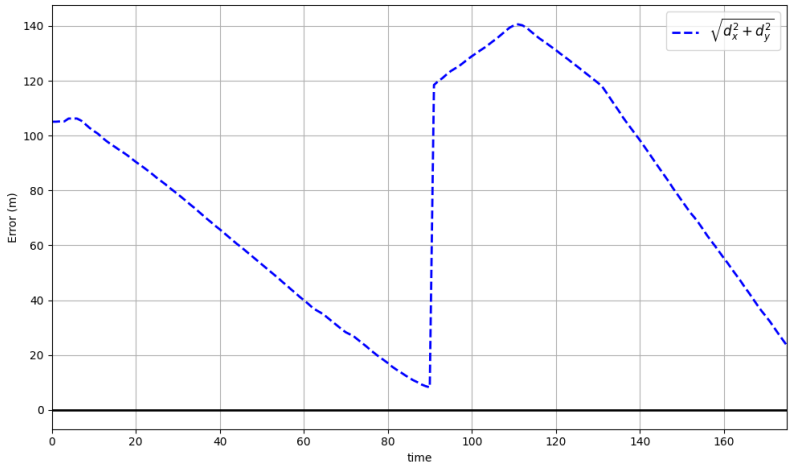


FIGURE 4 – L'erreur entre le bateau et la cible.

3 Dernier défi

Après avoir relevé le défi consistant à aller jusqu'à la bouée puis à revenir, nous avons reçu une dernière mission. Cette mission nécessitait que toutes les équipes travaillent ensemble en harmonie.

Ainsi, la mission consiste à faire naviguer un bateau sur un grand arc de circonférence de rayon R_1. Autour de lui, les autres bateaux devront « danser », c'est-à-dire évoluer en formation autour de sa trajectoire, avec un rayon R_2.

L'évolution de la position du bateau central, avait un rayon R_1 autour de un point C, avec une vitesse w_1, un temps de démarrage, t_o, comme quand on va commencer cette danse, et un temps de offset t_{ff}, cela est donnée par la formule :

$$p(t) = \begin{pmatrix} x_c + R_1 cos(w_1(t - t_o + t_{ff})) \\ y_c + R_1 sin(w_1(t - t_o + t_{ff})) \end{pmatrix} \tag{7}$$

Et pour les autres bateau qui vont tourner autour de cette bateau centrale ils ont une autre formule pour la position. avec un nouvelle rayon R_2, un nouvelle vitesse w_2 et une différence de phase entre eux, de sorte que chaque bateau se trouve dans

une position différente sur le cercle.

$$p(t,i) = \begin{pmatrix} p_1(t) + R_2 cos(w_2(t - t_o + t_{ff}) + \frac{2\pi}{N}\ i) \\ p_2(t) + R_2 sin(w_2(t - t_o + t_{ff}) + \frac{2\pi}{N}\ i) \end{pmatrix} \quad (8)$$

Et pour les valeur de chaque variable constant, nous avons :
- $R_1 = 240$ m
- C = $\begin{pmatrix} x_c = 48,199706 \\ y_c = -3,018484 \end{pmatrix}$
- $\omega_1 = \frac{1}{2R_1}$
- $t_o = 9h30$
- $t_{ff} = 100sec$
- $R_2(t) = 10\exp(\frac{t-t_o}{800}) + 5$
- $\omega_2 = \frac{1}{2R_2}$

4 Problèmes rencontrés

Le deuxième jour sur le lac, nous avons rencontré des problèmes liés au matériel de notre bateau. De ce fait, il a été nécessaire de changer de bateau plusieurs fois.

La première difficulté rencontrée sur le bateau numéro 8 concernait la lecture des données fournies par l'IMU : elles variaient de manière imprécise. En cherchant des problèmes physiques sur notre robot, nous avons découvert que l'IMU n'était pas fixée sur le bateau.

Après une intervention, un deuxième problème est survenu sur le bateau numéro 6, très similaire au précédent : l'IMU s'est également desserrée. Nous avons donc changé pour utiliser le bateau numéro 1, qui a bien fonctionné avec l'IMU, mais il captait des nombres imaginaires qui ont causé un problème avec le calcul de $\sqrt{Q}$. Après l'aide des vétérans, nous avons encore changé de bateau pour utiliser le bateau numéro 6 dont l'IMU avait été fixée. Ainsi, ces problèmes ont ralenti considérablement le développement et la réalisation des travaux que l'équipe devait faire.

5 Conclusion

Au cours de ce projet DDBOAT à Guerlédan, nous avons pu mettre en pratique plusieurs notions vues en cours, telles que la calibration de l'IMU, la modélisation du mouvement et le développement d'un système de navigation autonome. Malgré quelques difficultés techniques, comme les changements de bateaux, nous avons réussi à comprendre et à appliquer les principes nécessaires au bon fonctionnement du système.

Malheureusement, nous n'avons pas pu réaliser le dernier défi, qui consistait à faire les bateaux "capturer" et tourner autour d'un bateau central. Même si cette partie n'a pas été finalisée, les bases théoriques et les premières étapes de la programmation ont été mises en place.

En conclusion, cette expérience a été très importante : elle nous a permis de combiner théorie et pratique, de travailler en équipe et de mieux comprendre les

défis réels de la robotique. Ces apprentissages seront très utiles pour nos prochains projets, notamment celui du cours d'Ateliers.

UE 3.2 - DDBOAT (Guerlédan)
Rapport

Pedro Henrique BARBOSA NORI
pedro.barbosa@ensta.fr
Matheus BENIGNO DA SILVA
matheus.benigno@ensta.fr
Daniel FÁRLEI SILVA SOUZA
daniel.farlei@ensta.fr

Sommaire

1 Introduction et objectifs

Le présent rapport décrit le développement et la réalisation d'une expérimentation menée avec un bateau miniature sur le lac de Guerlédan, par les étudiants de robotique de la Promotion FISE27 de l'ENSTA Bretagne. Ce projet, avait pour principal objectif de mettre en pratique, dans un environnement réel, les concepts théoriques étudiés en salle de classe, en particulier ceux liés aux systèmes inertielles. L'activité a offert une occasion concrète de comprendre le fonctionnement et l'importance des capteurs ainsi que des méthodes de navigation utilisées en robotique mobile.

De plus, le lac a fourni un cadre idéal pour simuler des conditions réelles d'opération, permettant aux étudiants de se confronter aux défis typiques de la robotique de terrain, tels que les variations environnementales, les bruits de mesure et les limitations de communication. Ainsi, le projet a non seulement renforcé l'apprentissage technique, mais a également favorisé l'intégration entre théorie et pratique, offrant une expérience enrichissante dans le domaine de l'ingénierie et de la robotique appliquée.

1.1 Composants de bateau

La propulsion du bateau est assurée par deux moteurs *brushless* commandés par modulation de largeur d'impulsion (PWM) à travers ses ESC (*i.e.*, *Electronic Speed Controller*), offrant un contrôle de la vitesse et de la direction. Comme capteurs principaux de navigation, l'embarcation est équipée d'une unité de mesure inertielle (IMU) et d'un module GPS, responsables respectivement de l'estimation de l'orientation et de la détermination de la position absolue dans l'environnement. Pour garantir une autonomie suffisante durant toute l'activité, le système est alimenté par deux batteries 2S de 5000 mAh.

2 Mesures du magnétomètre et Calibration

L'objectif de la première journée de Guerlédan était de calibrer le magnétomètre et d'obtenir les angles d'Euler à partir des données de l'IMU. Pour bien effectuer la calibration, il est nécessaire de tourner le bateau autour de tous les axes et de réaliser environ 1000 captures de données, qui ont la forme d'un ellipsoïde déplacé par rapport à l'origine du système. Ceci est à cause des effets de *hard iron* et de *soft iron*. Puis effectuer la sphérisation. Ainsi, à la fin de la calibration, il est possible d'effectuer la Mission 1.

2.1 Méthode de Moindres Carrés

Dans le cadre de la calibration du magnétomètre, les mesures brutes ont été utilisées pour ajuster un ellipsoïde au moyen de la méthode des moindres carrés. Le modèle mathématique de la quadrique est exprimé sous la forme algébrique suivante :

$$p_1x^2 + p_2y^2 + p_3z^2 + p_4xy + p_5xz + p_6yz + p_7x + p_8y + p_9z = 1 \tag{1}$$

Cette équation représente la forme générale d'un ellipsoïde, où chaque coefficient p_i correspond à un paramètre à estimer. L'ensemble de ces coefficients est regroupé dans le vecteur suivant :

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \end{bmatrix} \tag{2}$$

Afin d'appliquer la méthode des moindres carrés, une matrice $\mathbf{M}$ est construite à partir des mesures du magnétomètre (x_i, y_i, z_i). Chaque ligne de cette matrice correspond à une mesure et contient les termes nécessaires à la formation du système linéaire associé :

$$\mathbf{M} = \begin{bmatrix} x_1^2 & y_1^2 & z_1^2 & x_1y_1 & x_1z_1 & y_1z_1 & x_1 & y_1 & z_1 \\ x_2^2 & y_2^2 & z_2^2 & x_2y_2 & x_2z_2 & y_2z_2 & x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N^2 & y_N^2 & z_N^2 & x_Ny_N & x_Nz_N & y_Nz_N & x_N & y_N & z_N \end{bmatrix} \tag{3}$$

Le vecteur $\mathbf{b}$, composé uniquement de valeurs égales à 1, correspond au terme indépendant du modèle de l'ellipsoïde normalisé :

$$\mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{4}$$

Le problème d'ajustement peut ainsi être écrit sous la forme du système linéaire suivant :

$$\mathbf{Mp} = \mathbf{b} \tag{5}$$

La solution du système consiste à minimiser l'erreur quadratique entre le modèle et les mesures. Les paramètres de la quadrique sont alors déterminés par :

$$\mathbf{p} = (\mathbf{M}^\top \mathbf{M})^{-1} \mathbf{M}^\top \mathbf{b} \tag{6}$$

Cette procédure permet d'estimer les coefficients décrivant l'équation de l'ellipsoïde associé aux mesures brutes du magnétomètre.

À partir des données calculées, il a été possible de générer l'ellipsoïde, comme le montrent le figure suivante :

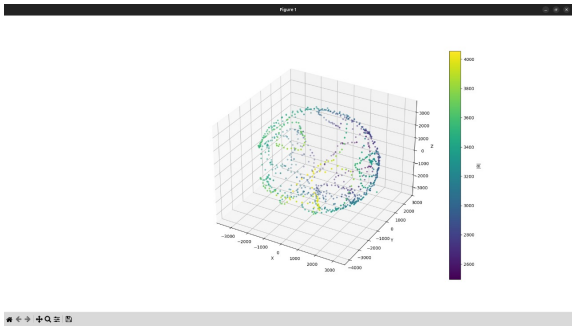


FIGURE 1 – Ellipsoïde

2.2 Sphérisation

Après l'estimation des paramètres de l'ellipsoïde par la méthode des moindres carrés, une étape supplémentaire consiste à déterminer la transformation permettant de la convertir en une sphère. Cette opération, appelée *sphérisation*, permet de corriger les effets de *soft iron* et d'obtenir un modèle isotrope du champ magnétique.

Pour cela, on construit d'abord la matrice symétrique $\mathbf{Q}$, définie à partir des coefficients quadratiques du modèle algébrique de l'ellipsoïde. Cette matrice regroupe les termes associés à x^2, y^2, z^2, xy, xz et yz. Elle est donnée par :

$$\mathbf{Q} = \begin{bmatrix} p_1 & \frac{p_4}{2} & \frac{p_5}{2} \\ \frac{p_4}{2} & p_2 & \frac{p_6}{2} \\ \frac{p_5}{2} & \frac{p_6}{2} & p_3 \end{bmatrix} \tag{7}$$

À partir de cette matrice, le centre de l'ellipsoïde est obtenu en appliquant la relation suivante, qui résulte de la mise sous forme canonique du modèle quadratique :

$$\mathbf{b} = -\frac{1}{2}\mathbf{Q}^{-1} \begin{bmatrix} p_7 \\ p_8 \\ p_9 \end{bmatrix} \tag{8}$$

Le vecteur $\mathbf{b}$ représente les coordonnées du centre de l'ellipsoïde dans le repère du magnétomètre. Une fois ce centre déterminé, il devient possible d'appliquer une transformation linéaire permettant de rendre l'ellipsoïde isotropique. Pour cela, on calcule la racine matricielle de $\mathbf{Q}$, notée $\sqrt{\mathbf{Q}}$, qui servira de base pour la normalisation et l'obtention d'une sphère de rayon constant.

Une fois déterminés le centre de l'ellipsoïde $\mathbf{b}$ et la racine matricielle $\sqrt{\mathbf{Q}}$, il est possible d'effectuer la transformation finale permettant de convertir les mesures du magnétomètre en points appartenant à une sphère. Cette transformation constitue l'étape essentielle de la *sphérisation*.

Ensuite, une mise à l'échelle anisotrope est appliquée à l'aide de la matrice $\sqrt{\mathbf{Q}}$. Cette opération corrige les déformations introduites par les effets de *soft iron*, transformant ainsi l'ellipsoïde obtenu en une sphère de rayon presque constant.

La transformation complète permettant d'obtenir les points sphérisés $\mathbf{Y}_i$ est donc donnée par :

$$\mathbf{Y}_i = \sqrt{\mathbf{Q}}\,(\mathbf{x}_i - \mathbf{b}) \tag{9}$$

où :
— $\mathbf{x}_i$ représente la i-ème mesure brute du magnétomètre ;
— $\mathbf{b}$ est le centre de l'ellipsoïde calculé précédemment ;
— $\sqrt{\mathbf{Q}}$ est la racine matricielle symétrique de la matrice $\mathbf{Q}$.

L'ensemble des vecteurs $\mathbf{Y}_i$ forme un nuage de points quasiment sphérique. Ce résultat confirme la bonne compensation des effets de *hard iron* et *soft iron*, permettant ainsi d'utiliser les données du magnétomètre pour l'estimation précise des orientations.

À partir des données calculées, il a été possible de générer la sphère, comme le montrent le figure suivante :

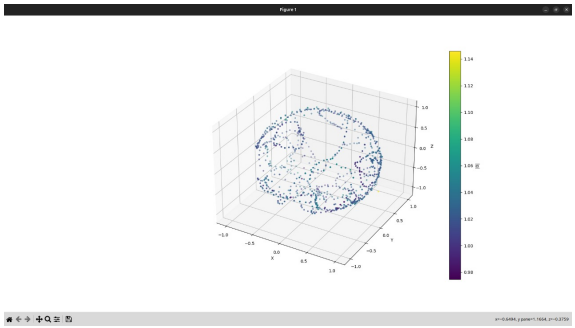


FIGURE 2 – Sphère

2.3 Estimation de l'orientation et angles d'Euler

On utilise la convention d'Euler ZYX (rotations successives : ψ autour de z, puis θ autour de y, puis φ autour de x). Ainsi la matrice de rotation qui transforme un vecteur exprimé dans le repère du robot en repère inertiel (*world*) est :

$$R \;=\; R_z(\psi)\, R_y(\theta)\, R_x(\varphi), \tag{10}$$

où

$$R_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix},$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{11}$$

En développant, on obtient la forme explicite de $R = [r_{ij}]$:

$$R = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\varphi - \sin\psi\cos\varphi & \cos\psi\sin\theta\cos\varphi + \sin\psi\sin\varphi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\varphi + \cos\psi\cos\varphi & \sin\psi\sin\theta\cos\varphi - \cos\psi\sin\varphi \\ -\sin\theta & \cos\theta\sin\varphi & \cos\theta\cos\varphi \end{bmatrix} \tag{12}$$

À partir des éléments r_{ij} de R, les angles se retrouvent par :

$$\theta = \operatorname{asin}(-r_{31})$$

$$\varphi = \operatorname{atan2}(r_{32},\, r_{33})$$
$$\psi = \operatorname{atan2}(r_{21},\, r_{11}) \tag{13}$$

La matrice R peut être obtenue à partir des vecteurs normalisés $\mathbf{y}_1$ du magnétomètre déjà calibré et $\mathbf{a}_1$ de l'accéléromètre.

$$\mathbf{R} = \left(\begin{array}{cc} \dfrac{\mathbf{y}_1 - (\mathbf{y}_1^{\mathrm{T}}\mathbf{a}_1)\,\mathbf{a}_1}{\|\mathbf{y}_1 - (\mathbf{y}_1^{\mathrm{T}}\mathbf{a}_1)\,\mathbf{a}_1\|} & \mathbf{a}_1 \wedge \dfrac{\mathbf{y}_1 - (\mathbf{y}_1^{\mathrm{T}}\mathbf{a}_1)\,\mathbf{a}_1}{\|\mathbf{y}_1 - (\mathbf{y}_1^{\mathrm{T}}\mathbf{a}_1)\,\mathbf{a}_1\|} \quad \mathbf{a}_1 \end{array} \right)^{\mathrm{T}} \tag{14}$$

Pour simplifier le calculs, le vecteur $\mathbf{a}_1$ a été fixé ainsi que :

$$\mathbf{a}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{15}$$

Par contre, on assume $\theta = \varphi = 0$ (pas de roulis ni de tangage), ce qui n'affecte pas l'application, puisque on s'intéresse qu'au cap ψ.

3 Mission 1 : Se déplacer vers l'ouest

.

3.1 Regulation et Données GPS

Au deuxième jour du projet, durant la matinée, un cours complémentaire a été consacré au concept de régulation de l'IMU, dans le but d'obtenir des valeurs angulaires plus cohérentes avec celles mesurées par des instruments de référence. Cette étape s'est révélée essentielle pour comprendre les imperfections inhérentes aux capteurs inertiels et la nécessité d'appliquer des méthodes mathématiques de calibration afin de corriger les écarts systématiques.

La méthode présentée consistait à déterminer une matrice de correction permettant d'ajuster les angles d'Euler fournis par l'IMU, de manière à les rapprocher des valeurs théoriques attendues. Pour cela, on considérait tout d'abord la matrice théorique, définie comme :

$$Z_{\text{théorique}} = \begin{pmatrix} \cos I & 0 & -\sin I \\ 0 & -\cos I & 0 \\ -\sin I & -\sin I & \cos I \end{pmatrix} \tag{16}$$

tandis que la matrice réelle mesurée par le système inertiel était représentée sous la forme :

$$Y = \left(\mathbf{Y_n} \mid \mathbf{Y_o} \mid \mathbf{Y_h}\right), \tag{17}$$

où chaque vecteur colonne correspond respectivement aux mesures selon les axes **Nord**, **Ouest** et **Haut**. Cette convention de repérage est essentielle pour garantir la cohérence entre le modèle théorique et les données issues du capteur.

À partir de ces deux matrices, il a été démontré qu'il est possible d'obtenir une matrice de correction, notée R_{IMU}, telle que :

$$R_{\text{IMU}} = Z_{\text{théorique}} \cdot Y^{-1}. \tag{18}$$

Cette matrice de correction permet de transformer les données brutes de l'IMU en valeurs plus conformes au comportement physique théorique, améliorant ainsi de manière significative la précision de l'estimation de l'orientation du bateau.

Il est important de souligner que l'équipe a rencontré de nombreuses difficultés lors de ce réglage, car l'IMU était dans une orientation différente de celle conventionnelle présentée. De plus, l'étalonnage du bateau pointant vers le haut s'est également avéré très complexe à réaliser, car tout mouvement ou inclinaison générait un vecteur complètement différent.

La matrice de correction obtenue expérimentalement est la suivante :

$$R_{\text{IMU}} = \begin{pmatrix} -0.94983 & -0.28222 & -0.04693 \\ 0.15636 & -1.03658 & -0.00337 \\ 0.00246 & -0.01864 & 0.98235 \end{pmatrix}. \tag{19}$$

Et elle s'applique à chaque nouvelle mesure prise par l'IMU.

3.2 Mesures GPS

À partir de la matrice de correction R_{IMU}, il était possible d'effectuer un contrôle des capteurs beaucoup plus précis. Grâce à l'utilisation de la fonction `sawtooth`, il était également possible de maintenir la continuité sur le cercle des angles, évitant ainsi les discontinuités lors de la mesure des orientations.

De plus, le GPS a été intégré au système. En utilisant les bibliothèques fournies, il permettait de déterminer la position du bateau en latitude et longitude avec une bonne précision. Pour tester le capteur, une expérience a été réalisée : le bateau a navigué vers l'ouest pendant 180 secondes, puis vers l'est pour la même durée. Pendant cette navigation, une série de mesures GPS a été enregistrée dans un fichier `.kml`, afin de permettre une visualisation et une analyse ultérieures de la trajectoire.

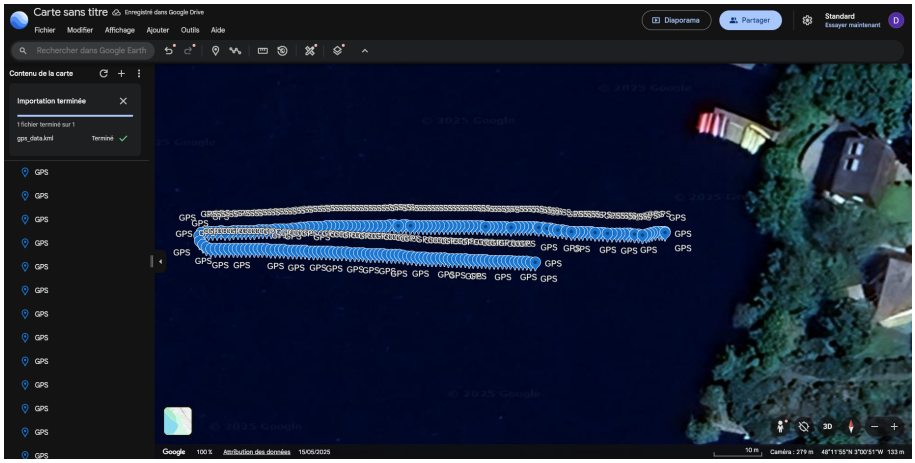


FIGURE 3 – Visualisation de la mission du bateau sur Google Earth.

Pour l'étape suivante, qui consistait à naviguer le bateau jusqu'à un point d'intérêt, il était plus pertinent d'utiliser des coordonnées exprimées en mètres plutôt qu'en latitude et longitude. Cette conversion permet de simplifier le calcul des distances et de planifier la trajectoire de manière plus intuitive.

Pour ce faire, les formules suivantes ont été employées :

$$x = \rho \cdot \cos(\text{ly}) \cdot (\text{lx} - \text{lxm}) \tag{20}$$

$$y = \rho \cdot (\text{ly} - \text{lym}) \tag{21}$$

où :

— ρ est le rayon moyen de la Terre,
— l_y est la latitude d'origine,
— l_x est la longitude,
— l_{xm} et l_{ym} sont respectivement la longitude et la latitude du point d'intérêt.

Cette transformation permet de passer d'un repère géographique (latitude/longitude) à un repère métrique local, facilitant ainsi le calcul de la trajectoire et le contrôle du déplacement du bateau vers le point cible.

4 Mission 2 : Aller à un point précis

La deuxième mission consistait à naviguer jusqu'à une bouée, dont les coordonnées étaient fournies, puis à retourner au quai. L'approche adoptée par le groupe a été de modéliser le problème dans un plan cartésien, où le bateau était toujours considéré à l'origine, et la bouée positionnée à des coordonnées décalées.

En utilisant les distances relatives entre les axes x et y, il était possible de calculer, à l'aide de la fonction tangente, l'angle nécessaire pour orienter le bateau dans la direction de la bouée. Cette approche permet de simplifier le calcul de trajectoire en transformant le problème de navigation en un problème géométrique dans un repère local.

Cependant, un problème est apparu : dans le plan cartésien, le nord était considéré comme correspondant à 90 degrés, alors que l'IMU indiquait 0 degré pour le

nord. Pour corriger ce décalage, les lignes de code suivantes ont été utilisées afin d'aligner le référentiel cartésien avec celui de l'IMU :

```
psi_ref = np.arctan2(y, x) + np.pi/2
psi_ref = (psi_ref + np.pi) % (2*np.pi) - np.pi
psi_ref = -psi_ref
```

Cette correction a permis de garantir que l'angle de référence ψ_{ref} soit correctement interprété par le système de contrôle.

Une fois que le bateau atteignait une distance inférieure à 10 mètres de la bouée, il devait automatiquement effectuer son retour vers le quai. Les coordonnées du quai étaient déjà connues et définies à l'avance, ce qui permettait de planifier le trajet de retour de manière précise.

Pour cela, la même approche en plan cartésien a été appliquée : le bateau était considéré à l'origine, et le quai était positionné selon ses coordonnées relatives (x, y). En utilisant la fonction tangente pour calculer l'angle nécessaire, le système de contrôle pouvait orienter le bateau dans la direction du quai, garantissant ainsi un retour sûr et précis après l'atteinte de la bouée.

Visualisation des résultats

Pour analyser les performances du bateau lors de la mission, plusieurs visualisations ont été réalisées. Trois graphiques principaux ont été considérés :

- L'évolution de l'angle de référence ψ_{ref} par rapport à l'angle mesuré par l'IMU, permettant de vérifier la précision de l'orientation.
- La distance entre le bateau et le point cible (bouée), montrant comment le bateau se rapprochait progressivement de la bouée avant d'inverser sa trajectoire.
- La trajectoire GPS du bateau visualisée dans Google Earth, mettant en évidence le chemin réel parcouru et la correspondance avec la trajectoire théorique.

Ces visualisations fournissent une compréhension complète du comportement du bateau, à la fois au niveau du contrôle de l'orientation et de la navigation vers le point d'intérêt, ainsi que de la précision du GPS.

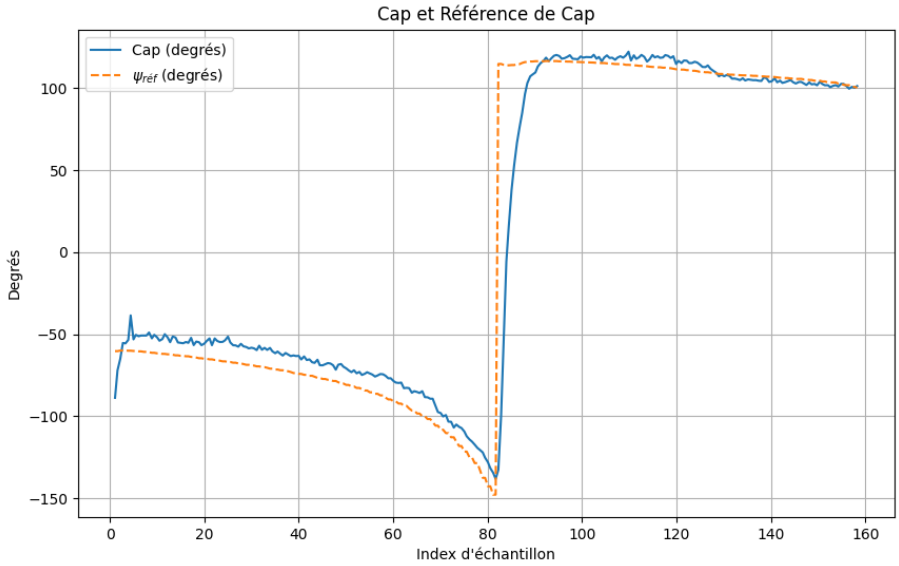


FIGURE 4 – Comparaison entre l'angle de référence ψ_{ref} et l'angle mesuré par l'IMU.

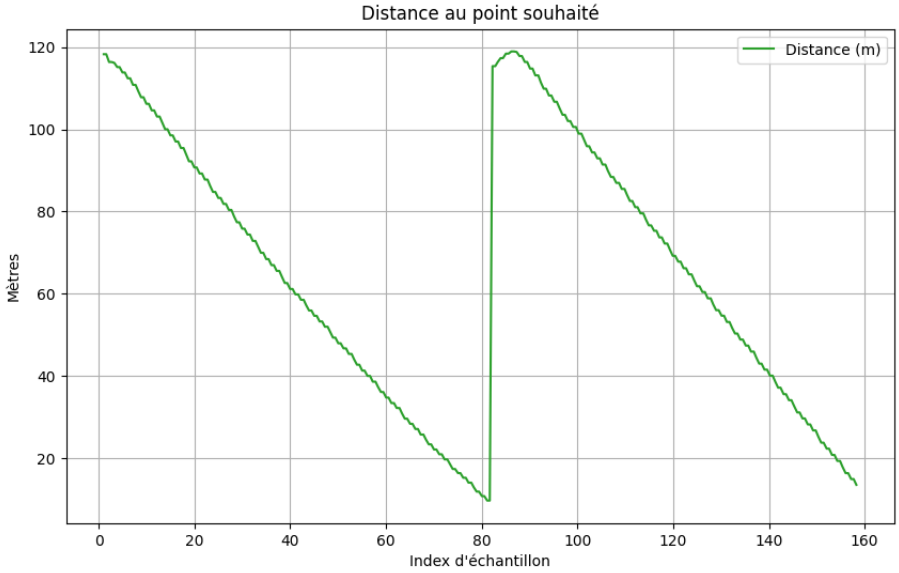


FIGURE 5 – Évolution de la distance entre le bateau et le point cible au cours de la mission.

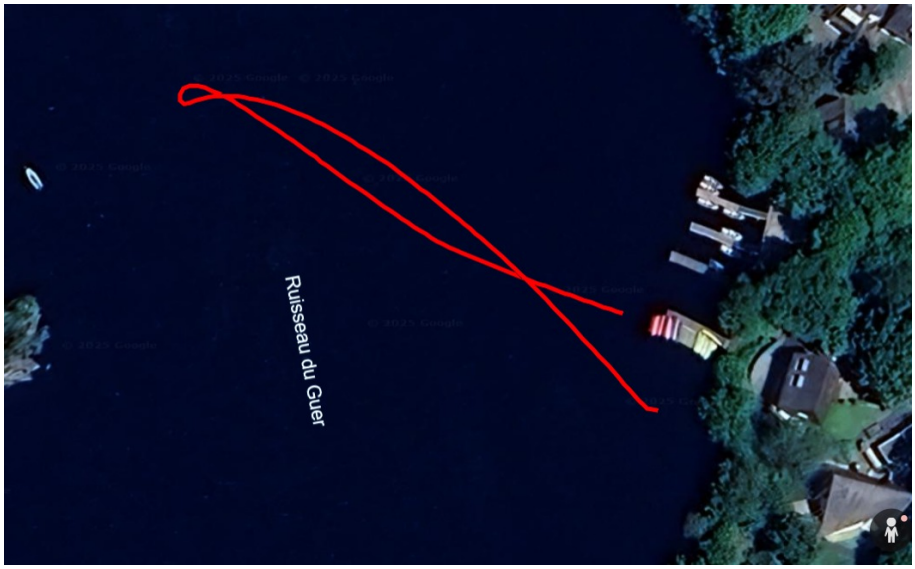


FIGURE 6 – Trajectoire GPS du bateau visualisée sur Google Earth.

5 Conclusion

À partir des cours dispensés durant les activités au lac de Guerlédan, il a été possible de réaliser les deux premières missions proposées. Ainsi, le bateau a réussi à se diriger vers l'ouest pendant 15 secondes, puis à se rendre jusqu'à la bouée et à revenir. En revanche, il n'y a pas eu de temps suffisant pour accomplir la troisième mission, qui consistait à suivre une trajectoire particulière le long du lac.

Malgré les efforts consacrés à une bonne calibration du magnétomètre, un écart de l'ordre de 15° a été observé par rapport à une même direction vérifiée à l'aide des boussoles de téléphones portables sur le site d'essai. Ce problème à affecté les deux missions, notamment la première, de se déplacer vers l'ouest. Parmi les causes possibles de cette erreur, il convient de souligner la différence entre le nord magnétique de la Terre, indiqué par le magnétomètre, et le nord géographique fourni par le GPS. Cette différence présente un ordre de grandeur relativement proche de l'erreur constatée.

Lors de l'exécution de la deuxième mission, il a été possible d'observer que le bateau ne s'est pas rendu à l'emplacement exact de la bouée (Figure 6). Cela s'explique par plusieurs facteurs, notamment l'imprécision des coordonnées GPS obtenues par rapport à la position réelle de la bouée ainsi que l'oscillation de celle-ci au cours du temps. De plus, afin d'éviter que le bateau cherche indéfiniment la position exacte de la bouée, une contrainte de proximité de 10 mètres a été ajoutée via logiciel. Quant à la trajectoire suivie par le bateau jusqu'à la bouée, son aspect courbé s'explique par l'effet des courants, du vent et par l'imprécision du magnétomètre. À mesure que le bateau s'approche des coordonnées où il estime que se trouve la bouée, la fonction de correction devient plus sensible et le bateau augmente l'intensité des ajustements de cap. Un effet similaire est observé sur le trajet de retour, mais dans une moindre mesure.

Eliaz Yahia, Evann Michel, Noé Carrayrou

Rapport : Guerlédan première semaine

Introduction

Au cours de cette première semaine de travail sur le projet *Guerlédan*, nous avons travaillé sur un **DD Boat** (bateau robotisé) équipé des composants suivants :

- Un **Raspberry Pi** assurant le contrôle général du système,
- Une **centrale inertielle 9 axes** (accéléromètre, gyromètre et magnétomètre),
- Un **module GPS** (émetteur/récepteur),
- Deux **motoréducteurs** entraînant les hélices de propulsion.

Notre bateau, identifié sous le **numéro 11**, avait pour objectif final d'embarquer un programme capable de **suivre automatiquement une trajectoire autour d'une cible mobile**, tout en respectant les contraintes imposées.

Pour atteindre cet objectif, plusieurs étapes intermédiaires ont été nécessaires :

- **Calibrer le magnétomètre** afin de l'utiliser comme compas et ainsi connaître le cap du bateau en temps réel ;
- **Rendre exploitables les données GPS** pour permettre au bateau de rallier une position cible (par exemple une bouée) depuis un point de départ aléatoire, puis de revenir vers l'opérateur de manière autonome ;
- Enfin, **adapter le code de navigation** pour assurer le suivi dynamique d'une cible en mouvement selon une trajectoire définie.

Pendant la semaine du 3 au 7 novembre 2025, nous avons eu l'opportunité de travailler sur des robots DDboat et de les programmer à effectuer différentes missions sur le lac de Guerlédan. Tout au long de la semaine, nous avons tenté de remplir les différents objectifs fixés en faisant face à des problèmes divers, mélangeant difficultés matérielles, logiciel et théoriques.

Les missions étaient au nombre de 3 :

- Calibrer le magnétomètre et aller 15 secondes vers l'ouest
- Aller jusqu'à une balise et revenir grâce à un suivi GPS
- Suivre un point GPS en mouvement sous la forme d'un autre DDBoat

Eliaz Yahia, Evann Michel, Noé Carrayrou

Calibration IMU :

La première étape consiste en la calibration du magnétomètre. Celui-ci renvoie des vecteurs représentant le champ magnétique qu'il mesure, exprimés dans un repère propre au capteur, noté (0, X, Y, Z). Il est donc essentiel, lors de la calibration, de ne pas oublier de compenser l'écart entre le repère du capteur et celui du robot, un décalage qui peut résulter du positionnement imprécis de l'IMU sur le robot.

Le problème qui motive la calibration du magnétomètre réside dans le fait qu'il mesure un champ magnétique qui est, en pratique, la somme de deux éléments :

Le champ magnétique ambiant du milieu (ici, le champ terrestre ou magnétique ambiant du bateau).

Les champs magnétiques parasites générés par certains composants du robot. Ces perturbations se divisent en deux catégories principales :

a. Le "**hard iron**", qui désigne les champs magnétiques permanents générés par des aimants (par exemple, des moteurs, des capteurs, ou d'autres composants avec des matériaux magnétiques permanents).
b. Le "**soft iron**", qui concerne des matériaux non magnétiques mais qui peuvent être influencés par un champ magnétique externe, modifiant ainsi la direction et l'intensité du champ mesuré. C'est notamment le cas des composants électroniques ou des pièces métalliques non magnétiques du robot.

Concrètement, l'effet du "hard iron" se manifeste par l'ajout d'un champ magnétique constant au champ réel, mobile avec le capteur. Cela entraîne simplement un déplacement des résultats mesurés, ce qui peut être corrigé par une translation des données du magnétomètre.
En revanche, le phénomène de "soft iron" est plus complexe, car il induit une déformation géométrique des mesures. Par conséquent, le nuage de points qui serait idéalement sphérique, représentant les mesures du champ magnétique dans différentes orientations, prend une forme ellipsoïdale.

Lors de la calibration, l'objectif est donc de déterminer une transformation affine qui permettra de corriger ces deux effets, en compensant à la fois le décalage du *hard iron* et la distorsion géométrique causée par le *soft iron*.

Eliaz Yahia, Evann Michel, Noé Carrayrou

La méthode de calibration employée consiste à ajuster un modèle d'ellipsoïde aux données mesurées par le magnétomètre. Elle se déroule en plusieurs étapes :

1. **Ajustement d'une équation quadratique** à un nuage de points, permettant de déterminer les effets de *hard iron* (déplacement constant) et de *soft iron* (distorsion géométrique).
2. **Calcul du centre** de l'ellipsoïde et de la **matrice de transformation affine** pour corriger ces perturbations.
3. **Application de la transformation** afin de recentrer le nuage de points et le transformer en une sphère centrée, compensant ainsi les effets de décalage et de déformation.

Un magnétomètre 3 axes renvoie des mesures magnétiques selon des axes X, Y et Z définis par le montage du capteur.

Notons $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ une mesure.

L'équation d'une ellipsoïde est :

$$(1)\quad p_1x_1^2 + p_2x_2^2 + p_3x_3^2 + p_4x_1x_2 + p_5x_1x_3 + p_6x_2x_3 + p_7x_1 + p_8x_2 + p_9x_3 = 1$$

Qui peut être remis sous la forme :

$$f(x) = x^T Ax + C^T x = 1$$

Avec : $A = \begin{pmatrix} p_1 & \frac{p_4}{2} & \frac{p_5}{2} \\ \frac{p_4}{2} & p_2 & \frac{p_6}{2} \\ \frac{p_5}{2} & \frac{p_6}{2} & p_3 \end{pmatrix}$, $C^T = (p_7 \quad p_8 \quad p_9)$ et $p_1, \dots, p_9 \geq 0$

Ou encore :

$$g(x) = (x - b)^T Q(x - b) = 1$$

Notons que g est l'équation d'une ellipsoïde centrée en b

Nous cherchons à calibrer la boussole, c'est-à-dire à obtenir $y = D(x - b)$ avec $\|y\| = 1$

Regardons le gradient de f :

$$\frac{df}{dx}(x) = 2Ax + C^T = 0$$

Donc nous avons au centre de l'ellipsoïde : $\frac{df}{dx}(b) = 2Ab + C^T = 0$

D'où : $b = -\frac{1}{2}A^{-1}C^T$

Par ailleurs : en posant $y = x - b$ c'est-à-dire en recentrant les données on a :

$$x^T Ax + C^T x = (y+b)^T A(y+b) + C^T(y+b) = 1$$

$$\Leftrightarrow y^T Ay + b^T Ay + y^T Ab + b^T Ab + C^T y + C^T b = 1$$

Or, A est symétrique positive et le résultat précédent nous donne également que $C^T = -2Ab$, donc :

$$\Leftrightarrow y^T Ay + 2b^T Ay + b^T Ab - 2b^T Ab = 1$$

$$\Leftrightarrow y^T Ay - b^T Ab = 1$$

$$\Leftrightarrow \frac{y^T Ay}{1 + b^T Ab} = 1$$

Par identification, on a donc $Q = \frac{A}{1+b^T Ab}$

On remarque donc que Q est symétrique réelle positive, donc qu'elle admet des valeurs propres toutes positives. Nous pouvons donc utiliser la racine carrée matricielle, ce qui nous donne :

$$g(x) = (x-b)^T \sqrt{Q}^T * \sqrt{Q}(x-b) = 1$$

$$\Leftrightarrow \left\| \sqrt{Q}(x-b) \right\|^2 = 1$$

D'où $y = \sqrt{Q}(x-b)$ correspond à une juste calibration de la boussole.

Cependant, il reste la question de comment obtenir les $p_1, \dots, p_9$.

Pour cela, nous allons procéder par moindres carrés, à l'aide de (1) et n mesures :

$$M * P = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Avec : $P = \begin{pmatrix} p_1 \\ \vdots \\ p_9 \end{pmatrix}$ et

$$M = \begin{pmatrix} x_1(1)^2 & x_2(1)^2 & x_3(1)^2 & x_1x_2(1) & x_1x_3(1) & x_2x_3(1) & x_1(1) & x_2(1) & x_3(1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1(n)^2 & x_2(n)^2 & x_3(n)^2 & x_1x_2(n) & x_1x_3(n) & x_2x_3(n) & x_1(n) & x_2(n) & x_3(n) \end{pmatrix}$$

Donc $P = (M^T M)^{-1} M^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$

Ainsi, en prenant différentes mesures du champ magnétique perçu dans toutes les orientations possibles du robot, nous pouvons estimer de façon relativement précise les paramètres de P, ce qui nous permet ensuite de déterminer Q et b pour calibrer correctement le magnétomètre.

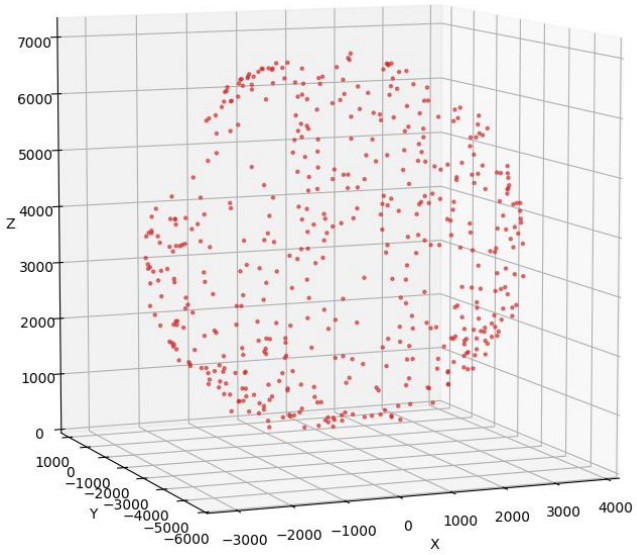


Figure 1 : Nuage de point ellipsoïdal issu des données brutes du magnétomètre

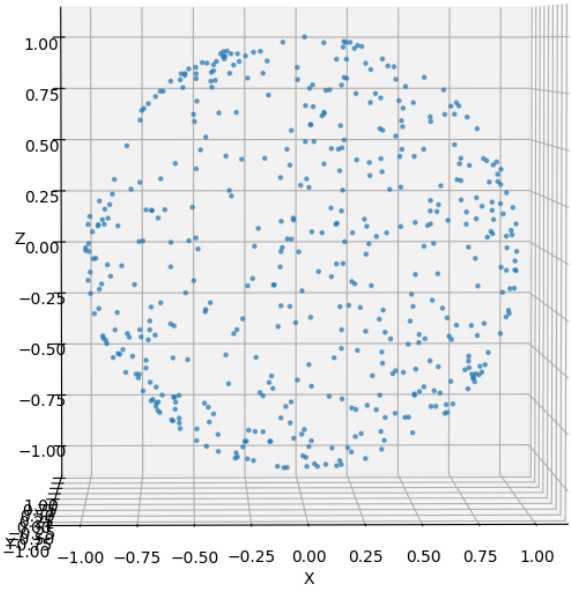


Figure 2 : Nuage de points sphérique et centré des données du magnétomètre après calibration

Ensuite, nous cherchons à obtenir les angles d'Euler du bateau à partir des mesures de champ magnétique et d'accélération. Ce qui nous intéresse plus particulièrement pour la suite est le cap du robot par rapport au nord, afin de pouvoir ensuite le diriger dans une direction précise.

En notant y_n le champ magnétique mesuré, corrigé et normalisé et a_n le vecteur accélération (permettant d'estimer l'axe vertical) mesuré et normalisé, dans le repère du robot, il est possible d'obtenir la matrice de rotation de celui-ci :

$$R = (Y \quad a_n \wedge Y \quad a_n)$$

Avec $Y = \frac{y_n - (y_n^T a_n) a_n}{\| y_n - (y_n^T a_n) a_n \|}$

Ce qui nous permet d'obtenir : $\phi = \arctan(R_{32}, R_{33})$

$$\theta = -\arccos(R_{31})$$

$$\psi = arctan2(R_{21}, R_{11})$$

Les mesures d'accélération étant relativement bruitées avec le mouvement du robot (vagues, vent), il a été nécessaire d'appliquer un filtre médian glissant sur les mesures d'accélération, afin de déterminer plus précisément la verticale.

Pour nous permettre d'exploiter aisément les données retournées par le magnétomètre après calibration, nous avons fait en sorte d'aligner le cap zéro du robot avec le nord réel. Une fois cela fait, le robot est capable de déterminer son cap ψ, avec une précision d'environ 5 degrés, ce qui est suffisant pour les besoins les missions que nous devons réaliser.

Mouvement vers point GPS :

Notre objectif est maintenant que le bateau rallie un point GPS puis revienne vers un point de récupération de manière autonome.
Pour réaliser cela, nous devons dans un premier temps traiter les données, GPS reçues par le robot. Nous convertissons alors les données GPS angulaires globale en coordonnées locale dans un repère (0, X, Y) dont on aura pris soin de placer l'origine comme demandé par les consignes à suivre. Les coordonnées dans le repère local de la position du robot et de la balise à atteindre nous donnent par calcul trigonométrique le cap à suivre par le robot pour atteindre son objectif, il ne reste alors plus qu'à commander les moteurs du robot en fonction de cela.

Nous avons opté pour une régulation proportionnelle dérivée par rapport à son erreur de cap. En effet, un correcteur proportionnel ne suffisait pas avoir des résultats. Cela

est dû aux perturbations auxquelles le robot est constamment confronté comme le vent, les vagues et les kayaks des copaings qui ne savent pas pagayer droit. Après plusieurs essais, nous avons déterminé des paramètres K_p et K_d permettant d'avoir une régulation en cap relativement stable et précise.

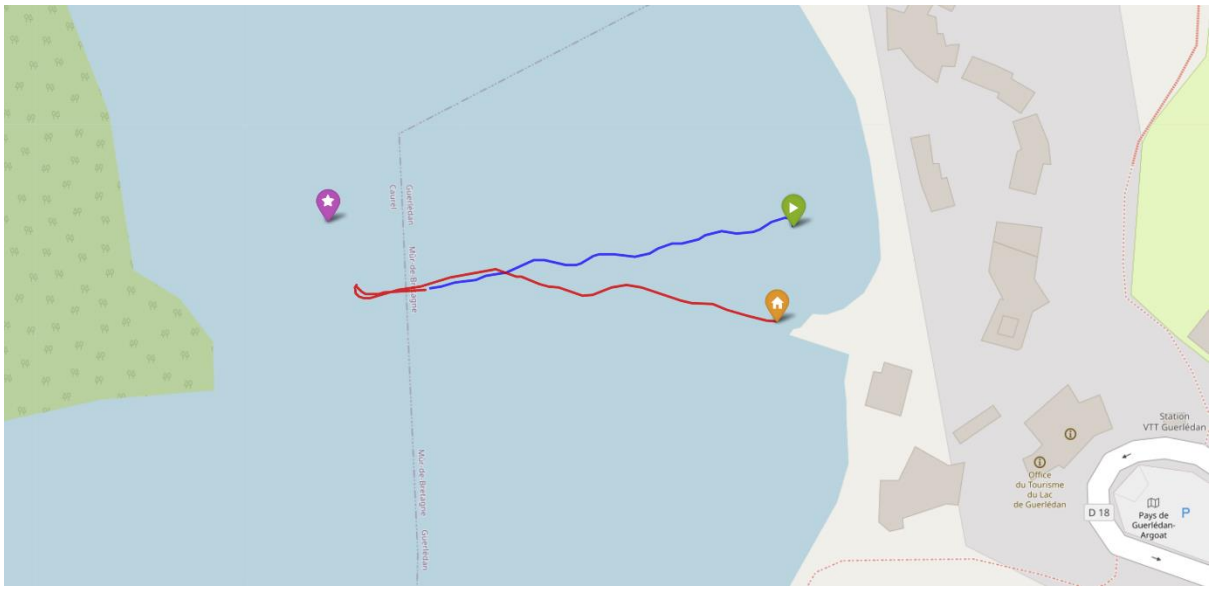


Figure 3 : Trajectoire suivie par le bateau lors de son aller-retour à la balise.

Mouvement vers un point GPS mobile :

Notre robot étant désormais capable de se diriger vers un point GPS donné. Il est désormais possible de faire suivre une trajectoire particulière au robot, en convertissant les positions successives sur la trajectoire, en point GPS à atteindre.

L'objectif de la mission était de faire effectuer une simulation de capture d'un DDboat, avec plusieurs autres DDboat. Ceux-ci devaient tourner autour de la cible, en réduisant progressivement la distance. Comme aucun programme de communication entre robots n'était fait, chaque robot devait avoir la même trajectoire de cible intégrée dans son programme, avec sa position autour de la cible déphasée par rapport aux autres poursuiveurs afin qu'il n'y ait pas de collision.

Une fois les paramètres de la trajectoire à suivre obtenus, il suffisait d'une fonction donnant les coordonnées du point où le robot devait se trouver à chaque instant, dans le repère local, pour que le robot détermine ensuite comment ajuster son cap pour suivre de façon relativement précise le point mobile.

Durant les essais, nous avons remarqué que le robot se mettait souvent à dépasser le point. Il devait alors opérer un demi-tour pour se remettre à le suivre. Cela

était dû au fait que la commande de vitesse était constante et trop élevée par rapport à la vitesse du point. Ce qui est nécessaire pour le rattraper lorsque le robot en est éloigné, mais devient problématique à courte distance. Pour pallier cela, nous avons fait en sorte que la commande de vitesse s'adapte par rapport à la distance entre le robot et le point à atteindre, celui-ci ralentit donc à l'approche du point et réaccélère si la distance devient trop grande. Cela permettait, après ajustement des paramètres de correction de vitesse grâce à des essais, d'avoir notre robot qui suit le point à une distance de 5 mètres en moyenne tout au long de la mission.
Nous pouvons voir que lorsque de notre dernier essai, vendredi matin, la trajectoire de notre robot était relativement précise.

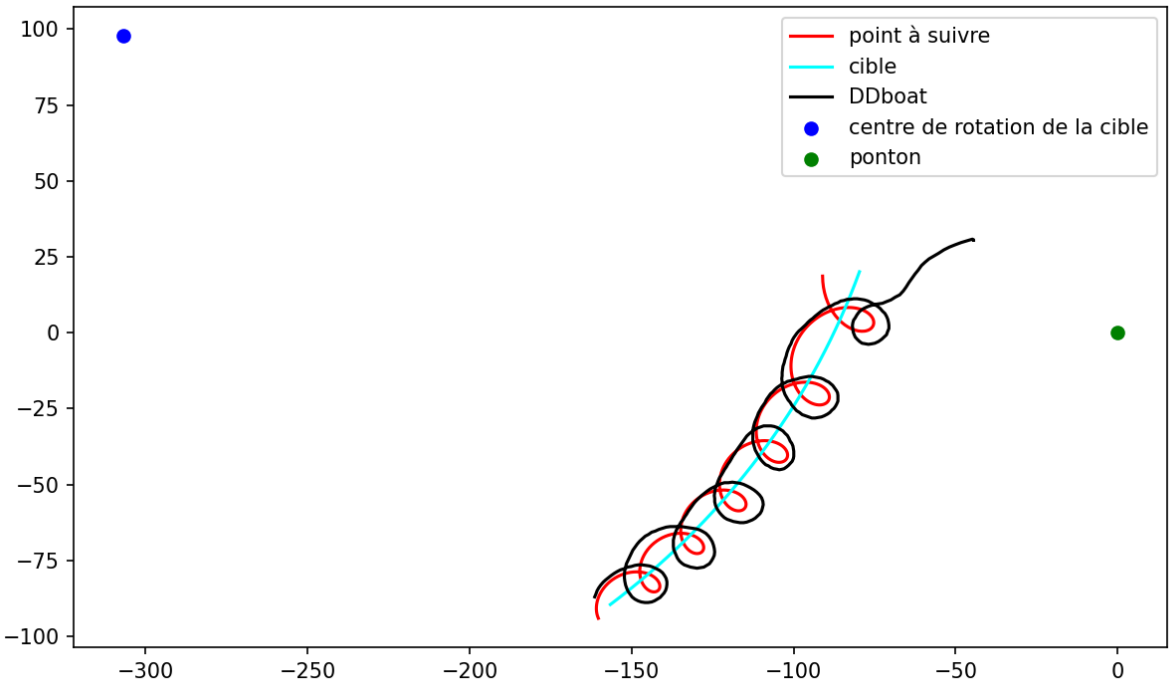


Figure 4: Trajectoires du DDboat et de sa cible dans le repère local

Avec plus de temps, nous aurions pu conduire plus de passages d'essais, afin d'affiner les coefficients de contrôle du robot, permettant in fine de gagner en précision.

Lors de l'exécution de la mission en groupe, la trajectoire que nous obtenons est la suivante :

Figure 5 : Trajectoire du DDboat autour de la trajectoire supposée de la cible lors de l'essai en groupe

On note que malgré un temps de convergence plus long que lors de notre essai précédent, le robot converge bien vers une trajectoire autour de la cible. Les images tournées lors de la réalisation de la manipulation le montrent aussi.

Rapport de Projet

Semaine au Lac de Guerlédan

Auteur(s) : Escoffier Jonathan, Habigand Louis

Github : https://gitlab.com/groupe-escoffier-habigand/ddboat_guerledan

Date : 17 novembre 2025

Table des matières

1 Suivi de cap

Objectif : calibrer le cap du bateau et assurer le suivi d'un cap défini.

Peu importe la manière dont nous effectuions les mesures avec l'IMU, nous rencontrions systématiquement le même problème. Il nous était impossible d'obtenir une sphère correcte en sortie du programme. Après plusieurs tests croisés avec le code de nos voisins et un changement de bateau, nous avons enfin obtenu un résultat satisfaisant (cf. figure 2).

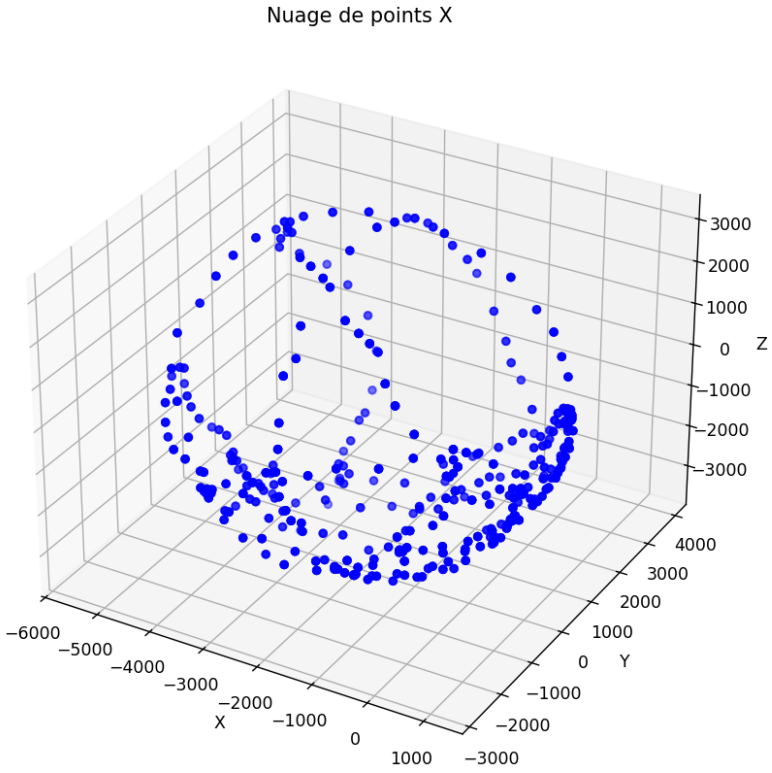


FIGURE 1 – Ellipse de l'IMU

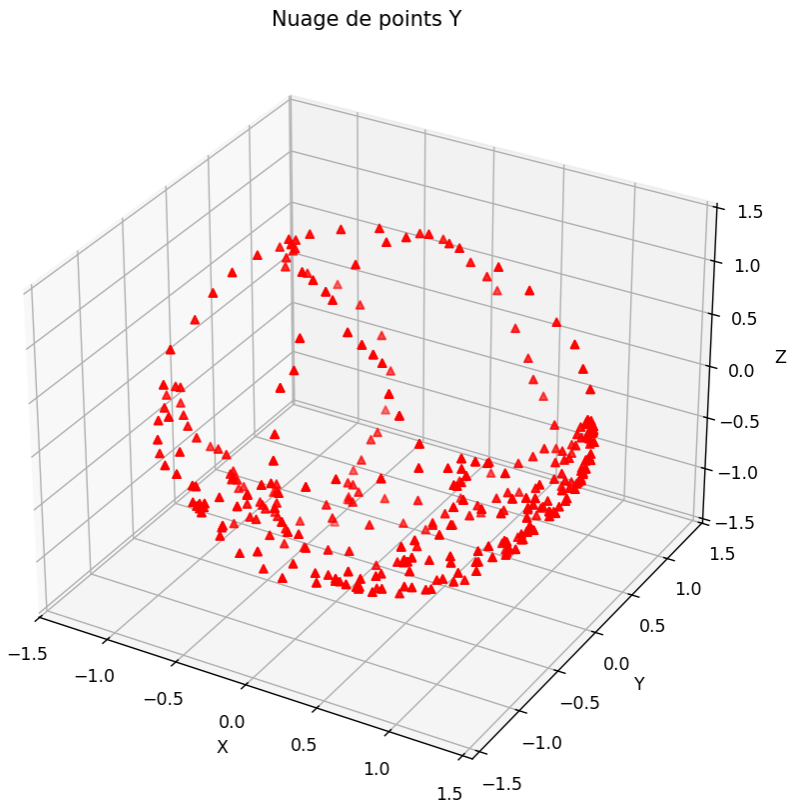


FIGURE 2 – Sphère de l'IMU calibrée

Après obtention des matrices de calibration, nous avons commencé à développer le programme de contrôle des moteurs : le Fast Forward Control. Ce programme nous a permis de gérer efficacement le comportement du bateau lors d'un changement de cap.

Enfin, nous avons pu finaliser notre premier programme complet pour le bateau : le suivi de cap. En fournissant simplement le cap à suivre, le bateau se dirigeait automatiquement dans la direction souhaitée.

2 Bouée GPS

Objectif : mettre en place la navigation vers un objectif GPS (bouée).

Le deuxième objectif consistait à implémenter un code permettant au bateau de suivre un waypoint. Une bouée fixe, positionnée précisément sur le lac, servait de cible. Le programme a été conçu en plusieurs étapes : d'abord, récupération et conversion des données GPS du bateau ; ensuite, calcul du cap à suivre pour atteindre le waypoint. Le bateau se dirigeait alors automatiquement vers la cible et s'arrêtait lorsqu'une distance minimale était atteinte.

Lors de la réalisation de cette mission, nous avons rencontré plusieurs difficultés, notamment des faux contacts dans certains fils et un dérèglement de l'IMU. La recalibration a été longue et fastidieuse, et nous avons finalement dû changer de bateau une nouvelle fois.

Plusieurs tests ont été effectués avec des distances minimales de 6 mètres et 2 mètres (figures à l'appui).

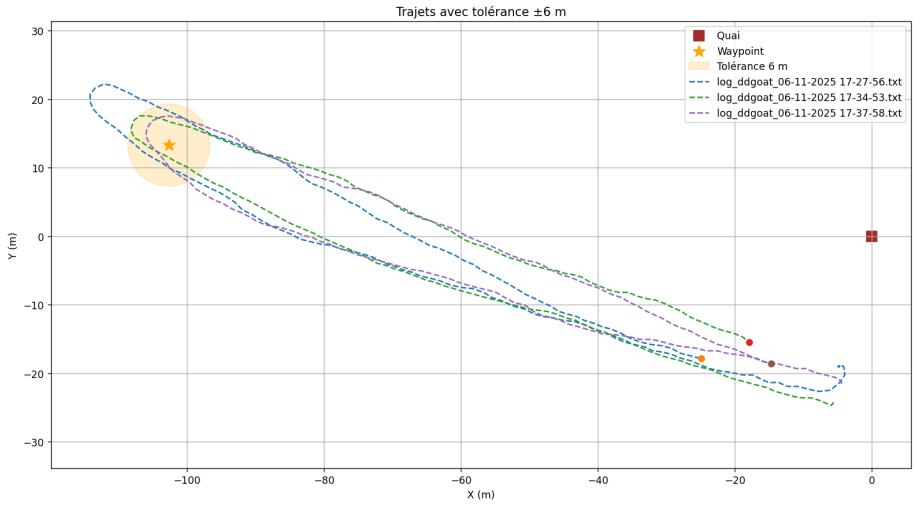


FIGURE 3 – Aller-retour à la bouée avec distance minimale de 6 mètres

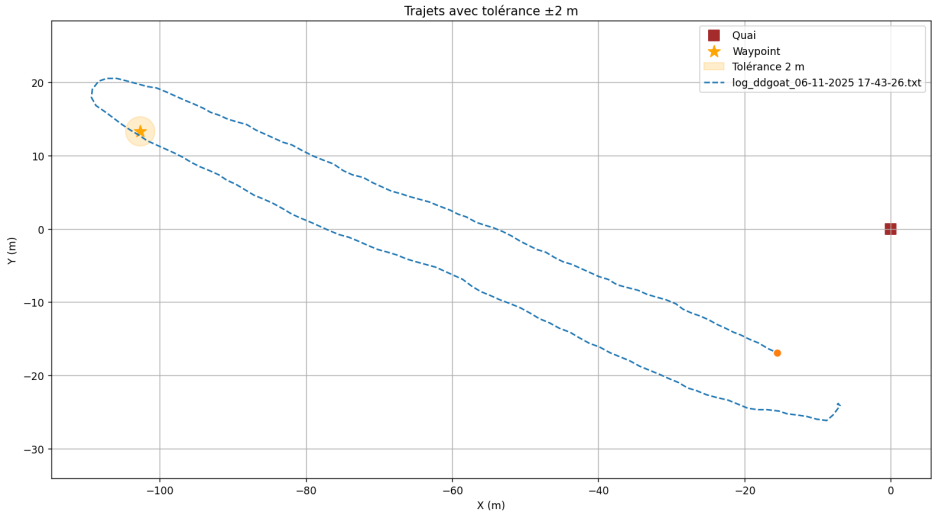


FIGURE 4 – Aller-retour à la bouée avec distance minimale de 2 mètres

3 Swarm

Objectif : tester le comportement du bateau en essaim.

La troisième et dernière mission consistait à implémenter une fonction permettant au bateau de suivre une trajectoire temporelle en temps réel. Malheureusement, nous n'avons pas pu finaliser cette mission par manque de temps. Cependant, nous avons pu effectuer quelques tests afin d'observer le comportement du bateau. Les figures suivantes illustrent un essai de navigation en essaim avec différentes trajectoires.

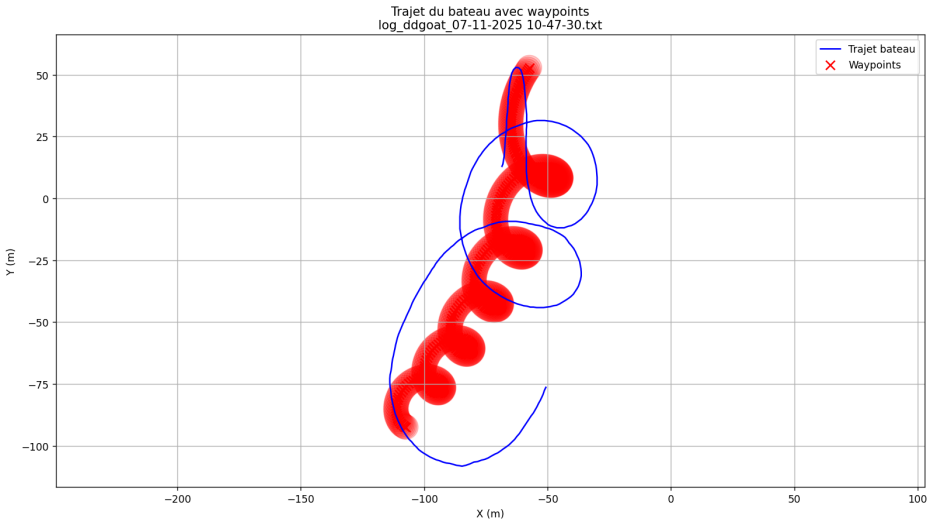


FIGURE 5 – Essai de navigation en essaim

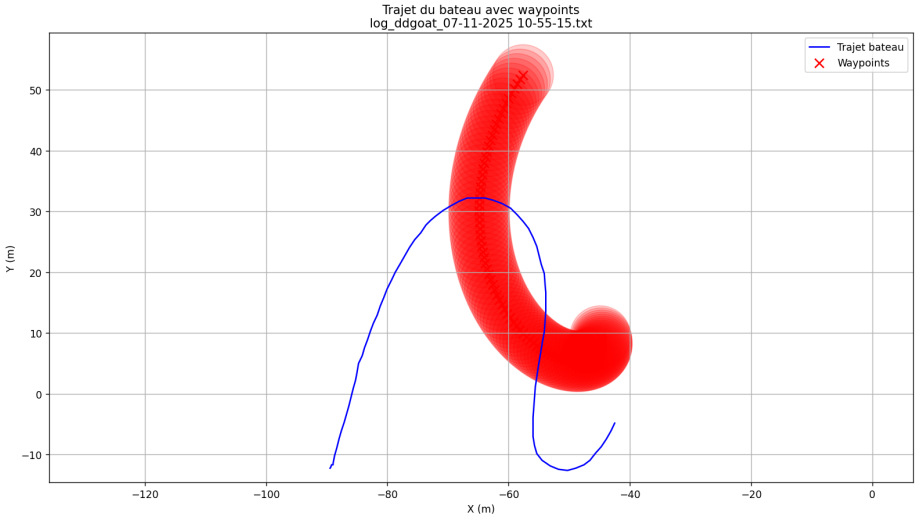


FIGURE 6 – Essai de navigation en essaim

Rapport projet Guerlédan

Juliette LOURDAULT juliette.lourdault@ensta.fr
Guilhem MERLET guilhem.merlet@ensta.fr
Florian MILIN florian.milin@ensta.fr

Table des matières

1 Introduction

Le projet Guerlédan consiste à programmer le DDboat, afin qu'il réalise l'objectif : faire un aller-retour à la bouée. Ce rapport détaille les différentes étapes du projet : la calibration de l'IMU, la détermination du cap, la navigation vers l'ouest, puis le trajet aller-retour jusqu'à la bouée.

2 Calibration

L'objectif de cette partie est de trouver le cap du bateau à l'aide d'un magnétomètre.

2.1 Théorie

Dans l'espace intersidéral, le champ magnétique ressenti est une sphère. Cependant, le soft-iron déforme cette sphère en une ellipse, et le hard-iron déplace cette ellipse en la centrant en $\mathbf{b}$.

L'équation d'une ellipse dans $\mathbb{R}^3$ peut s'écrire :

$$f(\mathbf{x}) = p_1x_1^2+p_2x_2^2+p_3x_3^2+p_4x_1x_2+p_5x_1x_3+p_6x_2x_3+p_7x_1+p_8x_2+p_9x_3, \quad \forall \mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$$

On cherche le vecteur des coefficients :

$$\mathbf{p} = (p_1, p_2, \dots, p_9)^T$$

Pour cela, nous disposons de n mesures $\mathbf{x}(1), \dots, \mathbf{x}(n)$, avec $\mathbf{x}(k) = (x_1(k), x_2(k), x_3(k))^T$. La forme quadratique peut alors s'écrire sous forme matricielle :

$$\mathbf{y} = M\mathbf{p}, \quad \mathbf{p} = (p_1, p_2, \dots, p_9)^T$$

où la matrice M est définie par :

$$M = \begin{pmatrix} x_1(1)^2 & x_2(1)^2 & x_3(1)^2 & x_1(1)x_2(1) & x_1(1)x_3(1) & x_2(1)x_3(1) & x_1(1) & x_2(1) & x_3(1) \\ x_1(2)^2 & x_2(2)^2 & x_3(2)^2 & x_1(2)x_2(2) & x_1(2)x_3(2) & x_2(2)x_3(2) & x_1(2) & x_2(2) & x_3(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1(n)^2 & x_2(n)^2 & x_3(n)^2 & x_1(n)x_2(n) & x_1(n)x_3(n) & x_2(n)x_3(n) & x_1(n) & x_2(n) & x_3(n) \end{pmatrix}$$

Les moindres carrés nous donnent alors :

$$\mathbf{p} \approx (M^TM)^{-1}M^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Une ellipse de centre $\mathbf{b}$ peut s'écrire :

$$f(\mathbf{x}) = (\mathbf{x}-\mathbf{b})^TQ(\mathbf{x}-\mathbf{b}) = 1, \quad \text{où } Q \in S_3(\mathbb{R}^+), \text{si } \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{ est compris dans l'ellipse, ce qui est notre cas ;}$$

Le gradient de cette fonction est :

$$\frac{df}{d\mathbf{x}} = 2(\mathbf{x}-\mathbf{b})^\top Q = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_3} \end{bmatrix}$$

La matrice Q et le centre $\mathbf{b}$ s'écrivent :

$$Q = \begin{bmatrix} p_1 & \frac{p_4}{2} & \frac{p_5}{2} \\ \frac{p_4}{2} & p_2 & \frac{p_6}{2} \\ \frac{p_5}{2} & \frac{p_6}{2} & p_3 \end{bmatrix}, \quad \mathbf{b} = -\frac{1}{2}Q^{-1} \begin{bmatrix} p_7 \\ p_8 \\ p_9 \end{bmatrix}$$

$(\mathbf{x}-\mathbf{b})^T Q(\mathbf{x}-\mathbf{b}) = 1$ donc pour transformer l'ellipse en sphère centrée, il faut alors tracer $\mathbf{y}$ tel que :

$$\mathbf{y} = \sqrt{Q}(\mathbf{x}-\mathbf{b})$$

2.2 Mise en pratique

Pour la mise en pratique, nous avons récupéré les données de l'IMU figure 1. Ces données permettent de visualiser l'ellipse magnétique mesurée, qui est déformée par le soft-iron et déplacée par le hard-iron.

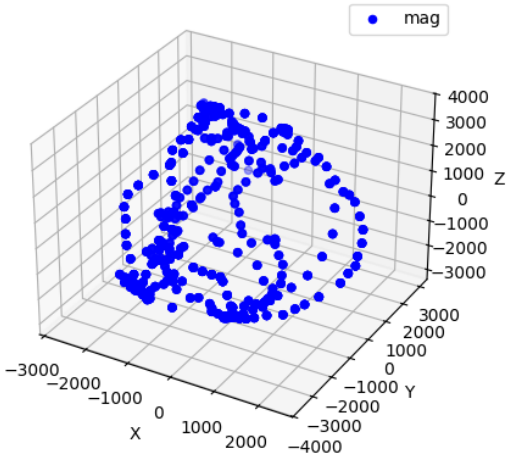


FIGURE 1 – Représentation de l'ellipse magnétique mesurée par l'IMU.

Comme montré sur la figure 1, l'ellipse est centrée sur $\mathbf{b}$ et déformée selon les effets du soft-iron.

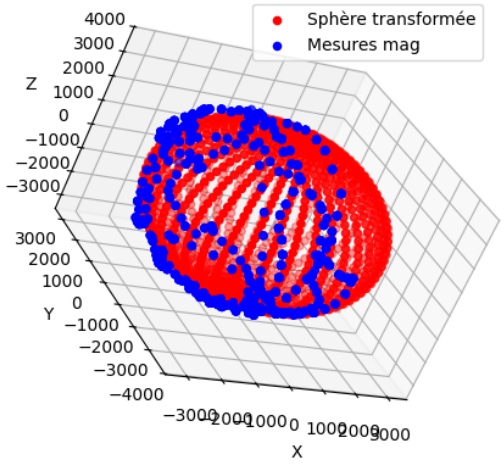


FIGURE 2 – Représentation de la sphère magnétique et de l'ellipse théorique.

La figure 2 montre en rouge la sphère unitaire transformée par la fonction :

$$\mathbf{x} \mapsto Q^{-\frac{1}{2}}\mathbf{x} + \mathbf{b}$$

On voit que les points bleus mesurés ne sont sur l'ellipse.

3 Détermination du cap

3.1 Théorie

Notons $\vec{y}$ le vecteur du champ magnétique terrestre mesuré par l'IMU du bateau trouvé précédemment. En orientant l'IMU vers le nord, l'ouest et le haut, on en censé obtenir respectivement :

$$y_N = \begin{bmatrix} cos(I) \\ 0 \\ sin(I) \end{bmatrix}, y_O = \begin{bmatrix} 0 \\ -cos(I) \\ -sin(I) \end{bmatrix}, y_{UP} = \begin{bmatrix} sin(I) \\ 0 \\ cos(I) \end{bmatrix}$$

Où $I = 64$ř l'angle d'incidence des lignes de champs magnétique à notre latitude.

Cependant, ceci est vrai si l'IMU est axé avec le bateau. Pour compenser le potentiel décalage d'axe, on applique une matrice notée R_{IMU} à $\vec{y}$, définie ainsi :

$$R_{IMU} = \begin{bmatrix} y_N & y_O & y_{UP} \end{bmatrix} \begin{bmatrix} y_{N,mes} & y_{O,mes} & y_{UP,mes} \end{bmatrix}^{-1}$$

On utilise dorénavant $\vec{z} = R_{IMU}\vec{y}$ pour vecteur champ magnétique terrestre ressenti par le bateau. On note$\vec{a_1}$ l'accélération du bateau. On note R, la matrice de rotation.

$$R = \begin{bmatrix} \dfrac{\vec{z} - (\vec{z}^T\vec{a}_1)\,\vec{a}_1}{\left\|\vec{z} - (\vec{z}^T\vec{a}_1)\,\vec{a}_1\right\|} & \dfrac{\vec{a}_1 \times (\vec{a}_1 \times \vec{z}) - (\vec{z}^T\vec{a}_1)\,\vec{a}_1}{\left\|\vec{a}_1 \times (\vec{a}_1 \times \vec{z}) - (\vec{z}^T\vec{a}_1)\,\vec{a}_1\right\|} & \vec{a}_1 \end{bmatrix}$$

On suppose ici que le bateau reste à peu prêt à plat sur l'eau. on prend alors pour accélération :

$$\mathbf{a_1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Ici, c'est le cap qui nous intéresse et donc l'angle d'Euler ψ. On trouve ce cap avec la matrice R définie précédemment.

$$\psi = arctan2(R_{3,2}, R_{3,3})$$

3.2 Mise en pratique : aller vers l'ouest

Pour la mise en pratique, nous avons récupéré les données de notre calibration faites précédemment. Ensuite nous récupérions en temps réelle les données du magnétomètre et de l'accéléromètre, ce qui nous permet de calculer en temps réel le cap de notre bateau. Cependant, atan2 ne donne pas un cap compris entre 0° et 360°. Après la transformation de notre cap de radian en degré, nous avions un problème : le Nord était à 180°, le Sud à 0°, l'Est à 270° et l'Ouest à 90°. Nous avons donc appliqué une transformation :

$$new\_cap = (cap + 360) \bmod 360$$

Nous avons, par la suite utilisé la fonction *sawtooth* afin de trouver ω_{d}, la vitesse angulaire à donner à nos moteurs, en donnant en argument la différence angulaire entre l'ouest et le cap actuel ne notre bateau. Nous avons ensuite créé une FFC afin de donner des consignes de vitesse angulaire à nos moteurs en fonction de la différence de cap précédemment calculée.

Vidéo de notre bateau allant vers l'ouest : DD-Boat allant vers l'ouest

4 Aller à la bouée

L'objectif suivant est de lâcher le bateau au ponton, qu'il fasse le tour d'une bouée et qu'il revienne.

4.1 Trouver le cap désiré

Le bateau étant capable d'aller vers l'ouest, il suffit de changer, en paramètre de notre *sawtooth*, le 270° correspondant à l'ouest par une variable ψ_d, le cap désiré. Pour cela, on définit un repère orthonormé $(\tilde{x}, \tilde{y})$ attaché au bateau, dont l'origine O correspond au bout du ponton. Le vecteur unitaire $\vec{e}_{\tilde{x}}$ pointe vers l'est et $\vec{e}_{\tilde{y}}$ pointe vers le nord. Le bateau nous donne ses coordonnées GPS, donc pour trouver $\tilde{x}$ et $\tilde{y}$ on utilise :

$$\begin{cases} \tilde{x} = \rho(lat - lat_{ponton})cos(lon) \\ \tilde{y} = \rho(lon - lon_{ponton}) \end{cases}$$

On a alors

$$\psi_d = arctan2(\tilde{x}, \tilde{y})$$

4.2 Aller-retour à la bouée

Le bateau n'avait aucun soucis à rejoindre la bouée depuis le ponton, cependant, lors de son arrivée à la bouée le bateau partait systématiquement vers le nord et on l'arrêtait avant qu'il ne franchisse la ligne de bouées interdite sous le pont suspendu. Et nous sommes restés bloqué un jour là dessus. En réalité, le bateau ne pointait pas vers le nord mais tournait simplement très lentement et avait un rayon de courbure énorme. Il n'arrivait pas à faire un 180° avant les bouées. La solution a été d'augmenter l'écart entre la consigne du moteur gauche et du moteur droit lors d'un virage. On a aussi mis un contrôleur de vitesse qui la réduit à l'approche de l'objectif. Après ces modifications, on a obtenu ces fichiers de log figure 3 :

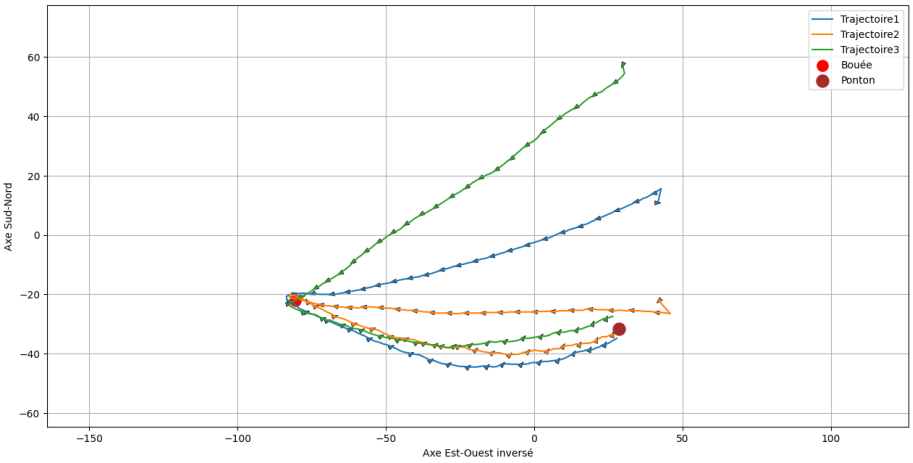


FIGURE 3 – Aller-retour à la bouée

Vidéo de notre bateau faisant l'aller-retour à la bouée : DD-Boat faisant l'aller-retour à la bouée

5 Proie-prédateur

5.1 Théorie

Une proie avance sur un cercle de centre C et de rayon R_1. Les coordonnées de la proie sont :

$$\mathbf{p} = \begin{bmatrix} x_C + R_1 cos(\omega_1(t - t_0 + t_{off})) \\ y_C + R_1 sin(\omega_1(t - t_0 + t_{off})) \end{bmatrix}$$

L'objectif des n prédateurs est d'encercler la proie et de tourner autour. Le i-ème prédateur suit alors, $\forall i \in [\![1, n]\!]$, le point :

$$\mathbf{p_i} = \begin{bmatrix} p[1] + R_2 cos(\omega_2(t - t_0 + t_{off}) + \frac{2\pi}{N}) \\ p[2] + R_2 sin(\omega_2(t - t_0 + t_{off}) + \frac{2\pi}{N}) \end{bmatrix}$$

5.2 Mise en pratique

Nous avons à l'aide de la théorie codé la trajectoire que notre DD-boat devait suivre afin de réaliser la chorégraphie. Pour cela, comme lorsque nous allions à la bouée nous récupérions en temps réelle les données du magnétomètre et de l'accélé- romètre, ce qui nous permet de calculer en temps réel le cap de notre bateau. Nous avons, également en temps réel, la position P_{sat} qui nous donne les coordonnées vers lesquels notre bateau doit aller. Nous utilisons donc la même méthode que précédemment afin de trouver le cap désiré. Nous avons cependant rencontré un problème. En effet, dû probablement à la défaillance de notre contrôleur qui avait du mal à gérer le bateau efficacement à l'approche de la cible.

6 Conclusion

Au terme de ce projet Guerlédan, nous avons atteint l'objectif principal : permettre au DD-boat de réaliser de manière autonome un aller-retour jusqu'à la bouée. Pour y parvenir, nous avons mis en œuvre plusieurs étapes essentielles, notamment la calibration de l'IMU, la détermination précise du cap et la mise en place d'un contrôleur permettant d'ajuster la vitesse angulaire du bateau en fonction de la direction souhaitée. Les ajustements effectués sur la commande moteur, en particulier l'augmentation du différentiel de vitesse entre les hélices lors des virages et la réduction automatique de la vitesse à l'approche de la cible, ont joué un rôle déterminant dans la réussite de la mission.

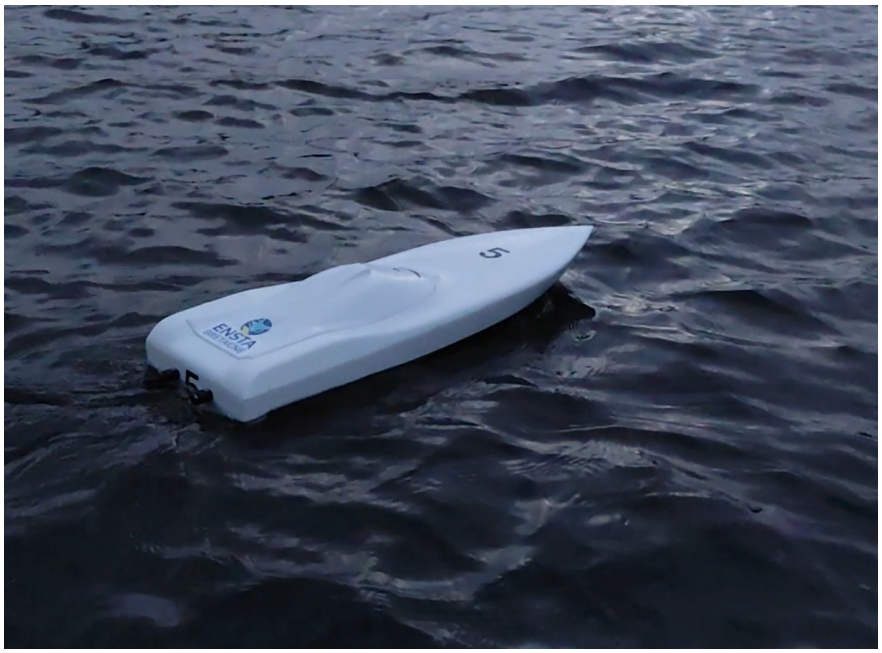
Le défi supplémentaire de la chorégraphie proie-prédateur n'a toutefois pas pu être mené à bien. Les difficultés rencontrées proviennent principalement des limites de notre contrôleur, qui éprouvait des difficultés à stabiliser efficacement le bateau lors de l'approche des points cibles successifs. Néanmoins, le travail effectué a permis de mettre en évidence des pistes d'amélioration, tant sur la partie commande que sur l'intégration des mesures issues des capteurs.

Ce projet nous a permis d'acquérir une compréhension approfondie des problématiques liées à la robotique mobile en environnement réel : incertitudes de mesures, contraintes mécaniques, limitations du modèle de commande et nécessité de validations expérimentales. Les résultats obtenus constituent une base solide pour des évolutions futures, qu'il s'agisse d'un contrôleur plus robuste, d'une meilleure gestion des capteurs ou de trajectoires plus complexes.

Compte Rendu

Année universitaire 2025-2026
Semestre 3

Guerlédan DD-BOAT 5



Groupe :
BIECHE Matys Adéas, RAMIS Lancelot

Encadrant :
M.JAULIN Luc

Ensta Campus de Brest

Engagement de non plagiat

Je soussigné, BIECHE Matys Adéas,

Je soussigné, RAMIS Lancelot,

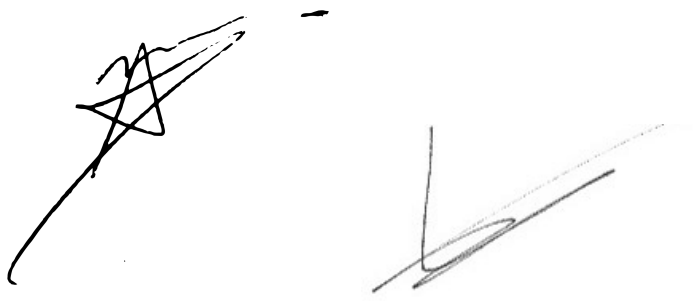
Déclare avoir pris connaissance de la charte de l'école et notamment du paragraphe spécifique au plagiat.

Je suis pleinement conscient(e) que la copie intégrale sans citation ni référence de documents ou d'une partie de document publiés sous quelques formes que ce soit (ouvrages, publications, rapports d'étudiant, internet etc...) est un plagiat et constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.

En conséquence, je m'engage à citer toutes les sources que j'ai utilisées pour produire et écrire ce document.

Fait le 08 novembre 2025

Signatures



Résumé

Dans le cadre de la seconde année d'école d'ingénieurs à l'ENSTA campus de Brest, nous avons travaillé sur la mise en oeuvre d'un bateau de surface robotisé, le DD-Boat, sur le lac de Guerlédan.

L'objectif principal est de réaliser chaque jour une mission (calibrage d'un capteur, implémentation du contrôle autonome...) avec/sur le DD-Boat afin de mieux appréhender la robotique et les concepts appris en cours.

Ce rapport a donc pour vocation de présenter nos travaux et nos choix d'implémentation, ainsi que les tests et résultats ayant pu être réalisés durant cette semaine.

De même, nous mettrons en lumière les compétences acquises, mais aussi les découvertes et recherches réalisées.

Sommaire

1 Introduction

Cette semaine à Guerlédan s'est articulée en deux missions principales pour notre DD-Boat :

— Effectuer un aller-retour de la rive à la bouée.
— Suivre une trajectoire programmée et synchronisée en essaim afin d'encercler une proie mobile de trajectoire connue à l'avance.

Afin d'effectuer ces missions, il était nécessaire de configurer le DD-Boat et notamment ses composants principaux :

— Sa centrale inertielle, et plus particulièrement son magnétomètre.
— Son GPS
— Ses moteurs

La calibration et le contrôle de notre bateau sont réalisés par le biais de scripts Python, transmis à la Raspberry Pi via SSH par le biais d'un réseau wifi accessible sur tout le site d'expérimentation.

Dans la suite, nous décrirons par jours les éléments de cours, ainsi que les tests et résultats qui nous ont permis d'effectuer divers objectifs sur toute la semaine, avec comme finalité la réalisation des deux missions décrites précédemment.

Il est à noter que le DDBoat-5 que nous avons manipulé n'utilise pas son GPS principal, mais le second, ce qui explique des différenciations dans l'initialisation des drivers par rapport aux autres équipes.

L'ensemble des scripts, une explication plus détaillée du code, ainsi que quelques vidéos sont disponibles via les liens suivants :

Gitlab ENSTA Bretagne : https://gitlab.ensta-bretagne.fr/biechema/DD-BOAT-5-Guerledan-1

Github : https://github.com/AdeMBCH/DD-BOAT-5-Guerledan-1

2 Jours 1

2.1 Objectif : Calibration / Autocalibration de l'IMU (magnétomètre)

L'objectif du jour était de supprimer les biais hard iron et les déformations soft iron du magnétomètre pour obtenir une norme de champ magnétique constante et un cap fiable. La démarche consiste à ajuster une ellipsoïde par moindres carrés sur les mesures brutes (x, y, z), puis de la transformer de manière affine en la ramenant à une sphère unité.

2.2 Rappels théoriques

2.2.1 Interférences Magnétiques

Il existe deux types d'interférences magnétiques :

Le **hard iron** est dû à la présence d'éléments ferromagnétiques et aux aimantations permanentes couplées au champ magnétique terrestre

Le **soft iron** est simplement dû à la manière dont se dirige le champ magnétique de la Terre en un point donné et comment il oriente les éléments non magnétisés.

2.2.2 Méthode par moindres carrés

Du fait de ces interférences, les mesures ne sont plus une sphère centrée mais une ellipsoïde décalée. Cette ellipsoïde peut être modélisée par une formule quadratique dont on cherche les coefficients par méthode des moindres carrées.

On cherche donc à minimiser :

$$J(Q, b) = \sum_{n=1}^{N} \Big((x_n - b)^\top Q (x_n - b) - 1 \Big)^2,$$

avec Q définie positive. On paramètre Q et b via le vecteur $p \in \mathbb{R}^{10}$ tel que

$$M(x)^\top p = 0, \quad M(x) = \big[x_1^2,\, x_2^2,\, x_3^2,\, x_1x_2,\, x_1x_3,\, x_2x_3,\, x_1,\, x_2,\, x_3,\, 1\big]^\top,$$

où

$$Q = \begin{pmatrix} p_1 & \frac{p_4}{2} & \frac{p_5}{2} \\ \frac{p_4}{2} & p_2 & \frac{p_6}{2} \\ \frac{p_5}{2} & \frac{p_6}{2} & p_3 \end{pmatrix}, \qquad b = -\tfrac{1}{2}\, Q^{-1} \begin{bmatrix} p_7 \\ p_8 \\ p_9 \end{bmatrix}, \qquad p_{10} = b^\top Q b - 1.$$

La calibration qui ramène le nuage sur la sphère unité se trouve en remarquant que :

En cherchant une transformation du type :

$$y = A^{-1}\,(x + b)$$

On a :

$$(x-b)^\top Q(x-b) = (x-b)^\top Q^{\frac{1}{2}} Q^{\frac{1}{2}}(x-b) = y^\top y = \|y\|^2 \approx 1.$$

D'où :

$$y = Q^{\frac{1}{2}}\,(x-b)$$

Cette transformation assure la calibration de nos données brutes du magnétomètre.

2.2.3 Angles d'euler

Afin de déterminer le cap du navire, on cherche à déterminer les angles d'euler :

Lorsque l'assiette devient non négligeable, le script `_euler.py` se sert de l'accéléromètre pour le repérage spatial.

Les données de l'accéléromètre sont dès lors acquises puis associées à celles du magnétomètre afin d'obtenir une matrice de rotation de laquelle on extrait les angles d'Euler, permettant d'orienter le DD-Boat dans l'espace :

$$a_1 = -\tfrac{1}{g}\; a_{mesur}, \qquad y_1 = \begin{bmatrix} x_{mag} \\ y_{mag} \\ z_{mag} \end{bmatrix}, \qquad R = \left[\tfrac{y_1 - (y_1^\top \cdot a_1)\cdot a_1}{||y_1 - (y_1^\top \cdot a_1)\cdot a_1||},\quad a \wedge \tfrac{y_1 - (y_1^\top \cdot a_1)\cdot a_1}{||y_1 - (y_1^\top \cdot a_1)\cdot a_1||},\quad a \right]$$

On récupère ainsi les angles d'euler :

$$\phi = arctan2(R_{3,2}, R_{3,3}), \quad \theta = -arcsin(R_{3,1}), \quad \psi = arctan2(R_{2,1}, R_{1,1}),$$

Il est à noter qu'un accéléromètre devient particulièrement imprécis/bruité lorsqu'il est utilisé dans un milieu propice aux vibrations en assiette. En revanche, l'usage du magnétomètre seul évite le filtrage des valeurs de l'accéléromètre si le robot reste à plat. Par conséquent, sous l'hypothèse d'un lac calme, la nécessité du filtrage est caduque.

On supposera donc qu'en tout temps, $a_1 = a_0$. La matrice R peut s'écrire :

$$R = \begin{pmatrix} \frac{x_{mag}}{\sqrt{x_{mag}^2 + y_{mag}^2}} & \frac{-y_{mag}}{\sqrt{x_{mag}^2 + y_{mag}^2}} & 0 \\ \frac{y_{mag}}{\sqrt{x_{mag}^2 + y_{mag}^2}} & \frac{x_{mag}}{\sqrt{x_{mag}^2 + y_{mag}^2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Ainsi on ne calculera plus les angles ϕ et θ mais seulement ψ donné par :

$$\psi = arctan2(y_{mag}, x_{mag})$$

2.3 Protocole d'acquisition

Le script `ReadIMU.py` permet d'acquérir les données :
— Environnement : éloignement des masses ferromagnétiques, moteurs coupés.
— Acquisition : magnétomètre, données bruts sur quelques minutes.
— Manœuvres : rotations du DDBoat afin de couvrir le maximum d'espace angulaire sur les trois axes.
— Export : fichier .txt avec trois colonnes $(x_{\text{mag}}, y_{\text{mag}}, z_{\text{mag}})$.

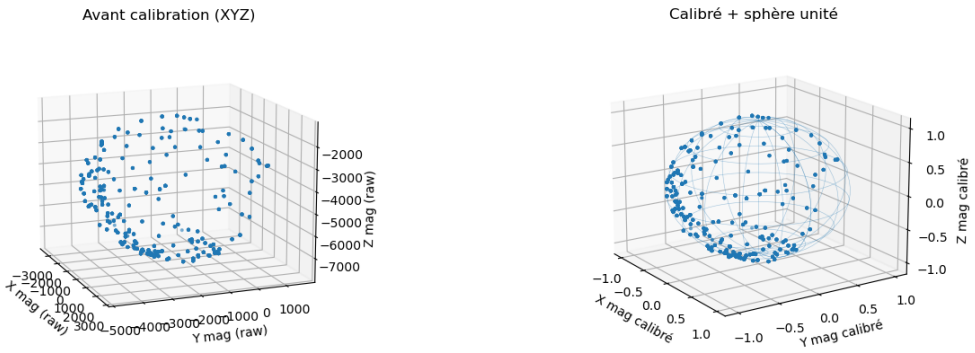
2.4 Algorithme d'autocalibration

Les données sont ensuite traitées dans notre script principal `Autonomous_capture.py`.

1. **Régression quadratique** par moindres carrés (SVD) pour estimer Q à partir des points (x_i, y_i, z_i).
2. **Restauration des paramètres géométriques** : Calcul du centre b, normalisation pour obtenir Q définie positive.
3. **Factorisation et calibration** point par point : $y = Q^{\frac{1}{2}}(x - b)$.
4. **Contrôles** : visualisation 3D avant/après via le script généré via ChatGPT pour le plot `plotIMUCalib.py`.

2.5 Validation

La figure figure 2.5 montre le résultat d'une calibration. On y remarque difficilement l'ellipsoïde excentrée (a), mais l'échelle permet de s'y convaincre. A droite (b), on obtient les points qui sont ajustés sur la sphère centrée.



(a) Nuage brut

(b) Nuage calibré + sphère unité

FIGURE 1 – Exemple de calibration magnétomètre par ajustement d'ellipsoïde et projection sur la sphère.

2.6 Remarques

Nous avons mis en évidence une erreur dans les calculs présentés par M.JAULIN dans la matinée. En effet, tels que présentés et une fois mis sous forme d'algorithme, la matrice Q n'était plus nécessairement définie positive. Après discussion avec d'autres groupes et différents tests, nous avons trouvé la méthode qui permettait d'obtenir le résultat. L'erreur a ensuite été transmise à M.JAULIN qui a effectué une correction le lendemain.

Sur cette première journée, nous avions aussi mis au point un algorithme de contrôle de cap afin d'aller à l'ouest pendant 15 secondes (Contrôleur avec un gain proportionnel sur l'erreur en cap). Nous ne le développerons pas dans ce rapport, puisqu'il a été modifié afin de coller avec les méthodes de contrôle présentées le lendemain. Malgré tout, nous avons pu aller vers l'ouest malgré un problème qui nous empêchait au départ d'y aller.

En effet, l'autocalibration nous a permis de mettre en évidence par la suite le fait que notre IMU n'était pas dans le bon sens. En pointant le nord, les points bruts obtenus se trouvaient du côté des x négatifs et des z positifs de l'ellipsoïde, ce qui ne correspondait pas au comportement attendu à Guerlédan avec un champ magnétique qui pointe à 64° dans le sol. Nous en avons conclus que nos axes x et z devaient êtres inversés. Puisqu'il nous était difficile physiquement de retourner l'IMU à cause du câble qui relie l'IMU à la Raspberry Pi, nous avons fais le choix d'ajouter à la main dans le driver de l'IMU des signes moins devant les données récoltées sur les axes x et z.

Nous avons donc réussi à réaliser toutes les missions demandées dans la journée.

3 Jour 2

3.1 Objectif : Régulation (optionnel), Feed Forward Control, GPS

L'objectif du jour était d'implémenter une étape de régulation (si la calibration n'était pas suffisante pour pointer dans la bonne direction avec le DDBoat), mais surtout un algorithme de Feed Forward Control qui nous permettrait de déplacer le robot via des coordonnées GPS. Il fallait être en mesure d'aller jusqu'à la bouée et de revenir à la base.

3.2 Théorie

Nous ne traiterons pas de la question de la régulation, puisque nous ne l'avons pas implémentée, car non nécessaire dans notre cas.

3.2.1 Modèle de commande FFC

Le feed forward control est un modèle de commande qui permet, en boucle ouverte, de mettre le DD Boat dans la bonne direction en amont du mouvement, ce qui, dans notre cas, nous évite une correction proportionnelle ou un PID en boucle fermée qui pourrait s'avérer capricieux. La figure 3.1 présente le principe de fonctionnement d'une FFC et comment elle est nourrie.

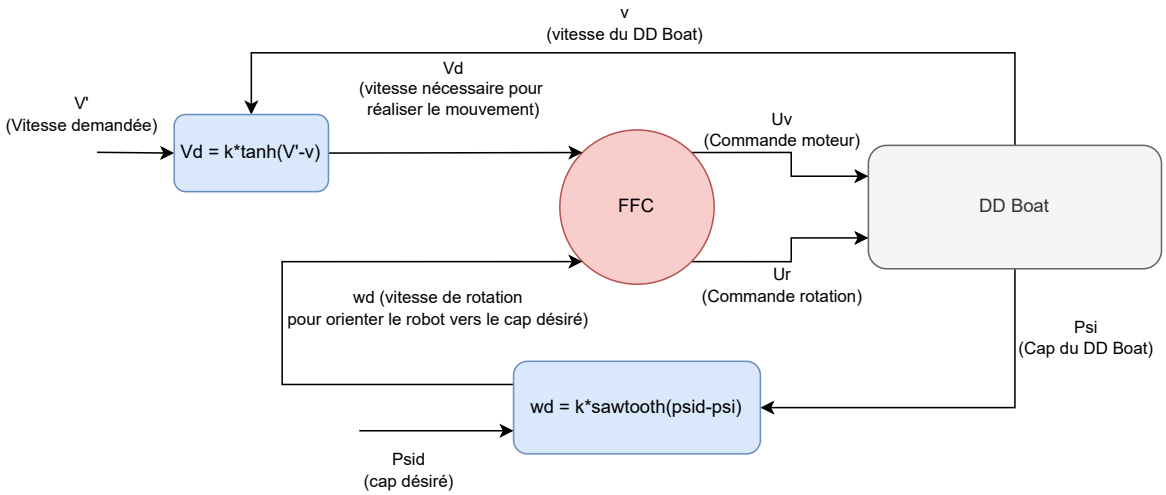


Figure 3.1 *Schéma du principe de Feed Forward Control*

On mélange une avance linéaire u_d et une vitesse de rotation souhaitée ω_d, corrigée par un gain K sur l'erreur de cap e_ψ. Les consignes moteurs gauche/droite sont :

$$e_\psi = \text{sawtooth}\pi(\psi - \psi_d),$$
$$\omega_{\text{cmd}} = \omega_d - k_{p\psi}\, e_\psi,$$

$$m_L = \mathrm{sat}_{\tanh}\big(\alpha_v u_d + \alpha_w \omega_{\mathrm{cmd}}\big),$$
$$m_R = \mathrm{sat}_{\tanh}\big(\alpha_v u_d - \alpha_w \omega_{\mathrm{cmd}}\big)$$

avec $\mathrm{sat}_{\tanh}(x) = m_{\max}\tanh(x/\lambda)$.

Dans notre cas nous avons choisis : $k_{p\psi} = 1{,}2$, $\alpha_v = 1{,}0$, $\alpha_w = 0{,}7$, $m_{\max} = 200$ à 230 selon essai, $\lambda = 3{,}0$.

3.3 GPS

3.3.1 Position GPS dans le repère local de Guerlédan

La trame NMEA *GLL* est convertie en degrés décimaux, puis en coordonnées locales (x, y) via une projection centrée sur les coordonnées GPS du ponton de Guerlédan notés l_x^m et l_y^m, créant ainsi un repère EST-NORD dans lequel peuvent être projetées n'importe quelles autres coordonnées GPS l_x et l_y grâce aux formules suivantes :

$$x = R\cos(l_y)(l_x - l_x^m)$$

$$y = R(l_y - l_y^m)$$

Nous avons choisi $R = 6378137$ m, ce qui n'est pas vrai (puisque nous devons en réalité prendre R égal au rayon projeté sur l'axe de rotation de la Terre), mais cette approximation suffit pour les distances parcourues sur le lac de Guerlédan.

3.3.2 Se diriger vers une position cible

Pour rejoindre une cible, on calcule le relèvement géographique ψ_d depuis le vecteur (x, y) exprimé en repère local Est-Nord :

$$\psi_d = \left(90^\circ - \mathrm{atan2}(y, x)\,\frac{180}{\pi}\right) \bmod 360^\circ.$$

Cette convention place 0° au Nord et croît dans le sens horaire, cohérente avec le magnétomètre du DD-Boat.

3.4 Estimation du cap ψ.

On lit le magnétomètre brut, et applique la calibration présentée dans la partie précédente.

3.5 Algorithmes mis en œuvre

Aller à un point GPS avec (`goto_point`) :

1. Lire la position (x, y). Si indisponible, conserver la dernière valide.

2. Calculer $\Delta x = x_g - x$, $\Delta y = y_g - y$ et la distance à la cible $d = \sqrt{\Delta x^2 + \Delta y^2}$.
3. Si $d \leq r$ (typiquement $r = 3\,\text{m}$), couper les moteurs, pause.
4. Calculer ψ_d depuis $(\Delta x, \Delta y)$ et appliquer les lois de commandes du moteur.

Cette stratégie suffit pour la navigation allant de la base jusqu'à la bouée et pour le retour.

3.6 Validation

Le test de la bouée a été effectué avec une méthode de contrôle différente de la FFC au départ (le fameux contrôleur PID). Nous avons validé par la suite son fonctionnement avec la FFC proposée. Le log suivant provient donc du trajet réalisé avec le contrôleur PID et non de la FFC. Du fait de notre sollicitation les jours suivants par les doctorants, nous n'avons pas repris de logs GPS du DD-Boat allant jusqu'à la bouée et revenant à la base avec le nouveau modèle de commande bien plus efficace.

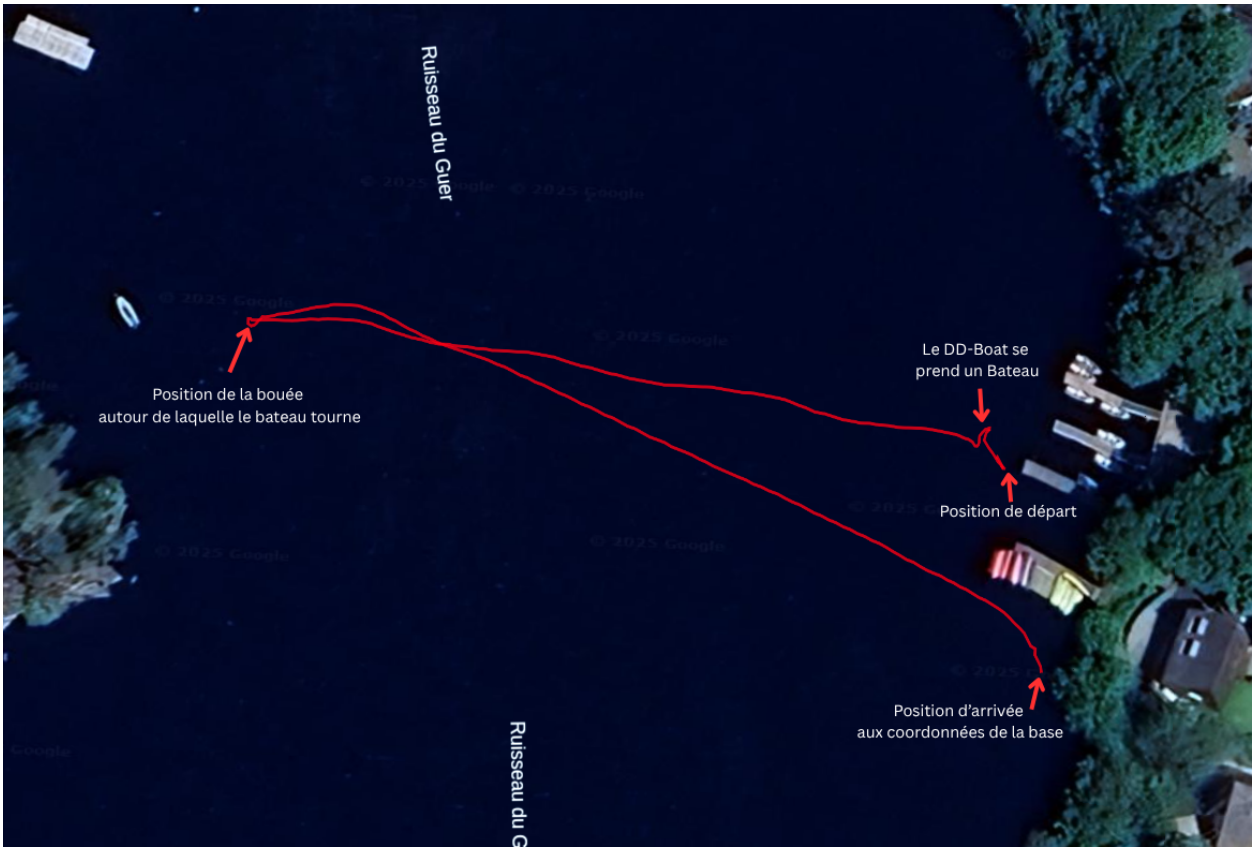


Figure 3.2 *Log GPS du parcours vers la bouée puis retour à la base*

La figure 3.2 montre le trajet réalisé par le DD-Boat vers la bouée puis son retour au ponton après une pause de quelques secondes.

3.7 Remarques

Comme dit précédemment, nous avons réalisé la mission dans les temps, mais en implémentant un contrôleur PID au départ. Elle a aussi été validée par la suite avec le FFC.

Nous avons éprouvé un certain nombre de difficultés à aller sur la bouée et surtout revenir sur la base à cause du PID qui n'était pas bien réglé. Nous avons donc changé les lois de commandes sur les moteurs, ajouté un "spirit" pour que notre DD-Boat aille droit et n'ait pas de déplacements courbés etc. Au final, à la fin de cette journée, nous avons fait le choix de reprendre de zéro et de tout refaire avec le Feed Forward Control.

Malgré tout, nous avons fait partie des groupes à avoir atteint la bouée et être revenus à la base au lancement de la mission à 11h30.

4 Jour 3 et 4

4.1 Objectif : Chorégraphie en essaim sur le lac de Guerlédan

L'objectif de ces deux jours a été l'implémentation d'un algorithme de suivi de trajectoire, afin de réaliser une chorégraphie de plusieurs DD-Boat sur le lac pour simuler la capture d'une "proie".

4.2 Calcul de la trajectoire

La chorégraphie est planifiée à l'avance par le biais de trajectoires définies par des paramètres précis. La recherche de ces paramètres sera décrite dans la prochaine section.

Concernant les trajectoires à suivre, elles correspondent à des épicycloides sur la proie qui avance suivant le cercle de centre connu $C = \begin{pmatrix} C_x \\ C_y \end{pmatrix}$, de rayon R_1 et de vitesse de rotation ω_1 et peuvent être décrites par des trajectoires circulaires dynamiques ayant pour centre les coordonnées de la proie à l'instant t, de rayon R_2 et de vitesse de rotation ω_2 de telle sorte que :

$$R_2 = (R_{max} - R_{min}) \times e^{(-\frac{t}{tau})} + R_{min}$$

La trajectoire de la proie est donc :

$$proie(t) = C + R_1 \times \begin{pmatrix} cos(\omega t - t_0 + t_{offset}) \\ sin(\omega t - t_0 + t_{offset}) \end{pmatrix}$$

Celle à suivre par le DD-Boat est :

$$predateur(t) = proie(t) + R_2(t) \times \begin{pmatrix} cos(\omega t - t_0) \\ sin(\omega t - t_0) \end{pmatrix}$$

Avec t_{offset} le décalage temporel nécessaire à la position de départ pour ne pas que les DD-Boat débutent la chorégraphie sous une passerelle considérée dangereuse.

4.3 Recherche des paramètres

Dans un premier temps, les paramètres ont été déterminés de manière relativement floue.

Pour la trajectoire épicycloïdale, il faut déterminer les rayons R_1 et R_2 et les pulsations ω_1 et ω_2 ainsi le mouvement décrit ci-dessus peut être correctement défini.

Les rayons sont d'abord définis arbitrairement selon l'espace accessible sur le site.

Les pulsations étaient initialement définies par essai-erreur. En effet une vitesse de consigne trop grande et le robot ne peut plus suivre. Une vitesse trop faible et le résultat devient moins visuel (des oscillations autour de la trajectoire de proie plutôt que des boucles).

Nous avons alors proposé une simulation qui ne s'appuyait non plus sur les paramètres vitesses de rotations ω, mais des vitesses maximum pouvant être atteintes par les DD-Boats et la vitesse souhaitée de la proie. De manière empirique, nous avons trouvé une vitesse maximum atteignable aux alentours de $1.5\,m.s^{-1}$. L'intérêt est ici de s'assurer qu'on calculera toujours en fonction des limites physiques et plus en fonction d'un résultat attendu. On fournit également un affichage des trajectoires pour visualiser le rendu et constater si le résultat correspond esthétiquement à la volonté initiale

Les vitesses de rotations se retrouvent alors via les formules classiques $\omega = \frac{v}{R}$. ω est donc choisi en connaissant les rayons des trajectoires, la vitesse max des DDBoats.

Le temps d'expérimentation est un paramètre important. Ici, il est choisi pour que la proie, allant à une vitesse donnée (ici $0.5m.s^{-1}$) puisse parcourir une distance donnée sans s'échouer tout en permettant au chasseur d'effectuer quelques révolutions.

La constante de temps qui permet d'établir la vitesse de rapprochement des chasseurs sur la proie est aussi un paramètre important. L'exponentielle $e^{\frac{-t}{\tau}}$ atteint 95% de sa valeur finale à $t = 3\tau$. Ainsi, connaissant la durée d'expérience, tau est choisi de telle sorte que $\tau < \frac{t_{exp}}{3}$ pour une évolution progressive vers la cible.

Au final le code et les démonstrations ont été proposés aux encadrants et ont permis de décider des paramètres suivants

$$R1 = 240m\,; \quad \omega_1 = \tfrac{1}{2R1}\,; \quad R_2 = 10 \times e^{-\frac{t}{200}} + 5\,; \quad \omega_2 = \tfrac{1}{3R_2(t)}$$

Cela donnera des vitesses maximales à $0.88\,m.s^{-1}$ et une vitesse de proie à $0.5\,m.s^{-1}$.

4.4 Synchronisation

La réalisation d'une chorégraphie organisée et synchronisée de l'essaim nécessite obligatoirement un démarrage simultané. Pour ce faire, deux options s'offrent à nous : utiliser l'heure fournis par le GPS ou bien l'horloge interne de la Raspberry Pi.

Compte tenu du délai et de la qualité de réception des gps, nous avont optés pour l'utilisation de `strftime('%X')` de la bibliothèque `time` pour avoir l'heure locale au format `"hh:mm:ss"` dans un `string` python. Cela permet via le code `wait.py` de patienter en affichant le temps restant et une fois l'heure voulue atteinte, de prendre comme référence `t0 = time.time()`.

L'arrêt de l'internet le 4ème jours a malheureusement démontré une limite de cette méthode. En effet, sans accès internet, l'heure locale n'est plus accessible. Bien que nous n'ayons pas eu le temps de l'implémenter, il faudrait pouvoir obtenir une mesure du temps effective par GPS lorsque l'accès à internet est rompu.

4.5 Implémentation

Le suivi de la trajectoire est possible grâce aux fonctions de suivis de points GPS qui s'actualisent dynamiquement, implémentées le jours 3.

4.6 Résultats

En récupérant l'évolution de la position GPS, on peut tracer sur Google Earth la trajectoire suivie lors des expériences.

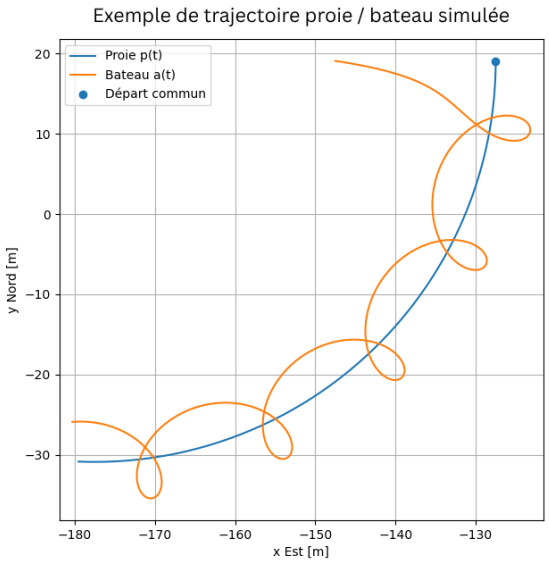


Figure 4.1 *Trajectoire simulée pour certains paramètres*

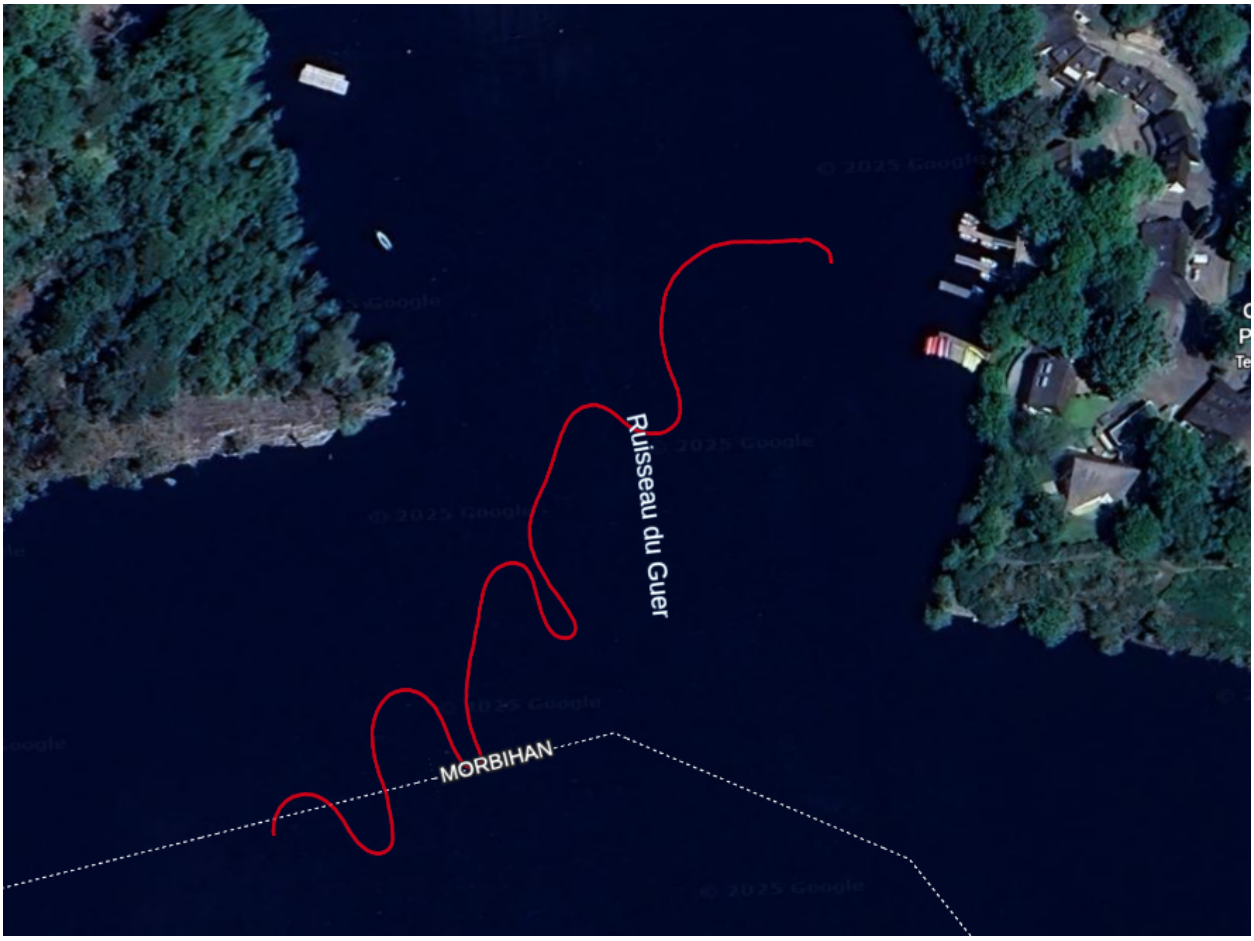


Figure 4.2 *Log GPS de la trajectoire réalisée par le DD-Boat avec les mauvais paramètres.*

Dans ce cas, la simulation nous donnait des boucles, mais dans la réalité, le DD-boat, ne parvient à réaliser les boucles qu'une fois son retard récupéré. Ainsi, 4.2 démontre que si la vitesse de consigne est trop importante, le robot ne fait que de tenter de suivre la trajectoire sans jamais réellement l'atteindre.

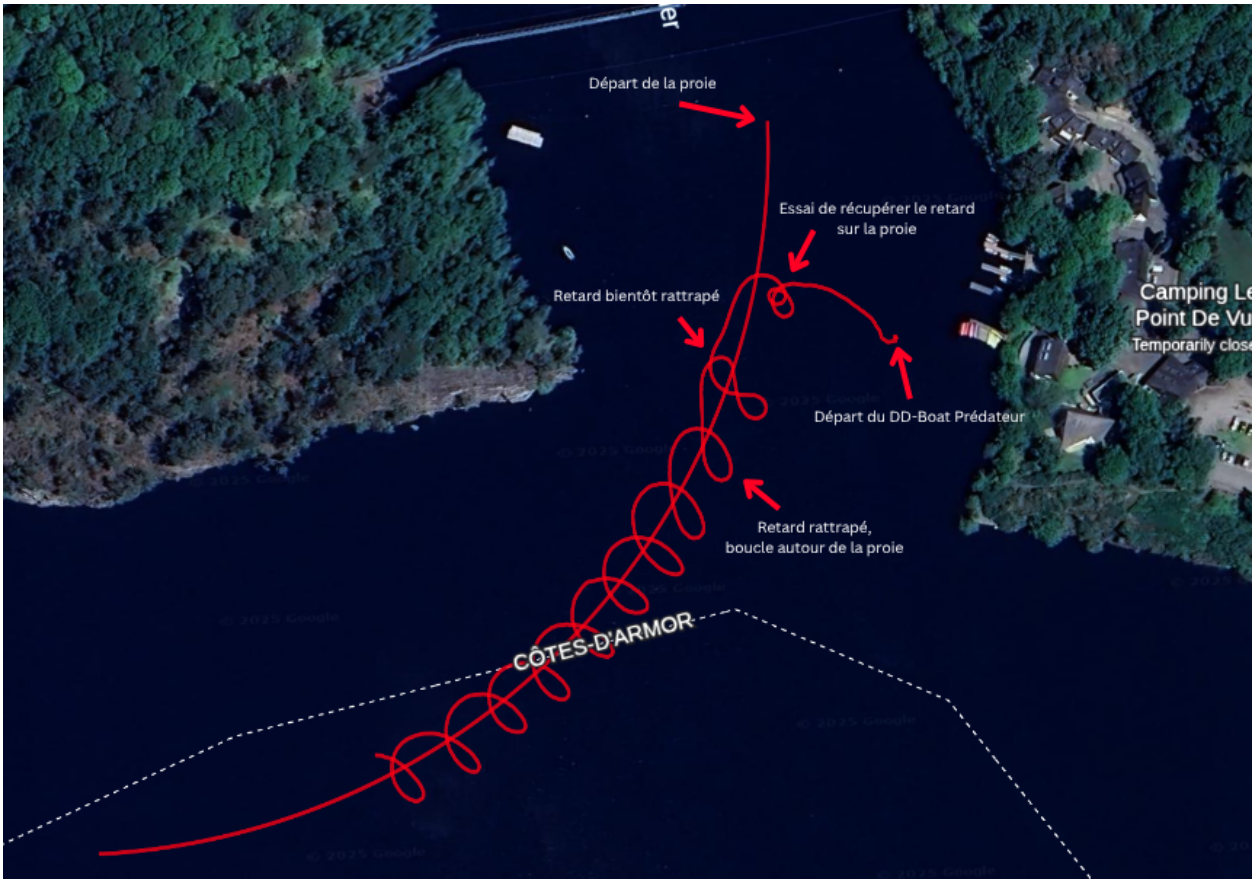


Figure 4.3 *Log GPS de la trajectoire réalisée par le DD-Boat avec les bons paramètres surperposé avec la trajectoire de la proie*

Nous pouvons voir sur la figure 4.3 que notre DD-Boat effectue la trajectoire simulée une fois les bons paramètres remplis et après une phase d'approche.

5 Jour 5

Ce jour est initialement prévu à la demonstration finale et la mise en commun de cette semaine de travail. Cependant personne n'avait préparé un DDBoat représentant la proie. Etant donné notre implication et la fiabilité de notre scirpt, nous avont étés chargés de lancer un programme qui permettait de reproduire le comportement de la proie.

5.1 Proie

Afin de reproduire le comportement de la proie, il suffit de suivre uniquement la trajectoire circulaire prédéfinie pour celle-ci. Cependant, ce changement demande une adaptation de certains paramètres. En effet, pour le suivi d'une trajectoire épicycloïdale, une vitesse

trop élevée entraine simplement des virages moins sérés et des oscillations moins visibles autour de la trajectoire. Cependant pour une trajectoire quasi rectiligne, un dépassement de l'objecif pousse le robot à tourner pour se replacer sur la trajectoire. limité par son rayon de courbure, le robot effectuera des grandes oscillation avec un dépassement systématique l'empéchant de retrouver la trajectoire voulue.

Le temps qui nous a été donné pour effectuer ces modifications étant bien trop courtes, la démonstration finale n'a pas pu aboutir.

5.2 Prédateur

Après un réglage de certains paramètres pour le comportement prédateur initialement prévu, nous obtenons un suivi efficace et une approche de la position désirée plus performante. La vitesse du DD-Boat est régulée à l'approche de la position cible afin d'éviter d'accumuler du retard ou au contraire de la dépasser, ce qui entraînerait des oscillations, ou une boucle non voulue sur la trajectoire.

Conclusion

En conclusion, cette semaine d'experimentation nous a fait toucher du doigt des aspects essentiels du métier d'ingénieur robotique et présenté la réalité du terrain.

En effet, l'approche théorique et simplifiée ou parfois surdéveloppée permet dans une certaine mesure de se préparer aux missions à accomplir. Cependant une perturbation, ou encore une non linéarité des moteurs (plage morte ou irrégulière sur des moteurs, IMU mal placé dans le bateau...) sortent du cadre pédagogique usuel et nous poussent à établir des solutions alternatives pour arriver au résultat attendu.

En clair, cette semaine aura été instructive et confirme d'autant plus notre intérêt pour la robotique.

ENSTA | INSTITUT POLYTECHNIQUE DE PARIS

ENSTA Campus de Brest

VA Robotique

DDboat Guerlédan - ROB 2A

Navigation Autonome d'un Bateau

Résumé du projet

Le projet consiste à coder un système de navigation autonome pour un bateau capable de se déplacer vers des coordonnées GPS précises. Le système utilise deux capteurs principaux : un GPS pour connaître la position actuelle et un compas magnétique pour maintenir le cap. Ce travail comprend la programmation des algorithmes de navigation, le contrôle des moteurs et la validation expérimentale du système embarqué.

Équipe de projet :

- Anatole LE BOUDEC–MOLENAAR
- Yoann MATARAZZI
- Hugo DORE

Novembre 2025

Table des matières

1 Introduction

1.1 Sujet

Comme dit dans le résumé, le sujet consiste en l'automatisation des déplacements d'un bateau afin de remplir différents objectifs.

1.2 Objectifs

Les objectifs, donnés au fil de la semaine, ont étés les suivants :

- Effectuer la calibration magnétique du DDboat afin de pouvoir obtenir le cap réel du bateau a partir de son cap magnétique
- Réussir a faire suivre au robot un cap fixé au préalable
- Réussir l'acquisition GPS de la position du DDboat afin de donner au bateau un point GPS "objectif" et lui faire aller à ce point
- Même idée que l'objectif précédent mais avec un point GPS "objectif" non fixe suivant une trajectoire dont l'équation était donnée

Le découpage du projet en différentes missions nous à permis de ne pas nous précipiter et de nous concentrer sur chaque objectif 1 à 1, chaque mission étant essentielle à la réalisation de la suivante.
Ce rapport est par ailleurs segmenté selon les différents objectifs.

2 Calibration magnétique

2.1 Théorie

L'objectif de cette première mission est d'effectuer la calibration de la boussole du DDBOAT. La boussole du bateau s'oriente à l'aide du champ magnétique terrestre, ce dernier variant selon la latitude et la longitude du bateau. Il est donc nécessaire de calibrer la boussole de temps en temps. La théorie veut qu'à l'aide d'un nuage de données non calibrées, on récupère les matrices de transformation pour pouvoir calibrer le bateau pour chaque nouvelle donnée.

On dispose d'un nuage de points $\mathbf{x} = (x_1, x_2, x_3)$ (non calibrés) nous permettant de récupérer f, l'équation de l'ellipsoïde formée par ces points où :

$$f(\mathbf{x}) = p_1x_1^2 + p_2x_2^2 + p_3x_3^2 + p_4x_1x_2 + p_5x_2x_3 + p_6x_3x_1 + p_7x_1 + p_8x_2 + p_9x_3 = 1 \quad (1)$$

On peut ensuite récupérer $\mathbf{Q}$ et $\mathbf{b}$ (les matrices de transformation) via les formules suivantes :

$$\mathbf{Q} = \begin{bmatrix} p_1 & \frac{1}{2}p_4 & \frac{1}{2}p_6 \\ \frac{1}{2}p_4 & p_2 & \frac{1}{2}p_5 \\ \frac{1}{2}p_6 & \frac{1}{2}p_5 & p_3 \end{bmatrix} \quad (2)$$

$$\mathbf{b} = -\frac{1}{2}\mathbf{Q}^{-1}\begin{bmatrix} p_7 \\ p_8 \\ p_9 \end{bmatrix} \quad (3)$$

On obtient ainsi les valeurs magnétiques calibrées via le calcul :

$$\mathbf{y} = \sqrt{\mathbf{Q}}(\mathbf{x} - \mathbf{b}) \quad (4)$$

La matrice de rotation $\mathbf{R}$ est définie par :

$$\mathbf{R} = \begin{bmatrix} \frac{\mathbf{y}_1 - (\mathbf{y}_1^\top \mathbf{a}_1)\mathbf{a}_1}{\dots} & \widehat{\mathbf{a}_1}\frac{\mathbf{y}_1 - (\mathbf{y}_1^\top \mathbf{a}_1)\mathbf{a}_1}{\dots}\mathbf{a}_1 \end{bmatrix} \quad (5)$$

où $\widehat{\mathbf{a}_1}$ désigne l'adjoint de $\mathbf{a}_1$ et $\mathbf{y}_1$, $\mathbf{a}_1$ sont respectivement une donnée calibrée et un vecteur permettant de prendre en compte l'accélération. Ce dernier est fixé à

$$\mathbf{a}_1 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \quad (6)$$

Les angles d'Euler sont extraits de la matrice $\mathbf{R}$ selon les formules suivantes :

$$\varphi = \arctan\left(\frac{R_{32}}{R_{33}}\right) \quad (7)$$

$$\theta = \arcsin(-R_{31}) \quad (8)$$

$$\psi = \arctan\left(\frac{R_{21}}{R_{11}}\right) \quad (9)$$

où R_{ij} désigne l'élément de la i-ème ligne et j-ème colonne de la matrice $\mathbf{R}$.

2.2 Pratique et résultats

Nous avons récupéré les données à l'aide d'une acquisition du magnétisme du bateau en le faisant décrire des rotations autour des 3 axes. Nous avons donc pu récupérer les matrices de transformation souhaitées et tracer la sphère suivante :

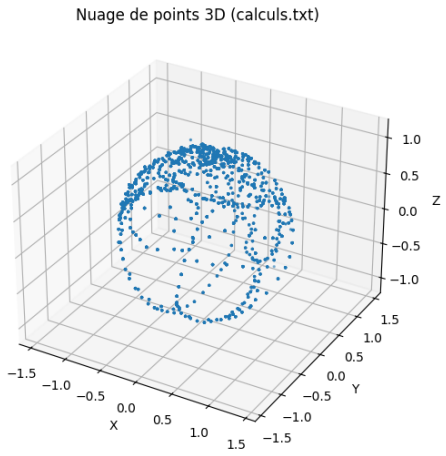


FIGURE 1 – Sphère des données magnétiques calibrées

Nous avons aussi pu récupérer la matrice **R** nous donnant la valeur des angles d'Euler et donc du cap du bateau en fonction des données magnétiques. Grâce à cela, nous avons pu effectuer notre premier test sur le lac pour lancer notre bateau cap Ouest pendant 15 secondes. Nous n'avions malheureusement pas récupéré de données lors de ce test. Ce premier objectif est la fondation des autres missions. Il permet au robot de suivre un cap magnétique donné. Dans la partie suivante, on reliera ce cap magnétique aux points cardinaux afin de permettre au DDBOAT de suivre un cap lié aux points cardinaux.

3 Suivi d'un cap

3.1 Théorie et mise en œuvre du contrôleur

Pour cette mission, l'objectif était de permettre à notre DDboat de maintenir un cap donné de manière autonome, afin de se déplacer en ligne droite vers une direction cible. Pour cela, nous avons implémenté en Python un contrôleur proportionnel (P) basé sur l'erreur d'orientation du bateau.

L'idée principale consiste à comparer en continu :

— le cap désiré ψ_d,

— et le cap mesuré par l'IMU ψ.

L'erreur $e = \psi_d - \psi$ est ensuite injectée dans une fonction proportionnelle, ici :

$$\omega_d = k \cdot \tanh(\text{sawtooth}(\psi_d - \psi)) \tag{10}$$

et en fonction de son signe on choisit de quel côté envoyer plus de puissance pour corriger l'erreur.

La fonction tanh permet de limiter la commande afin d'éviter des corrections trop brutales. Le signal ω_d correspond à la différence de vitesse à appliquer entre les deux moteurs : plus l'erreur est grande, plus le bateau doit accentuer la rotation pour corriger son cap.

Enfin, le module *feed-forward control* convertit cette différence de vitesse en deux commandes moteur cmd_L et cmd_R, tout en garantissant que la vitesse maximale autorisée n'est pas dépassée.

Voici l'implémentation Python du contrôleur proportionnel et du module *feed-forward* :

Listing 1 – Implémentation du contrôleur de cap

```
def regul_cap(psid, psi, k, imu):
    return k*np.tanh(sawtooth(psid-psi))


# Mise en forme commandes moteurs
def feed_forward(vd, wd, imu):
    k_moteur = 50
    v_moteur = k_moteur * vd  # vd est la vitesse desiree

    if v_moteur + abs(wd) >= 256:
        # condition qui permet de venir reduire la vitesse
        # pour eviter qu'on atteigne 256 ou plus
        v_moteur -= 50
        cmdL = v_moteur
        cmdR = v_moteur
    else:
        if wd > 0:
            cmdL = v_moteur + wd
            cmdR = v_moteur - wd
        if wd < 0:
            cmdL = v_moteur - (-wd)
```

```
            cmdR = v_moteur + (-wd)

    return cmdR, cmdL
```

3.2 Choix du gain et réglages sur le terrain

Au départ, nous avions choisi un gain $k = 20$, mais après plusieurs essais sur le lac de Guerlédan, nous avons constaté que cette valeur était trop faible pour compenser efficacement les petites dérives, notamment en présence d'un léger courant.

Après plusieurs ajustements, nous avons déterminé qu'un gain $k = 65$ offrait un compromis optimal :

— corrections suffisamment réactives,

— trajectoire plus stable,

— absence d'oscillations parasites.

La vitesse nominale du bateau a été fixée à $v_d = 4$.

3.3 Résultats et analyse

Une fois les erreurs de syntaxe et les détails d'implémentation corrigés, notre régulation s'est révélée fiable et performante. Dès le mardi matin, vers 11h45, notre bateau était capable de suivre un cap de manière stable et d'avancer vers l'ouest, même dans des conditions de vent ou de courant légers. Cette réussite a constitué une première étape importante et très motivante pour l'équipe. Après la pause de midi, nous avons donc pu passer sereinement à la partie suivante du projet : la navigation GPS

4 Suivi GPS d'un point fixe

4.1 Théorie

Pour réussir à rejoindre la bouée, on aura besoin dans cette partie de la FFC implémentée à la partie d'avant. Ce qui va changer dans cette partie est plus lié au GPS, et au fait de trouver un cap désiré avec uniquement des coordonnées GPS. L'acquisition GPS que nous avions grâce au bateau nous permettait d'obtenir la latitude et la longitude du bateau en Degrés et Minutes décimales (DMM), que nous devions ensuite convertir en Degrés Décimaux (DD), pour enfin se ramener en radian. Et ce afin de pouvoir appliquer la formule suivante pour trouver la position (x,y) de notre DDBOAT (on suppose que sur des petites distances comme au lac de Guerlédan notre problème est plan), dans le référentiel que nous avons choisi. Avec comme point d'origine le bord du ponton, l'axe des x étant orienté vers l'Est et l'axe des y étant orienté vers le Nord.

Les équations de conversion des coordonnées géographiques vers le repère local sont :

$$\vec{x} = \rho \cdot \cos(\text{lat}_p) \cdot (\text{lon} - \text{lon}_p) \tag{11}$$

$$\vec{y} = \rho \cdot (\text{lat} - \text{lat}_p) \tag{12}$$

$$\vec{z} = \rho - \rho_p \tag{13}$$

Avec cela on arrive ainsi à obtenir les coordonnées de la bouée notée (xb,yb) dans le repère lié au ponton. Pour trouver le cap désiré à chaque instant, en fonction des coordonnées de la bouée et des coordonnées (x,y) il faut utiliser arctan2().

4.2 Pratique

Notre régulation de cap marchant bien dès la matinée du deuxième jour, il ne nous restait "plus qu'à" implémenter la fonction `get_position()` et `define_dir()` qui renvoyaient respectivement (x, y) du bateau et le cap désiré.

Listing 2 – Fonction de localisation

```
def get_position(lon: float, lat: float, rho: float,
                 lon_p: float, lat_p: float, rho_p: float,
                 imu):
    x = rho * np.cos(lat_p) * (lon - lon_p)
    y = rho * (lat - lat_p)
    return x, y, 0.0
```

Listing 3 – Calcul du cap désiré (bateau → bouée)

```
def define_dir(xb, yb, x, y, imu):
    # vecteur bateau -> bouee
    dx = xb - x
    dy = yb - y
    # angle math: 0=Est
    a_math = np.arctan2(dy, dx)
    # convertir en boussole: 0=Nord
    a_compass = np.pi/2 - a_math
    return wrap_2pi(a_compass)
```

Nous avons passé plus d'un jour et demi à essayer de changer notre code, notre régulation de cap, refaire notre calibration, régler l'offset lié à l'IMU et plus encore. Le résultat était le même à chaque fois, le cap désiré n'était pas le bon.

Grâce à Antoine qui nous a conseillé d'afficher le cap désiré à chaque instant, et en essayant de placer des bouées fictives (en fixant des coordonnées pour lesquelles nous connaissions parfaitement le cap désiré, par exemple : $x_b = 10\,000$ et $y_b = 0$ était censé nous renvoyer un cap vers l'Est...), nous avons pu remonter à la source du problème et réaliser qu'il manquait des étapes de conversion d'angles.

Ces deux étapes sont celles surlignées en rouge dans le code : la première permet de retenir un angle cohérent avec l'orientation de notre repère. Quant à la deuxième, elle sert juste à se ramener à un cap entre 0 et 2π car nous avons choisi de prendre les angles comme cela avec notre régulation de cap.

4.3 Résultats et analyse

Après de longues heures à chercher le problème qui faisait échouer toutes nos tentatives, longues heures qui ont failli nous décourager, nous avons réussi à décortiquer notre code et à mettre en place certaines stratégies pour comprendre et régler ces problèmes.

Le problème a été résolu mercredi soir à minuit sur le ponton, la satisfaction et le bonheur ressenti à ce moment-là était si intense que les longues heures sans avancées furent oubliées instantanément. Après une nuit sans tracas nous avons pu tester notre programme de jour, et notre DDBOAT a enfin pu aller embrasser cette jolie bouée (Antoine en est témoin).

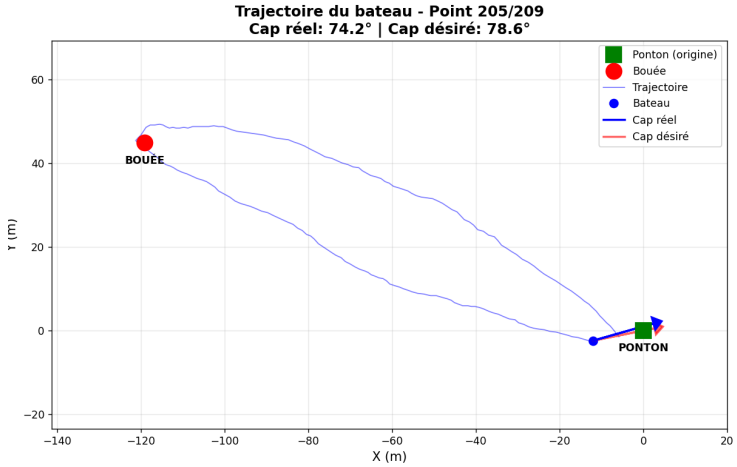


FIGURE 2 – Trajectoire du bateau lors du trajet ponton - bouée

5 Suivi GPS d'un point mobile

5.1 Mise en place et corrections apportées

Dès le jeudi soir, notre bateau était déjà capable de suivre la trajectoire demandée. Cependant, nous avons rapidement constaté que la première version de la fonction de trajectoire fournie n'était pas correcte : elle ne correspondait pas au mouvement attendu de la proie.

Les doctorants nous ont transmis une version corrigée le vendredi matin. Après l'avoir intégrée dans notre code, nous avons pu participer aux tests de suivi de proie en conditions réelles, synchronisés avec les autres groupes.

5.2 Résultats et problème de synchronisation

Le test final s'est avéré partiellement concluant. La trajectoire suivie par notre bateau était correcte dans sa forme : le DDboat restait proche de la fonction proie définie par notre programme (à moins de 4 mètres) et parvenait à tourner autour de la cible comme attendu.

Cependant, nous avons observé un problème de désynchronisation temporelle :

— la proie utilisait le temps système de l'ordinateur,

— tandis que notre bateau utilisait le temps fourni par son module GPS, plus lent et soumis à des décalages de mise à jour.

Cette différence de référentiel temporel a entraîné un léger décalage dans le suivi, expliquant pourquoi notre bateau suivait une proie différente et un peu plus rapide que la proie réelle.

De plus, sur les logs, on a du mal à se rendre compte si la trajectoire de l'objectif est réellement celle qui était attendue, mais en revanche notre bateau suit bien l'objectif.

Malgré cette contrainte, le comportement global restait satisfaisant : le DDboat suivait bien la direction instantanée de la cible et conservait une distance faible, ce qui validait notre stratégie de poursuite.

De plus, sur les logs, on a du mal à se rendre compte si la trajectoire de l'objectif est réellement celle qui était attendue. En effet, la "bouée" correspond au point objectif qui tourne autour de la proie, il est donc difficile à partir des logs de percevoir le mouvement cycloïdal dû à la rotation du point rouge autour de la proie car la trajectoire de la proie n'est pas affichée sur ce log. En revanche, notre bateau suit bien l'objectif.
Voici les logs que nous avons capturés le vendredi matin :

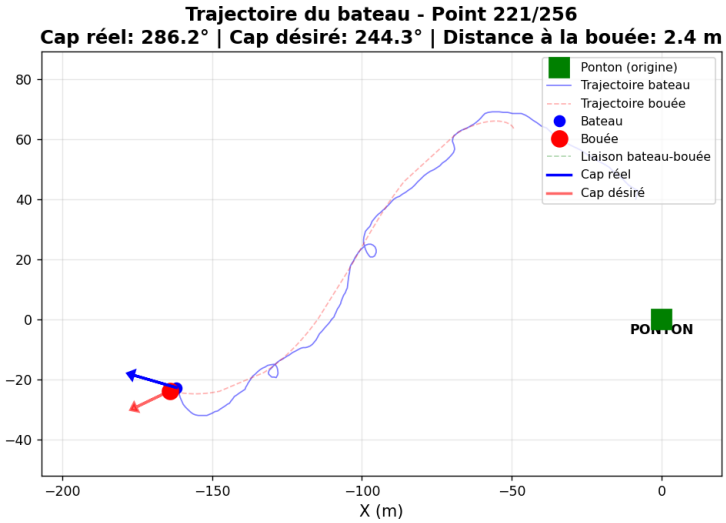


FIGURE 3 – Trajectoire du bateau lors du défi proie - prédateur

6 Conclusion

En conclusion, nous sommes tous les trois d'accord pour dire que cette semaine à Guerlédan était très intéressante sur plusieurs points.

D'abord, cette semaine nous a apporté une vision bien plus claire et concrète de ce que pourrait être un projet plus tard. Elle nous a aussi montré à quel point on ne sent pas les heures de travail passées lorsque nous sommes intéressés et impliqués.

Le fait de pouvoir observer en temps réel nos essais, nos erreurs et les corrections apportées a été un atout considérable. Cette expérience nous a montré que, sur des projets à plus grande échelle (comme ceux impliquant un paquebot par exemple) où l'on ne peut pas enchaîner une vingtaine de tests par jour, la mise au point devient bien plus longue et complexe. Grâce à nos tests rapides sur le DDboat, nous pouvions immédiatement identifier ce qui fonctionnait ou non et ajuster notre approche en conséquence.

Pour couronner le tout, le cadre et l'ambiance de cette semaine de travail étaient superbes, avec de beaux paysages, une météo clémente et des kayaks à disposition afin de découvrir ce lac.

A Code source principal

Lien GITLAB : https://gitlab.ensta-bretagne.fr/leboudan/megatron
Le code à exécuter est celui nommé "arduino_driver.py"

ENSTA Campus de Brest

VA Robotique

Rapport Guerlédan

Navigation Autonome d'un Bateau

Résumé du projet

Ce projet s'inscrit dans le cadre de l'unité d'enseignement de robotique et vise à développer les capacités d'un véhicule de surface autonome (DDboat) à naviguer, à maintenir un cap et à atteindre des points de passage (waypoints) définis par GPS. L'effort principal a porté sur la programmation des algorithmes de navigation et d'asservissement en Python, le contrôle des propulseurs en environnement réel via le protocole SSH, et la résolution des défis inhérents à l'intégration des systèmes embarqués (IMU et GPS) en milieu lacustre.

Équipe de projet :

- Alexandre PINARD
- Thomas MOSKOVTCHENKO

Novembre 2025

Table des matières

1 Introduction

1.1 Contexte du Projet

Dans le cadre de la deuxième année du cursus de robotique (ROB 2A), notre équipe a participé à une semaine intensive au bord du Lac de Guerlédan. Ce projet avait pour objectif principal de confronter la théorie du contrôle des systèmes embarqués à la réalité d'un environnement semi-marin. Le support technique était assuré par des véhicules de surface autonomes (DDboats) dont le contrôle moteur et la gestion des capteurs (IMU et GPS) étaient réalisés via une bibliothèque Python communiquant par protocole SSH. Cette mise en œuvre pratique visait à développer des compétences en algèbre linéaire appliquée, en asservissement et en résolution des problèmes complexes de navigation.

1.2 Objectifs

Les objectifs principaux, définis de manière incrémentale tout au long de la semaine, visaient à établir une chaîne de commande et de navigation complète. Ce rapport est structuré autour de ces missions successives, chacune étant essentielle à la réalisation de la suivante.

2 Objectifs de la semaine

2.1 Objectifs Techniques Spécifiques

1. **Calibration de l'Unité de Mesure Inertielle (IMU) :**
   Assurer la précision du capteur magnétique en compensant les effets *Hard Iron* et *Soft Iron*.
2. **Validation des Actionneurs :**
   Mettre en service et ajuster les propulseurs pour garantir un mouvement linéaire sans déviation pour une commande symétrique.
3. **Asservissement de Cap Fixe :**
   Mettre en place un algorithme de contrôle pour maintenir le DDboat sur un cap prédéfini (ex : Cap Ouest) pendant une durée de 15 secondes.
4. **Navigation sur Point de Passage (Waypoint) Fixe :**
   Développer un système de guidage basé sur les coordonnées GPS pour rejoindre une bouée stationnaire.
5. **Suivi de Cible Dynamique (La Chasse) :**
   Adapter l'algorithme de navigation pour suivre un autre DDboat dont les coordonnées GPS évoluent selon une trajectoire prédéfinie et dépendante du temps.

2.2 Objectifs Organisationnels et Personnels

Cette semaine a également servi de cadre pour développer des compétences transversales :

— Maîtriser le travail en équipe et la répartition des tâches complexes.

— Gérer le temps efficacement entre la phase de codage, les tests sur terre et les essais en milieu lacustre.

— Gagner en autonomie face à des problèmes d'intégration de systèmes embarqués.
— Progresser dans l'application pratique des algorithmes de contrôle et de navigation.

3 Description des Activités et Développements Techniques

3.1 Objectif 1 : Calibration de l'IMU

La première étape cruciale fut de calibrer le magnétomètre, composant essentiel de l'IMU, pour obtenir un cap fiable.

3.1.1 Principes de l'Interférence Magnétique

Deux types de perturbations ont été corrigées :

(a) **Hard Iron (Biais constant) :**
Représente l'ajout d'un champ magnétique constant au champ terrestre mesuré. Il est causé par des aimants permanents ou des composants métalliques magnétisés sur le bateau (moteurs, batteries). Mathématiquement, cela se traduit par un vecteur de biais constant, $\mathbf{b}$, qui décale le centre de la sphère de mesure vers un ellipsoïde.

(b) **Soft Iron (Distorsion multiplicative) :**
Correspond à la déformation du champ magnétique terrestre causée par la présence de matériaux ferromagnétiques non aimantés (comme certaines vis ou supports métalliques). Ce phénomène ne crée pas de champ propre mais modifie les lignes de champ, se traduisant par un étirement, une rotation ou une inclinaison de la sphère de mesure en un ellipsoïde. Il est corrigé par une matrice de compensation, $\mathbf{Q}$.

3.1.2 Algorithme de Compensation par Moindres Carrés

Nous avons utilisé la méthode de l'ajustement par moindres carrés (Least Squares Ellipsoid Fit) basée sur l'équation générale d'une ellipse. La correction vise à obtenir des données corrigées, $\mathbf{y}$, à partir des mesures brutes, $\mathbf{x}$, via la relation :

$$\mathbf{y} = \sqrt{\mathbf{Q}}(\mathbf{x} - \mathbf{b})$$

où $\sqrt{\mathbf{Q}}$ est la racine carrée de la matrice de compensation. Les résultats de cette compensation sont visualisés en annexe, figure 2.

3.1.3 Difficultés et Solutions

Initialement, la matrice $\mathbf{Q}$ calculée à partir de 1000 points de mesure présentait des **valeurs propres négatives**. Ceci indiquait que l'équation ne décrivait pas une ellipsoïde physiquement réalisable (mais une hyperboloïde), conduisant à des nombres complexes lors du calcul de la racine carrée matricielle $\sqrt{\mathbf{Q}}$. En réduisant l'échantillon à 500 points et en assurant une meilleure rotation du bateau durant la phase de collecte, nous avons pu obtenir une matrice $\mathbf{Q}$ définie positive. Ultérieurement, nous avons compris que la calibration devait idéalement se faire loin de toute masse métallique, ce qui a nécessité une légère modification des formules et un ajustement de l'environnement de test. L'opération a finalement abouti à une calibration du cap très satisfaisante.

3.2 Objectif 2 : Fonctionnement des Propulseurs

La mise en service des propulseurs a révélé une inversion de la commande entre le moteur droit et le moteur gauche par rapport à la convention logicielle. La principale difficulté technique fut d'assurer une trajectoire rectiligne pour une commande symétrique (ex : vitesse $V_G = 100, V_D = 100$). Le DDboat présentait une déviation systématique vers la gauche. Pour compenser ce biais mécanique (probablement dû à une asymétrie de poussée ou de traînée hydrodynamique), nous avons dû introduire un ajustement asymétrique des valeurs de vitesse au sein de la boucle d'asservissement.

3.3 Objectif 3 : Asservissement de Cap Fixe (Ouest)

Cette mission a permis de valider l'exactitude de la mesure du cap (ψ).

3.3.1 Calcul du Cap

Le cap est dérivé de la matrice de rotation d'attitude, $\mathbf{R}$, calculée à partir du vecteur magnétique corrigé ($\mathbf{y}$) et du vecteur de gravité $\mathbf{a}_0$. Le cap ψ est extrait par la formule trigonométrique :

$$\psi = \arctan 2(R_{21}, R_{11})$$

où R_{ij} sont les composantes de la matrice $\mathbf{R}$ reliant le repère du bateau au repère Nord-Est-Bas (NED).

3.3.2 Problème de l'Hypothèse de Gravité

Nous avons posé l'hypothèse simplificatrice $\mathbf{a}_0 = (0, 0, 1)^T$, supposant que le bateau restait parfaitement à plat. Cette hypothèse s'est avérée satisfaisante sur le lac (milieu très calme), mais a causé d'importantes confusions lors des tests sur terre. En effet, toute inclinaison du bateau au sol modifiait le cap mesuré, nous faisant chercher en vain un offset fixe et incohérent. Ce temps perdu a mis en évidence le fait que la matrice $\mathbf{R}$ nécessite théoriquement la compensation simultanée du tangage et du roulis (via un accéléromètre) pour fournir un cap fiable indépendamment de l'assiette du bateau. Finalement, les résultats obtenus sur l'eau furent très satisfaisants.

3.4 Objectif 4 : Navigation sur Waypoint Fixe

3.4.1 Problèmes Initiaux de Déviation

Lors des premiers essais pour rejoindre la bouée, le DDboat déviait fortement (jusqu'à 30°) vers l'ouest. Les tentatives d'ajout d'un offset dans l'asservissement de cap se sont révélées peu concluantes, l'erreur variant de manière erratique d'un essai à l'autre (de $-5°$ à $+40°$).

3.4.2 La Découverte de la Racine du Problème : Incohérence des Conventions

Nous avons soupçonné l'utilisation de la bibliothèque 'pyproj' d'introduire des erreurs, car nous n'avions pas accès aux définitions internes de ses fonctions de conversion. Nous avons alors décidé de revenir aux formules de cours pour le calcul des déplacements locaux **dx** et **dy** à partir des coordonnées GPS (latitudes et longitudes) du bateau et du point de référence.

Cependant, le problème fondamental résidait dans l'**incohérence entre les conventions angulaires du cap mesuré (ψ) et de l'angle GPS désiré (gisement)**. Les définitions du Nord, de l'Est et du sens de rotation (horaire vs anti-horaire) différaient entre les systèmes. Après avoir perdu du temps en vérifications compliquées sur le terrain (sensibilité à l'inclinaison, GPS imprécis à l'arrêt), nous avons opté pour une approche virtuelle et logicielle. En simulant des déplacements simples (Est, Nord) sur l'ordinateur, nous avons pu déduire empiriquement la conversion correcte :

$$\psi_{\text{correct}} = (-\psi_{\text{mesurée}} + 360°) \pmod{360}$$

Une fois cette conversion validée, le bateau a pu naviguer et revenir à la bouée avec une précision d'environ 3 mètres, ce qui est très convenable compte tenu de la précision nominale d'un GPS civil bon marché ($\approx$ 5 mètres). Les trajectoires sont visibles sur la figure 3 de l'annexe.

3.5 Objectif 5 : Suivi de Cible Dynamique (La Chasse)

L'objectif était de suivre un autre DDboat dont la trajectoire, définie par une formule mathématique, était connue à l'avance. Le principe théorique restait identique à l'objectif 4, mais les coordonnées de la cible (latitude et longitude) devenaient des fonctions du temps. L'implémentation de la dépendance temporelle représentait la seule difficulté majeure. Malgré l'intégration réussie du code Python prenant en compte les variations de coordonnées en fonction du temps, un seul essai a pu être réalisé et il n'a pas été concluant. Le manque de temps n'a pas permis le débogage.

4 Compétences Développées et Difficultés

4.1 Compétences Développées

Au cours de cette semaine à Guerlédan, nous avons acquis des compétences techniques, méthodologiques et organisationnelles qui nous permettent aujourd'hui de travailler de manière autonome sur un DDboat. Sur le plan technique, nous maîtrisons désormais l'utilisation du terminal Linux pour lancer et surveiller des scripts Python, ainsi que le fonctionnement des capteurs embarqués tels que l'IMU, le GPS et les propulseurs. Nous savons transformer des formulations théoriques en programmes opérationnels, en intégrant la calibration des capteurs, la gestion des vitesses et le suivi de trajectoire. D'un point de vue méthodologique, nous avons appris à structurer notre travail en équipe, à organiser les tâches et à collaborer avec d'autres groupes, en partageant connaissances et observations pour améliorer les performances de nos dispositifs.

4.2 Difficultés Rencontrées et Apprentissage

Les difficultés rencontrées lors de la semaine ont été nombreuses et variées, mais chacune a constitué un moteur d'apprentissage. L'une des principales sources de complexité était la superposition d'erreurs provenant de l'IMU, du GPS ou des conversions angulaires, rendant parfois l'interprétation des essais difficile. Nous avons compris qu'il était essentiel de vérifier chaque étape séparément avant d'effectuer un test complet sur l'eau, adoptant un raisonnement méthodique et cartésien pour isoler l'origine des problèmes. Nous avons

également découvert l'importance du contexte physique : des tests au sol, réalisés avec une mauvaise orientation ou une légère inclinaison du bateau, pouvaient produire des mesures erronées et induire des conclusions fausses. Ces expériences nous ont appris à prendre du recul, à faire des pauses pour revenir avec un regard neuf, et à ne pas persister aveuglément lorsqu'un problème semblait insoluble. Enfin, l'échange avec d'autres groupes s'est révélé indispensable pour identifier des erreurs communes et bénéficier de solutions complémentaires. Suite aux conseils précieux de Basile, Antoine, Romain et Damien, nous avons appris à décomposer les problèmes de manière systématique.

5 Résultats Obtenus

— **Suivi d'un Cap Fixe :** Validé avec succès.

— **Navigation sur Waypoint Fixe :** Validé avec une précision d'environ 3 mètres.

— **Suivi de Cible Dynamique (Chasse) :** En cours, non validé par manque de temps de débogage.

6 Conclusion

Au nom de notre équipe, cette semaine de projet a été une expérience d'apprentissage intensive et hautement formatrice. Elle nous a permis de concrétiser l'écart qui existe entre la théorie du contrôle et sa mise en œuvre pratique. Nous nous sommes rendu compte que des modèles théoriquement parfaits peuvent échouer en pratique à cause de facteurs environnementaux ou matériels non idéaux (interférences magnétiques, dérives hydrodynamiques, conventions GPS non standards). Le fait de devoir constamment remettre en question les formules, les résultats des capteurs et les hypothèses de base a été le principal moteur de notre progression. Nous avons particulièrement apprécié la polyvalence de l'approche, alternant entre le codage sur PC et les essais sur le terrain. Ce projet a non seulement confirmé notre attrait pour la robotique et les systèmes embarqués, mais nous a également préparés à affronter des défis d'intégration et de débogage complexes dans de futurs projets.

7 Annexes

7.1 Illustrations de la Calibration et de la Trajectoire

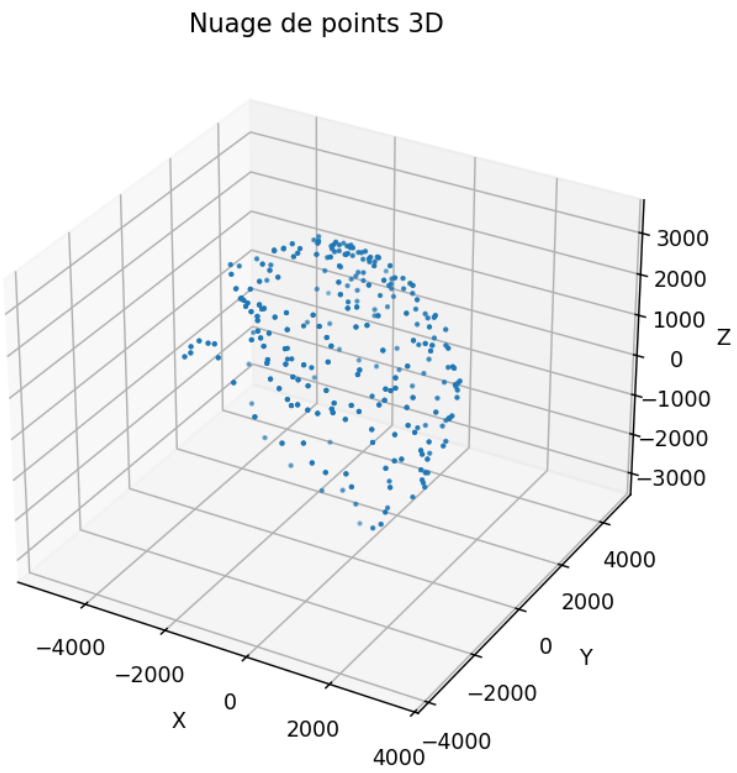


FIGURE 1 – Nuage de points magnétiques bruts (ellipsoïde déformée) avant compensation.

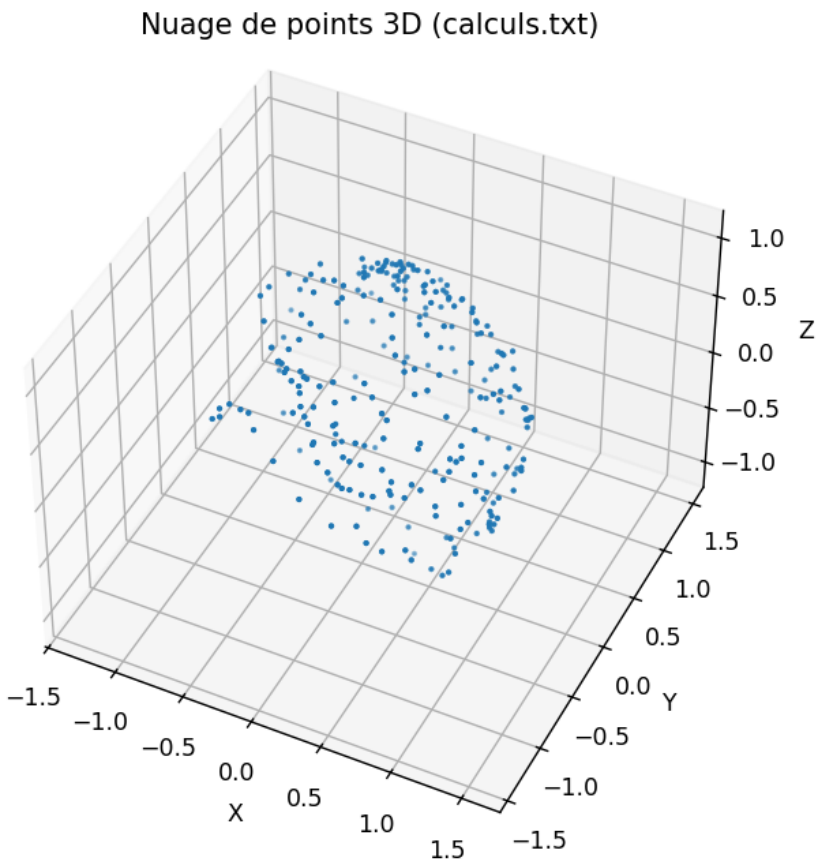


FIGURE 2 – Nuage de points magnétiques corrigés (sphère) après application des matrices **Q** et **b**.

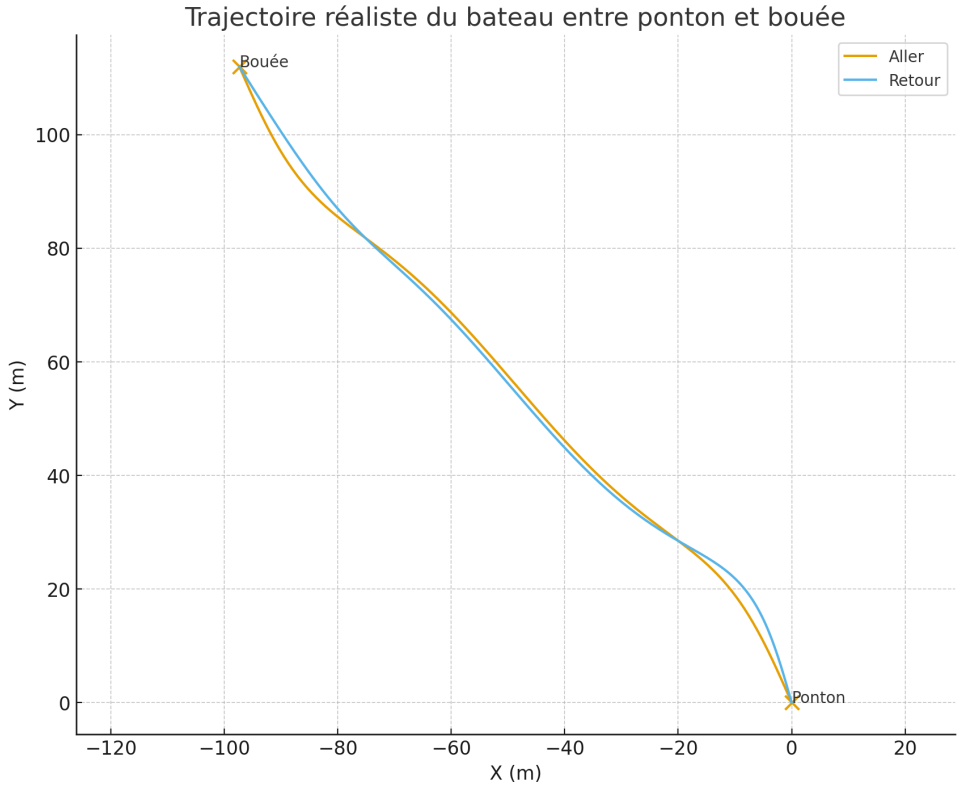


FIGURE 3 – Exemple de simulation de la trajectoire du DDboat avec dérive due au courant.

DDBoat (Guerlédan)

Kadijatou DIALLO - **kadijatou.diallo@ensta.fr**

Maël MABILOTTE - **mael.mabilotte@ensta.fr**

Tom VOISINE - **tom.voisine@ensta.fr**

FISE27

Novembre 2025

Figure 1: Image du Lac de Guerlédan

Table des Matières

1 Introduction

Ce document présente le travail réalisé dans le cadre de la semaine DDBoat à Guerlédan. Nous avons programmé et testé un bateau autonome à travers plusieurs missions visant à maîtriser sa calibration, son contrôle et sa navigation. Au fil des essais, nous avons affiné notre compréhension des capteurs, des algorithmes de commande et des contraintes du milieu. Ce rapport synthétise notre démarche et les résultats obtenus.
Notre bateau était le numéro 8 : MagicBoat.

2 Présentation général du projet

2.1 Structure des robots

Les DDBoats utilisés cette année sont des modèles de seconde génération, construits à l'ENSTA, anciennement ENSTA Bretagne. Ils conservent l'électronique des versions précédentes mais disposent d'une coque et d'une propulsion révisées. Chaque robot embarque une IMU, un récepteur GNSS et une radio LoRa, le tout alimenté par deux batteries 2S d'environ 8 V. Deux moteurs arrière assurent la propulsion, tandis qu'une télécommande radio permet de reprendre le contrôle en cas de problème. Les programmes sont transférés directement sur la Raspberry du bateau via la connexion Wi-Fi DARTAP établie spécifiquement pour le projet.

2.2 Communication avec le robot

La communication avec le bateau reposait principalement sur la connexion Wi-Fi déployée autour du lac. Grâce à ce réseau, nous pouvions ouvrir une session SSH sur la Raspberry embarquée afin de lancer nos programmes, suivre les capteurs en direct et analyser le comportement du robot pendant les essais. Les mises à jour du code étaient envoyées directement vers le robot ou effectuées dessus une fois connectés, ce qui facilitait les ajustements rapides lorsque nous devions tester une correction ou modifier un comportement juste avant un essai. Une fois les fichiers à jour, les missions étaient exécutées directement depuis le terminal du robot

2.3 Méthode de travail

Contrairement à d'autres groupes, nous n'avons pas utilisé Git pour structurer notre développement. Nous avons commencé la semaine en travaillant systématiquement à trois sur les mêmes fichiers afin de bien comprendre le fonctionnement du bateau et d'identifier ensemble les premières sources d'erreurs. Très vite, les imprévus techniques se sont multipliés : calibrations instables, comportements incohérents du bateau, bugs qui apparaissaient uniquement sur l'eau. Cela nous a poussés à adapter notre organisation au jour le jour. Dans la seconde moitié de la semaine, nous avons adopté une répartition beaucoup plus flexible. Pendant qu'un membre de l'équipe avançait sur le code de la mission suivante, les deux autres se concentraient sur la résolution des problèmes urgents, qu'il s'agisse de comprendre un bug logiciel, de vérifier la cohérence des mesures des capteurs ou de diagnostiquer un souci matériel. Cette manière de fonctionner nous a permis de progresser malgré les difficultés, en gardant un bon niveau de réactivité et en évitant que l'un de nous reste bloqué trop longtemps sur une tâche.

3 Les missions

3.1 Calibrage du DDBoat

Avant de commencer la réalisation des missions, il a été nécessaire de calibrer le magnétomètre trois axes dont est équipé le DDBoat. En effet, au sein du bateau, la présence de composants ferromagnétiques pertube les mesures du magnétomètre : on parle d'un effet Soft Iron. Sans correction, les

mesures du champ magnétique prennent la forme d'une ellipsoïde. L'objectif de la calibration est de corriger ces mesures pour qu'elles prennent la forme d'une sphère normée. L'équation de l'ellipsoïde est :

$$p_1x_1^2 + p_2x_2^2 + p_3x_3^2 + p_4x_1x_2 + p_5x_1x_3 + p_6x_2x_3 + p_7x_1 + p_8x_2 + p_9x_3 = 1 \tag{1}$$

On applique alors la méthode des moindres carrées sur une série de mesures de x_1, x_2, x_3 afin d'estimer les valeurs des paramètres $p_1, ..., p_9$.
Matriciellement, l'équation (1) peut s'écrire :

$$x^T \begin{pmatrix} p_1 & p_4/2 & p_5/2 \\ p_4/2 & p_2 & p_6/2 \\ p_5/2 & p_6/2 & p_3 \end{pmatrix} x + \begin{pmatrix} p_7 & p_8 & p_9 \end{pmatrix} x = 1 \tag{2}$$

avec

$$c = \begin{pmatrix} p_7 & p_8 & p_9 \end{pmatrix} \tag{3}$$

$$A = \begin{pmatrix} p_1 & p_4/2 & p_5/2 \\ p_4/2 & p_2 & p_6/2 \\ p_5/2 & p_6/2 & p_3 \end{pmatrix} \tag{4}$$

Le gradient de cette fonction s'annule en le centre b de l'ellipsoïde, ce qui donne : $2Ab + c = 0$ d'où $b = -\frac{1}{2} * A^{-1}c$.
Posons à présent : $y = x - b$. Il vient alors :

$$(y + b)^T A(y + b) + c^T(y + b) = 1 \tag{5}$$

On développe pour en déduire finalement :

$$y^T Ay = 1 + b^T Ab \tag{6}$$

Or, l'équation d'une ellipsoïde de centre b s'écrit également :

$$(x - b)^T Q(x - b) = 1 \tag{7}$$

Où Q est une matrice symétrique définie positive telle que : $Q = \frac{A}{1+b^T Ab}$
Afin de sphériser notre ellipsoïde, remarquons que :

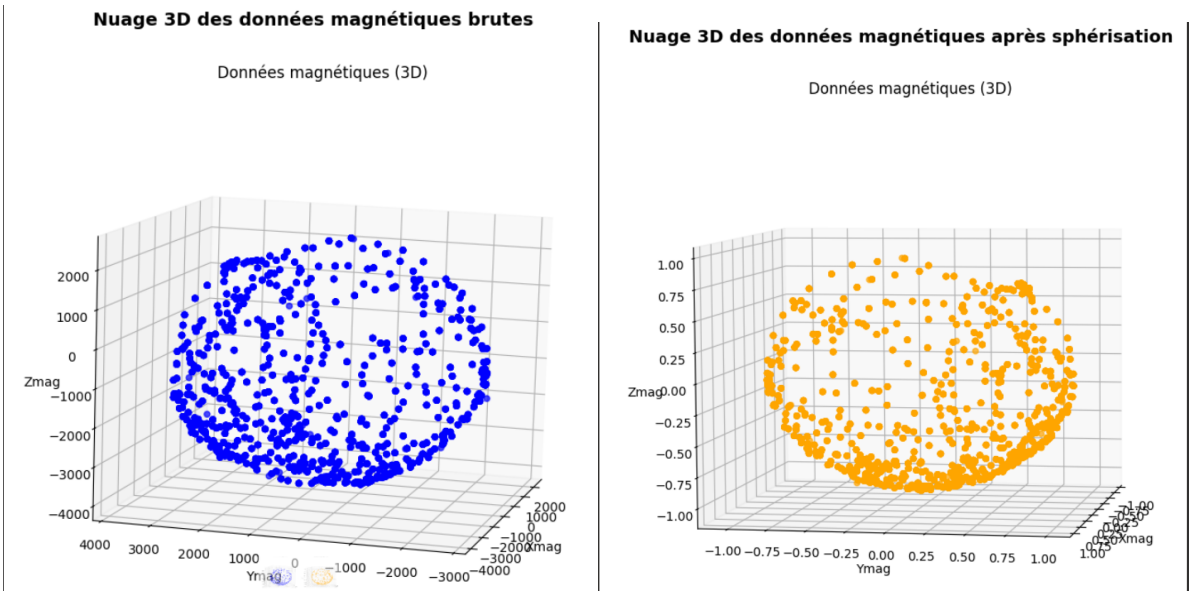
$$(x - b)^T Q(x - b) = 1 \iff (x - b)^T \sqrt{Q} \underbrace{\sqrt{Q}(x - b)}_{\text{y}} = 1 \tag{8}$$

On en déduit finalement la correction à apporter à nos mesures :

$$\underbrace{y}_{\text{mesure corrigée}} = \sqrt{Q}(\underbrace{x}_{\text{mesure non corrigée}} - b) \tag{9}$$

Certaines difficultées ont été rencontrées lors du premier jour de travail pour l'obtention de cette sphère. En raison d'une mauvaise définition de la matrice Q, nous nous sommes aperçus que celle-ci n'était pas définie positive et que par conséquent, sa racine carrée algébrique ne pouvait être définie. Les corrections nécessaires ont été apportées et le problème a pu être résolu.
Voici les données obtenues lors de la calibration :

Bien que les mesures du bateau aient désormais été corrigées, leur précision peut encore être améliorée



(a) Ellipse

(b) Sphère

Figure 2: Schémas de calibration du DDBoat

en utilisant la relation suivante : $z = y{*}R_{IMU}$ où z représente les mesures finales retenues et considérées comme précises. Initiallement on pose :

$$z = \begin{pmatrix} cos(I) & 0 & -sin(I) \\ 0 & -cos(I) & 0 \\ -sin(I) & sin(I) & cos(I) \end{pmatrix} \tag{10}$$

avec I = 64°

3.2 Suivi de cap (Ouest)

La première mission qu'il a fallu réaliser consistait à faire se déplacer le bateau vers l'ouest pendant une durée déterminée. Le DD-BOAT devait être capable de s'orienter selon un cap préalablement calculé, le suivre pendant un temps donné, et le réajuster si besoin (dérive, propulsion différentes des deux moteurs). Pour cela, nous devions exploiter les angles d'Euler. À l'aide des mesures normalisées du magnétomètre y_1 et de l'accéléromètre a_1, nous pouvons construire la matrice d'orientation du robot comme suit :

$$R = \left(\frac{y_1 - (y_1^T a_1)a_1)}{\|y_1 - (y_1^T a_1)a_1)\|}, \quad a_1 \wedge \frac{y_1 - (y_1^T a_1)a_1)}{\|y_1 - (y_1^T a_1)a_1)\|}, \quad a_1 \right) \tag{11}$$

De cette matrice nous pouvons extraire les 3 angles d'Euler associés à la pose du robot :

$$\begin{cases} \varphi = atan2(r_{32}, r_{33}) \\ \theta = -arcsin(r_{31}) \\ \psi = atan2(r_{21}, r_{11}) \end{cases}$$

Le contrôle du cap est réalisé par la fonction `control` présente dans le fichier `controler_way_point.py`. Celle-ci consiste en un contrôleur proportionnel semblable au Feed Forward Control. On communique

à la fonction l'erreur de cap, i.e la différence entre son cap actuel et son cap objectif. Ensuite, cette erreur est multipliée par un gain Kp. Afin que la correction à apporter au robot soit proportionnelle. Ensuite on vérifie que la valeur absolue de la quantité $Kp * erreur$ ne dépasse pas un seuil fixé, afin que la correction ne soit pas trop brutale même en cas de grosses erreurs. A la place de majorée la correction nous aurions pu utiliser la dérivé de l'erreur (ie implémenter un controleur PD) mais nous aurions eu un coefficient supplémentaire à paramétrer, ce qui était plus complexe et ne se nécessitait pas. Enfin, on adapte alors la vitesse des deux moteurs en ajoutant ou retranchant à leur vitesse nominale la correction calculée afin de faire tourner le bateau pour que son erreur de cap diminue. Dernièrement, on s'assure que les vitesses des moteurs ainsi calculées appartiennent bien à leur domaine de fonctionnement (dans notre cas : de -255 à 255).

Figure 3: Suivi du cap GPS via Google Earth

3.3 Navigation par points GPS

Afin que le DDBoat se rende à un Waypoint il faut définir un repère (O, x, y). Cependant seules les longitudes et latitudes nous sont renvoyés par le bateau. Pour convertir tout cela en point x, y, z nous avons les relations suivantes :

$$\begin{cases} \tilde{x} = \rho cos(l_y) * (l_x - l_x^{reference}) \\ \tilde{y} = \rho(l_y - l_y^{reference}) \\ \tilde{z} = \rho - \rho^{reference} \end{cases}$$

avec l_x la longitude et l_y la latitude renvoyées par le DDBoat. Le $\tilde{z}$ ne nous sera pas utile par la suite car nous avons fait l'hypothèse que nous travaillions dans un plan.

```python
def go_to_waypoint(self, lat_WP, lon_WP):
    # entrée : coordonnée du waypoint en degré, décimal de degré
    x_WP, y_WP = self.get_coords_in_pontoon_frame(np.radians(lon_WP), np.radians(lat_WP)) # coordonnée du WP dans le repère cartésien

    dist_to_waypoint = np.inf
    t = 0
    while dist_to_waypoint > 3:

        dist_to_waypoint = self.dist_to_waypoint(x_WP, y_WP)
        if t%10 == 0:
            target_heading = self.calcul_target_heading(x_WP, y_WP) # calcul du target heading toutes les 10 secondes au cas où le bateau a dérivé
        if t%2 == 0:
            self.save_log(dist_to_waypoint)             # enregistrement de la position du bateau et de la distance jusqu'au WP toutes les 2 secondes

        cap = self.calcul_cap_DDBOAT() # cap en degré entre 0 et 360
        error_cap = self.calcul_error_cap(cap, target_heading) # erreur de cap entre -180 et 180
        left_speed, right_speed = self.calcul_vitesses_motor(error_cap)
        self.ard.send_arduino_cmd_motor(left_speed,right_speed)

        print("cap : ", cap, "       target_heading : ", target_heading, "    error : ", error_cap)
        time.sleep(1)
        t += 1
    print("Waypoint atteint")
    self.ard.send_arduino_cmd_motor(0,0) # stop
```

Figure 4: Méthode `go_to_waypoint` de la classe `classe_boat`

La méthode juste au dessus sert à rejoindre un waypoint à l'aide de ses coordonnées GPS. La méthode `self.get_coords_in_pontoon_frame` renvoie les coordonnées du waypoints en $\tilde{x}$ et $\tilde{y}$ dans une repère où le ponton en est l'origine. Tant que le bateau n'est pas à 3 mètres du waypoint il s'en rapproche tout en ajustant son cap. Dans le cas où il y aurait plusieurs points fixes, alors une méthode `mission_DDBOAT` est implémentée afin que le DDBoat se déplace d'un point à un autre jusqu'à balayer l'entièrté des waypoints qui constituent la liste.

```python
def mission_DDBOAT(self):
    for waypoint in self.liste_waypoints:
        lat_WP, lon_WP = waypoint
        self.go_to_waypoint(lat_WP, lon_WP)
        time.sleep(10)
    # self.download_log()
```

Figure 5: Méthode `mission_DDBOAT` de la classe `classe_boat`

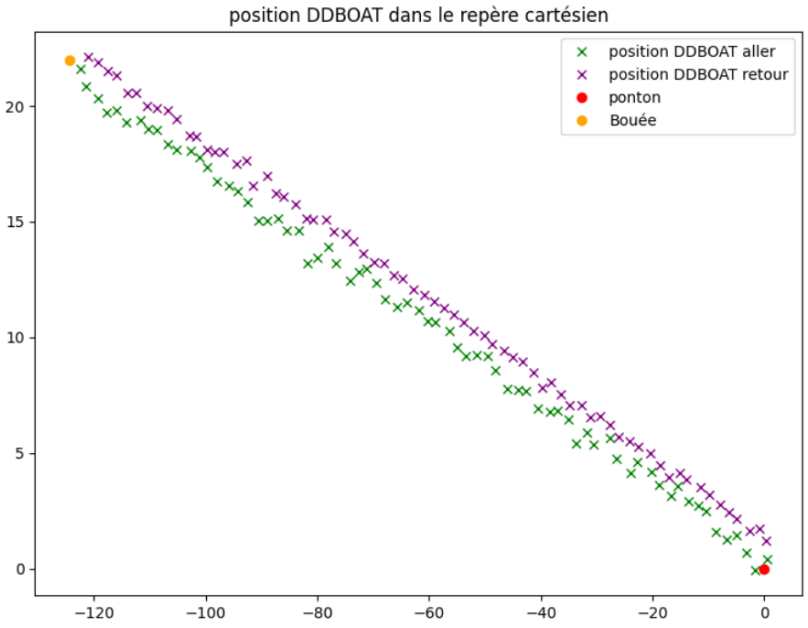


Figure 6: Aller-retour du MagicBoat vers le waypoint (x en abscisse, y en ordonnée)

Voici le trajet du MagicBoat pour l'aller-retour vers la bouée.

3.4 Suivie de la Proie

Le bateau est maintenant capable de rejoindre un waypoint fixe, qui durant la semaine était une bouée. En adptant un peu la consigne des waypoins à suivre, il est donc aussi capable n'importe quel objet dont on connait la position, y compris un objet mobile. La dernière mission était de capturer une proie qui suivait une trajectoire connue. La trajectoire était circulaire de centre le point C (voir schéma suivant).

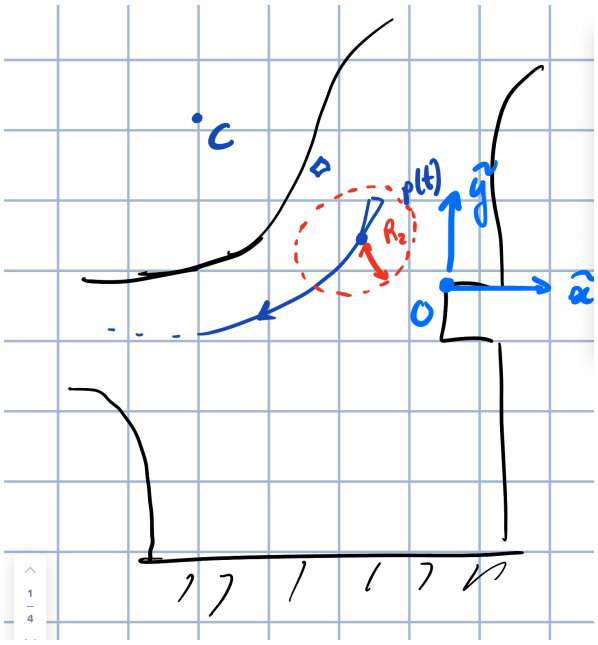


Figure 7: Schéma de cours sur la mission finale

$P(t)$ est la position de la proie et est utilisée dans la méthode $p_{sat}(t, i)$ afin que le bateau i parmi les N bateaux tourne autour de celui-ci durant sa navigation.

$$p(t) = \begin{pmatrix} x_c + R_1 cos(\omega(t - t_0 + t_{off}) \\ y_c + R_1 sin(\omega_1(t - t_0 + t_{off})) \end{pmatrix} \tag{12}$$

Avec $t_f - t_0 = 300$ i.e que la mission dure 300 secondes.

$$p_{sat}(t, i) = \begin{pmatrix} p_1(t) + R_2(t) cos(\omega_2(t - t_0 + t_o ff)) + i\frac{2\pi}{N} \\ p_2(t) + R_2(t) sin(\omega_2(t - t_0 + t_o ff)) + i\frac{2\pi}{N} \end{pmatrix} \tag{13}$$

Avec

$$R_2(t) = 50 exp(\frac{t - t_0}{1000}) + 10 \tag{14}$$

Et :

$$\begin{cases} T_c = 2000 \\ \omega - 1 = -\frac{2\pi}{T_c} \\ \omega_2 = \frac{2}{R_2} \\ t_0 = 16h30 \\ R_1 = 240m \end{cases}$$

Pour réaliser cette mission nous avons implementé une fonction de suivi de trajectoire qui reprend les méthodes précédentes de suivi de waypoint. Toutes les secondes la position de la proie est recalculé grace à l'équation de sa trajectoire et le robot l'utilise comme nouveau waypoint à rejoindre. Bien que cela semble efficace nous n'avons pas pu tester car nous avons eu des problèmes d'IMU et de charge des batteries le dernier jour.

4 Conclusion

Cette première semaine à Guerlédan nous a permis de prendre en main les DDBOAT et de mettre en place les briques essentielles pour nos futures missions. Nous sommes désormais capables de calibrer une IMU et de contrôler les bateaux dans des scénarios simples, comme le suivi de cap ou le suivi de cible.

Ce cours nous a beaucoup plu. Il nous a offert une première immersion concrète dans la robotique, et nous avons trouvé cela très enrichissant.

Rapport de projet DDBoat 9 : DDBoatthebest

CONAN Elodie, VERILHAC Oscar, LAFARGE Mahé

ENSTA Campus de Brest

3 au 7 novembre 2025 Guerlédan

Résumé

Ce rapport présente le travail effectué sur un DDBoat pendant la semaine à Guerlédan. A travers différentes mission nous avons développé un code python permettant de diriger le DDBoat grâce à sa centrale inertielle.

Table des matières

1 Introduction

Les codes que nous avons utilisé pendant cette semaine ainsi que les logs importants sont sur le GitHub suivant : GitHubDDBoatthebest.

Notre travail a porté sur le calibrage de la centrale inertielle (IMU), ensuite nous avons tenté de réaliser plusieurs missions avec le robot : le suivi de cap, la navigation vers un point GPS et enfin la mission de capture.

2 Calibrage de l'IMU

Avant de pouvoir effectuer des missions avec le DDBoat il a fallu calibrer la centrale inertielle.

2.1 Sphérisation et centralisation des données

Pour calibrer la centrale inertielle on aquiert d'abord le champ magnétique qu'elle perçoit en la tournant dans plusieurs directions. Ces données forment une ellipsoïde non centrée au lieu de la sphère centrée attendue.
En effet le magnétomètre est soumis à des distorsions de champ magnétique. Le hard iron est la distorsion lié à un aimant permanent, il décentralise la sphère des données. Le soft iron est dû au champ magnétique induit dans le fer à proximité de l'IMU. Cette distorsion déforme la sphère, elle devient donc une ellipsoïde. Pour pouvoir utiliser les données il nous faut sphériser ces points et centrer la sphère. Pour cela nous utilisons une méthode des moindres carrés. L'équation d'une ellipsoïde est :

$$p_1x_1^2 + p_2x_2^2 + p_3x_3^2 + p_4x_1x_2 + p_5x_1x_3 + p_6x_2x_3 + p_7x_1 + p_8x_2 + p_9x_3 = 1$$

En écriture matricielle on a :

$$MP = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

avec :

$$M = \begin{pmatrix} x_1^2(1) & x_2^2(1) & \cdots & x_2(1) & x_3(1) \\ x_1^2(2) & x_2^2(2) & \cdots & x_2(2) & x_3(2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^2(n) & x_2^2(n) & \cdots & x_2(n) & x_3(n) \end{pmatrix}$$

et P le vecteur des paramètres.
On calcule P grâce à la méthode des moindres carrés :

$$P = (M^T M)^{-1} M^T$$

L'équation d'une ellipsoïde de centre b peut aussi s'écrire sous la forme :

$$(x-b)^T Q(x-b) = 1$$

En identifiant les deux équations d'ellipsoïde, on obtient les valeurs des matrices Q et b :

$$Q = \begin{pmatrix} p_1 & \frac{1}{2}p_4 & \frac{1}{2}p_5 \\ \frac{1}{2}p_4 & p_2 & \frac{1}{2}p_6 \\ \frac{1}{2}p_5 & \frac{1}{2}p_6 & p_3 \end{pmatrix}$$

$$b = -\frac{1}{2}Q^{-1} \begin{pmatrix} p_7 \\ p_8 \\ p_9 \end{pmatrix}$$

Et on obtient les données corrigées grâce à la formule :

$$y = \sqrt{Q}(x-b)$$

Nous avons tourné le bateau selon toutes les directions loin de potentielles perturbations extérieures. Le module *acquire_mag_data.py* enregistre les données du magnétomètre dans un fichier texte. Ensuite nous avons affiché les données et effectué les calculs pour la calibration dans le module *plot_mag_3D.py*.

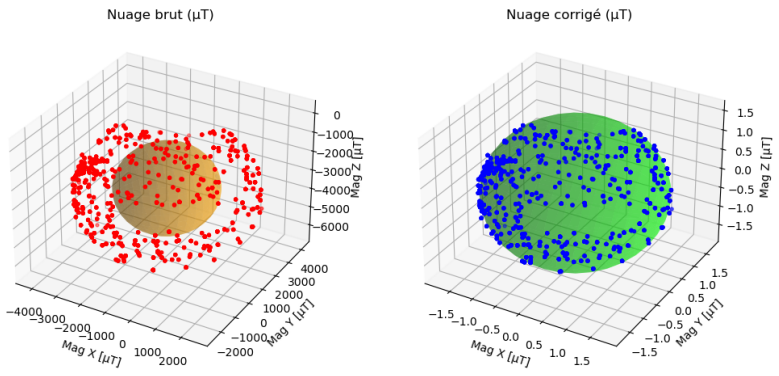


FIGURE 1 – À gauche les données du magnétomètre avant calibration, et à droite après calibration.

Nous en avons déduit notre matrice Q :

```
Q = np.array([[9.19344322e-08, 1.49580480e-08, -1.05545320e-08],
              [1.49580480e-08, 8.86431491e-08, 8.28949163e-09],
              [-1.05545320e-08, 8.28949163e-09, 1.40532829e-08]])
```

FIGURE 2 – Matrice de calibration Q

2.2 Calibrage du cap

Les données fournies par le magnétomètre peuvent être mal calibré, il nous faut donc les ajuster en comparant avec une référence connue. Nous sommes sur le lac de Guerlédan où le champ magnétique de la Terre rentre dans le sol avec un angle $I = 64$ avec celui-ci. Ainsi, nous pouvons calculer une matrice de référence contentant les coordonnées cartésiennes de différentes directions : Nord, Ouest, Haut. On obtient la matrice de référence suivante :

$$Z = \begin{pmatrix} \cos(I) & 0 & -\sin(I) \\ 0 & -\cos(I) & 1 \\ -\sin(I) & -\sin(I) & \cos(I) \end{pmatrix}$$

Ce trièdre direct va nous permettre de calibrer notre magnétomètre pour qu'il nous donne les mêmes valeurs lorsqu'il est pointé dans ces directions et permet donc de faire coïncider le repère du magnétomètre avec le repère magnétique théorique. Ensuite nous avons fait 3 acquisitions avec le bateau immobile pointant dans les 3 directions. Pour chaque acquisition le magnétomètre donnes les coordonnées cartésiennes x_{mag}, y_{mag}, z_{mag} de la direction pointée. Lors de l'acquisition nous avons pris 600 points de mesures et fait une moyenne pour obtenir le vecteur donnée par le magnétomètre. Pour une *direction* donnée on a le vecteur :

$$D_{direction} = \begin{pmatrix} x_{mag} \\ y_{mag} \\ z_{mag} \end{pmatrix}$$

On peut alors construire une matrice représentant le trièdre du magnétomètre.

$$Y = \begin{pmatrix} D_{nord} & D_{ouest} & D_{haut} \end{pmatrix}$$

Ainsi, il existe une matrice R_{inu} qui permet de faire la transformation de la matrice Y à la matrice Z. On a donc :

$$Z = R_{inu}Y$$

Ce qui nous permet de calculer R_{inu} pour calibrer les données acquises avec la formule :

$$R_{inu} = ZY^{-1}$$

L'ensemble du traitements des données et du calcul de R_{inu} est présent dans le fichier

Calibrage_ mag.py.

Cependant après avoir calculé R_{inu} à plusieurs reprises les résultats sur les données après modification étaient moins bonnes que les brutes. Nous avons notamment remarqué que lorsque nous effectuions une tour complet avec le bateau le cap ne variait pas de façons linéaire. Ainsi, nous avons décidé de ne pas se servir de cette correction pour calibrer le magnétomètre mais d'utiliser un offset de -10°.

3 Calibrage des moteurs

DDBoatthebest possèdes 2 systèmes de propulsions que l'on commande avec une valeur comprise dans l'intervalle $[-255, 255]$. comme les 2 hélices sont placées respectivement à droite et a gauche du bateau il faut mettre la même commande pour qu'il aille avance droit. Cependant, le notre suivait une courbe. Ceci implique que les systèmes de propulsions n'ont pas une puissance identique. Nous avons donc choisit d'appliquer une valeur d'offset à chaque moteur pour compenser leur différences de puissances. Ainsi, pour une commande $u \in [-255, 255]$ chaque moteur reçoit :

$$u_{mgauche} = u + 10 \qquad u_{mdroite} = u - 10$$

De plus nous avons choisit de n'utiliser les moteurs que vers l'avant, donc de borner notre commande dans l'intervalle $[0, 255]$.

4 Suivi de cap

L'objectif de cette mission consistait à donner une consigne de cap au DDBoat et qu'il la respecte. Pour cela nous avons utilisé une correction proportionnelle à l'erreur entre la consigne de cap et le cap calculé avec les données du magnétomètre calibré.

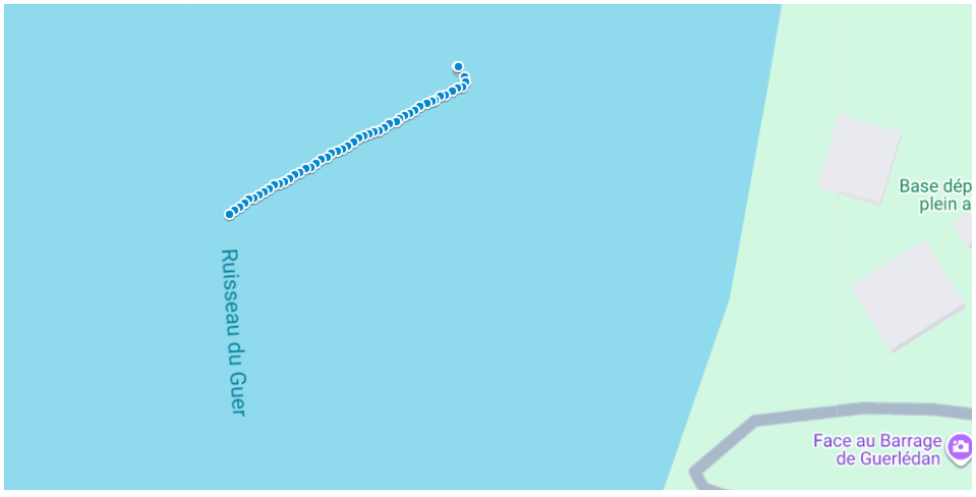


FIGURE 3 – Trajectoire du DDBoat pour une commande de suivi de cap de 240 degrés

5 Atteinte d'un point GPS

L'objectif est que notre DDBoat atteigne une bouée dont les coordonnées GPS sont connues et revienne au ponton.

5.1 Acquisition et traitement des données GPS

Le code utilisé pour la mission est le code *GoBeaconNBack.py*. D'abord nous récupérons les coordonnées GPS de notre DDBoat et faisons la convertion en mètres dans le repère de Lambert grâce à un code fourni par M.Zerr lors du cours de découverte des DDBoat. En comparant les coordonnées du point que nous souhaitons atteindre et les coordonnées GPS du DDBoat nous en déduisons le cap qu'il doit suivre et sa distance au point.

5.2 Mission bouée

Nous utilisons un correcteur PID pour régler la vitesse des moteurs.
Lorsque la distance entre les coordonnées de la bouée et les coordonnées de DDBoatthebest est inférieure à 5 mètres il considère qu'il a atteint son objectif et rentre au ponton.

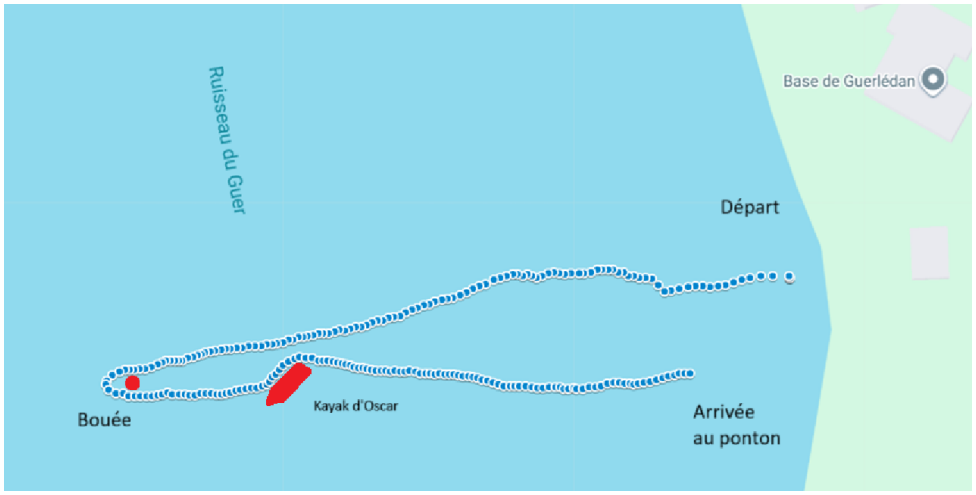


FIGURE 4 – Trajectoire du DDBoat

Sur le retour DDBoatthebest ne garde pas tout du long le même cap à cause d'une collision avec le kayak d'Oscar.
Grâce au suucès de cette mission, nous pouvons désormais lancer un programme *GoBack.py* à chaque fois que nous voulons récupérer DDBoatthebest après des essais sur le lac. C'est très pratique.

6 Mission de capture

Lors de cette mission un DDBoat se déplacera selon une trajectoire connue, il jouera le rôle de proie. L'objectif de cette mission est de capturer cette proie. Pour cela DDBoatthebest sera un chasseur et devra partir du ponton, se rapprocher de la proie et enfin tourner autour d'elle avec un rayon définie. De plus, c'est une mission coopérative : l'ensemble des DDBoat doivent tourner ensemble autour de la proie avec un déphasage entre eux prédéfinie et constant.

DDBoatthebest n'est pas équipé de moyen de communication pour détecter la proie donc pour l'atteindre nous devons avoir connaissance de sa trajectoire qui est donnée par l'équation paramétrique du cercle suivante :

$$p(t) = C + R_1 \begin{pmatrix} \cos(\omega_1(t - t_0) \\ \sin(\omega_1(t - t_0) \end{pmatrix}$$

Avec C les coordonnées GPS du centre du cercle, R_1 le rayon du cercle, ω_1 la vitesse angulaire et t_0 l'heure du départ de la mission.

Le cercle décrit par les DDBoat chasseurs est paramétré par l'équation :

$$a(t) = p(t) + R_2 \begin{pmatrix} \cos(\omega_2(t - t_0) + \frac{2\pi i}{N}) \\ \sin(\omega_2(t - t_0) + \frac{2\pi i}{N}) \end{pmatrix}$$

Avec R_2 le rayon du cercle, ω_1 la vitesse angulaire, N le nombre total de DDBoat et i le numéro de DDBoatthebest.

Cette mission peut être décomposé en 2 étapes clés : atteindre la proie puis lui tourner autour.

6.1 Atteindre la proie

DDBoatthebest avait pour objectif d'atteindre la position GPS donnée par l'équation de la trajectoire de la proie. Le code de la mission est *FollowBigBoat.py*. Lors de nos premiers essais, le bateau dépassait certaines positions GPS et donc faisait demi-tour pour l'atteindre. Pour résoudre ce problème nous avons augmenté la distance avec laquelle le bateau devait atteindre le point avant de passer au suivant. Nous avons aussi ajouté une condition de temps : si le DDBoat met plus de t secondes à atteindre la balise il passe à la suivante. Nous avons choisi t = 1 seconde. La figure suivante présente la position GPS de DDBoatthebest par rapport à la position simulée de la proie.

FIGURE 5 – En jaune la trajectoire du DDBoat, et en rouge une simulation de la trajectoire de la proie

On peut voir que le DDBoat ne respecte pas présisément la trajectoire de la proie. Cette erreur est due à un temps accordé pour atteindre la première balise trop court (10 secondes). Le temps accordé pour atteindre les balises suivantes est également trop court.

6.2 Tourner autour de la proie

Nous n'avons pas eu le temps de réaliser cette étape de la mission de capture.

7 Conclusion

Durant cette première semaine à Guerlédan nous nous sommes familiarisé avec le DDBoat. Nous avons réussi à obtenir des données de l'IMU et du traceur GPS et les utiliser pour commander le bateau. Malgré un début de semaine difficile nous avons réussi presque toutes les missions. Nous reviendrons pour tenter des missions plus complexes comme celle de la capture.

ENSTA | INSTITUT POLYTECHNIQUE DE PARIS

Le DD-Boat à l'épreuve

Rapport du projet expérimental de Guerlédan

EXPÉDITION 1

Clément SARTRE — clement.sartre@ensta.fr
Simon KLÉCHA — simon.klecha@ensta.fr

16 novembre 2025

Table des matières

Résumé

Du 03 au 07 novembre 2025, la classe de ROB 27 est partie a Guerlédan pour configurer les DD-Boat. Le groupe constitué de Simon KLÉCHA et Clément SARTRE a rencontré de nombreux problèmes (et parfois leurs solutions) pour répondre correctement aux différentes missions imposées par Luc Jaulin. Ceux-ci font l'objet de ce rapport.

Mots clés

DD-Boat, calibration, cap, localisation, IMU, GPS.

Abstract

From November 3 to 7, 2025, the ROB 27 class went to Guerlédan to configure the DD-Boat. The group consisting of Simon KLÉCHA and Clément SARTRE encountered numerous problems (and sometimes their solutions) in order to correctly complete the different missions assigned by Luc Jaulin. These are the subject of this report.

Keywords

DD-Boat, calibration, heading, localization, IMU, GPS.

Remerciements

Luc Jaulin et Benoît Zerr pour l'encadrement sur place, ainsi que Antoine et Basile, et Fabrice Le Bars pour notre indispensable préparation. Nous exprimons un sentiment de reconnaissance particulier envers le personnel du self, à qui nous devons bien 90% de notre réussite.

Liste des acronymes

ENSTA École Nationale Supérieure de Techniques Avancées

IMU Inertial Measurement Unit

GPS Global Positioning System

1 Introduction

Ce séjour s'est organisé en objectifs évolutifs. Nous avons découvert le bateau le lundi 03 novembre à 11h30 ; ensuite, chaque action s'est faite en vue d'un objectif daté. Les étapes de cette aventure sont détaillées par ordre chronologique dans la suite, en vue de répondre à la question suivante : avons-nous réussi le défi Guerlédan EXPÉDITION 1 ?

2 Découverte générale du hardware

L'aspect hardware du projet a été pour nous un défi. D'une part, plusieurs heures ont été gâchées à faire des expériences avec des batteries trop déchargées, rendant nos observations erronées. L'installation préalable des composants nous a laissé quelques surprises, et en particulier notre boîtier principal n'était pas fixé à la coque. L'IMU (visible sur la figure 1) s'est donc délogé de son support en polystyrène, et 48 heures furent nécessaires pour nous en rendre compte.

FIGURE 1 – Photo de l'équipe 13 devant l'IMU désolidarisé

Cela a empêché chacune de nos tentatives de calibration de fonctionner. En outre, il nous a fallu improviser pour garantir l'immobilité des composants dans le bateau par nos propres moyens, en particulier avec l'usage de ruban adhésif. Nous avons donc pris l'habitude, grâce à ce projet, de vérifier systématiquement l'installation hardware avant de tester nos codes.

3 Cap à l'Ouest

C'était une pré-mission, en quelque sorte. Notre bateau devait être capable d'avancer vers l'Ouest pendant 15 secondes. Il s'est majoritairement agi de découvrir notre IMU.

3.1 Réorientation fictive de l'IMU

Une fois notre IMU solidarisé avec le DD-Boat, il a fallu le calibrer, c'est-à-dire transformer une ellipsoïde en sphère, et l'aligner avec le bateau, car il n'est jamais parfaitement positionné.

En effet, la correction des effets *hard-iron* et *soft-iron* nous permet de ramener les mesures magnétiques brutes de l'IMU sur une sphère centrée. Exemple sur la figure 2.

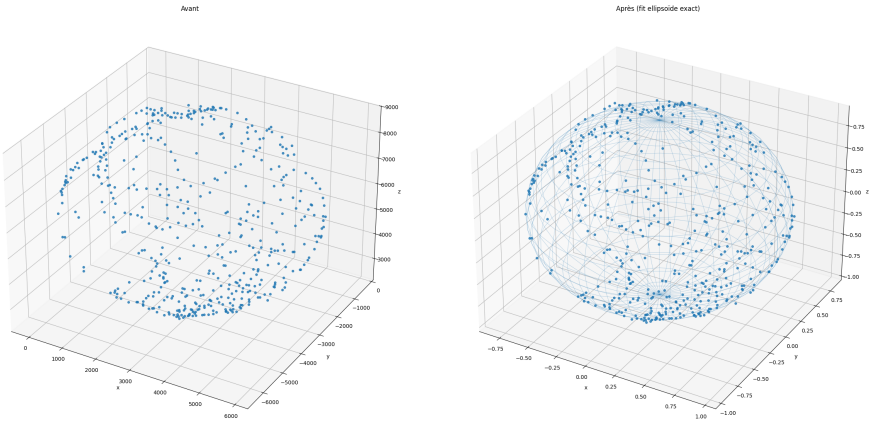


FIGURE 2 – Affichage des nuages de points avant et après calibration

Cette étape garantit une norme cohérente du champ magnétique, mais ne fixe pas encore l'orientation du repère associé à cette sphère. En pratique, l'axe x de la sphère corrigée peut très bien correspondre à l'axe y du repère que l'on souhaite utiliser pour le bateau. Il faut donc effectuer une rotation supplémentaire pour aligner la base de l'IMU avec le repère de référence choisi.

3.1.1 Formulation mathématique

On note $\mathbf{m}_{\text{raw}} \in \mathbb{R}^3$ la mesure magnétique brute. Les étapes de correction précédentes nous fournissent un vecteur de biais $\mathbf{b}$ et une matrice de transformation $\mathbf{C}$ qui replie l'ellipsoïde de mesures en sphère. La mesure calibrée s'écrit alors

$$\mathbf{m}_{\text{cal}} = \mathbf{C}\,(\mathbf{m}_{\text{raw}} - \mathbf{b}). \tag{1}$$

La matrice $\mathbf{C}$ n'est pas nécessairement orthogonale : elle corrige les distorsions d'échelle et les effets d'inclinaison induits par le *soft-iron*.

Pour imposer un repère de référence cohérent avec le bateau, nous avons réalisé trois mesures calibrées de l'IMU dans des orientations contrôlées :

— $\mathbf{m}_{\text{N}}$ avec l'étrave pointée vers le nord (axe x recherché) ;
— $\mathbf{m}_{\text{W}}$ avec l'étrave vers l'ouest (axe y recherché) ;
— $\mathbf{m}_{\text{U}}$ avec la coque tournée vers le haut et la proue vers l'ouest (axe z recherché).

Ces vecteurs sont rassemblés dans la matrice

$$\mathbf{Y} = \begin{bmatrix} \mathbf{m}\text{N} & \mathbf{m}\text{W} & \mathbf{m}_{\text{U}} \end{bmatrix}. \tag{2}$$

3.1.2 Construction pratique de la matrice Y

Pour rendre explicite la construction de $\mathbf{Y}$, nous immobilisons le bateau dans trois poses orthogonales et enregistrons le vecteur magnétique calibré $\mathbf{m}_{\text{cal}}$ à chaque fois :

1. **Proue vers le nord** : l'axe longitudinal du bateau (futur axe x du repère navire) est aligné avec le nord géographique. Le vecteur mesuré est noté $\mathbf{m}_{\text{N}}$.
2. **Proue vers l'ouest** : même procédure en faisant pivoter le bateau de 90° autour de l'axe vertical. On obtient $\mathbf{m}_{\text{W}}$, associé au futur axe y.

3. **Coque vers le haut** : le bateau est posé sur le flanc (ventre vers l'ouest) afin d'aligner l'axe vertical z recherché avec le champ terrestre. La mesure fournie est notée $\mathbf{m}_{\mathrm{U}}$.

Ces trois vecteurs sont linéairement indépendants et forment les colonnes de $\mathbf{Y}$. Les orientations sont résumées dans la figure 3.

Par construction, chaque colonne de la matrice théorique

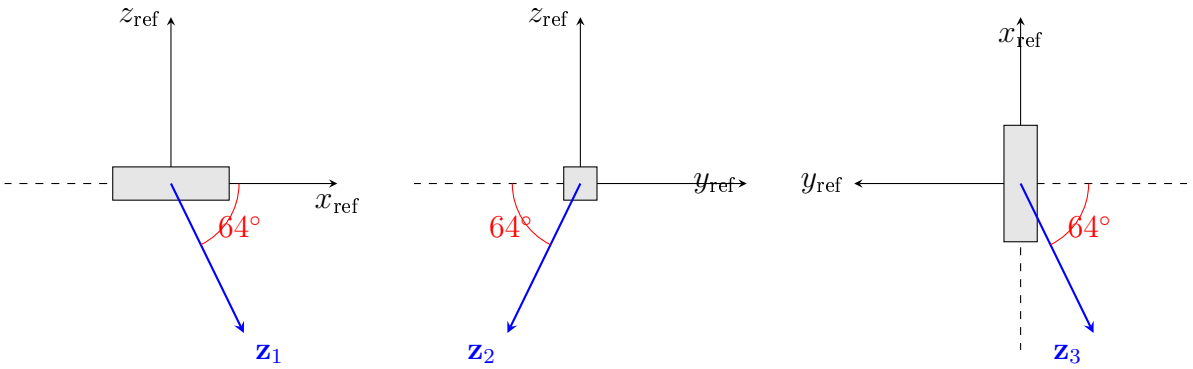
$$\mathbf{Z} = \begin{bmatrix} \cos I & 0 & -\sin I \\ 0 & -\cos I & -\cos I \\ -\sin I & -\sin I & 0 \end{bmatrix}$$

décrit la direction attendue du champ magnétique pour une pose donnée :

$$\mathbf{z}_1 = \begin{bmatrix} \cos I \\ 0 \\ -\sin I \end{bmatrix} \quad \text{(proue vers le nord, composante uniquement dans le plan } x\text{–}z\text{)},$$

$$\mathbf{z}_2 = \begin{bmatrix} 0 \\ -\cos I \\ -\sin I \end{bmatrix} \quad \text{(proue vers l'ouest, composantes } y \text{ et } z \text{ négatives)},$$

$$\mathbf{z}_3 = \begin{bmatrix} -\sin I \\ -\cos I \\ 0 \end{bmatrix} \quad \text{(coque vers le haut, direction purement horizontale)}.$$



Pose 1 : proue nord | Pose 2 : proue ouest | Pose 3 : coque vers le haut

FIGURE 3 – Vecteurs du champ magnétique (en bleu).

À Guérledan, nous avons l'inclinaison magnétique $I = 64°\pi/180$. Le champ théorique exprimé dans le repère du navire s'écrit alors

$$\mathbf{Z} = \begin{bmatrix} \cos I & 0 & -\sin I \\ 0 & -\cos I & -\cos I \\ -\sin I & -\sin I & 0 \end{bmatrix}, \tag{3}$$

dont les colonnes correspondent aux vecteurs cibles pour les trois orientations imposées.

La rotation cherchée est la matrice $\mathbf{R}_{\mathrm{IMU}} \in \mathrm{SO}(3)$ qui mappe les mesures sur leurs références :

$$\mathbf{R}_{\mathrm{IMU}} = \mathbf{Z}\,\mathbf{Y}^{-1}. \tag{4}$$

Cette relation suppose que les trois mesures de référence sont linéairement indépendantes, condition respectée dès lors que les axes sont suffisamment séparés. On obtient finalement la mesure réorientée

$$\mathbf{m}_{\text{ref}} = \mathbf{R}_{\text{IMU}}\, \mathbf{m}_{\text{cal}}, \tag{5}$$

désormais exprimée dans le repère de référence voulu. Cette dernière rotation correspond à la « réorientation fictive » : elle ne modifie pas la magnitude des mesures, mais aligne virtuellement l'IMU avec l'architecture mécanique.

4 Aller-retour bouée

4.1 Cheminement

Notre première mission a été de partir du ponton et d'atteindre une bouée aux coordonnées supposément fixes, avant d'en revenir. Bien qu'il soit possible d'effectuer cette mission uniquement en gardant le bon cap à une vitesse fixe pendant un temps choisi, il nous est apparu que cette méthode était peu fructueuse. Le courant du lac et le déplacement perpétuel de la bouée rendaient cette opération inachevable. Le seul moyen d'y arriver est donc de combiner un IMU (bien calibré) et un récepteur GPS.

4.2 Réussite

Bien qu'ayant échoué à accomplir cette mission au moment initialement demandé (mercredi à 11h30), nous avons réussi 24 h plus tard, et les données du trajet sont affichées figure 4.

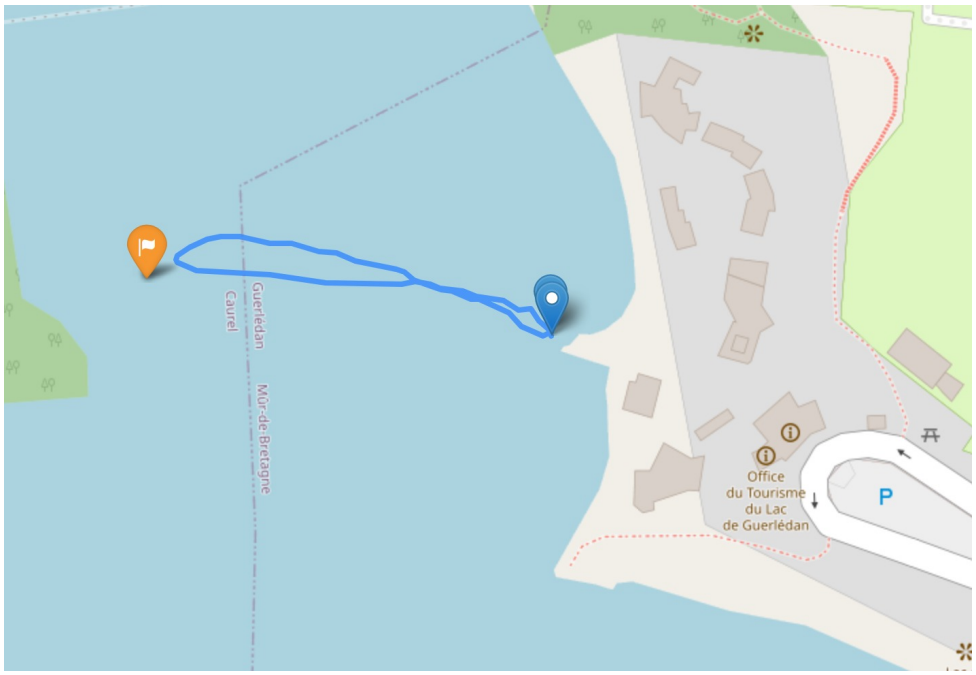


FIGURE 4 – Affichage sur une carte du trajet effectué par le DD-Boat

Une preuve vidéo de notre succès est également disponible. On notera l'échec de Clément à aller plus vite que le DD-Boat.

5 Capture

Notre dernière mission a consisté à donner collectivement l'illusion d'une capture en spirale d'un bateau proie. Individuellement, les DD-Boats ne communiquant pas encore entre eux, il fallait en réalité faire un suivi de cible : comme l'aller-retour bouée, mais avec une bouée qui décrit un grand cercle. Notre trajectoire était donc une cycloïde. Le vendredi 07 novembre à 10h47, notre bateau a glorieusement effectué le parcours. Luc Jaulin est probablement le seul détenteur de la vidéo par drone de l'évènement, mais nous avons enregistré les coordonnées du trajet. Elles sont affichées sur la figure 5 : on peut apercevoir en rouge le trajet cible, et en bleu le trajet réel.

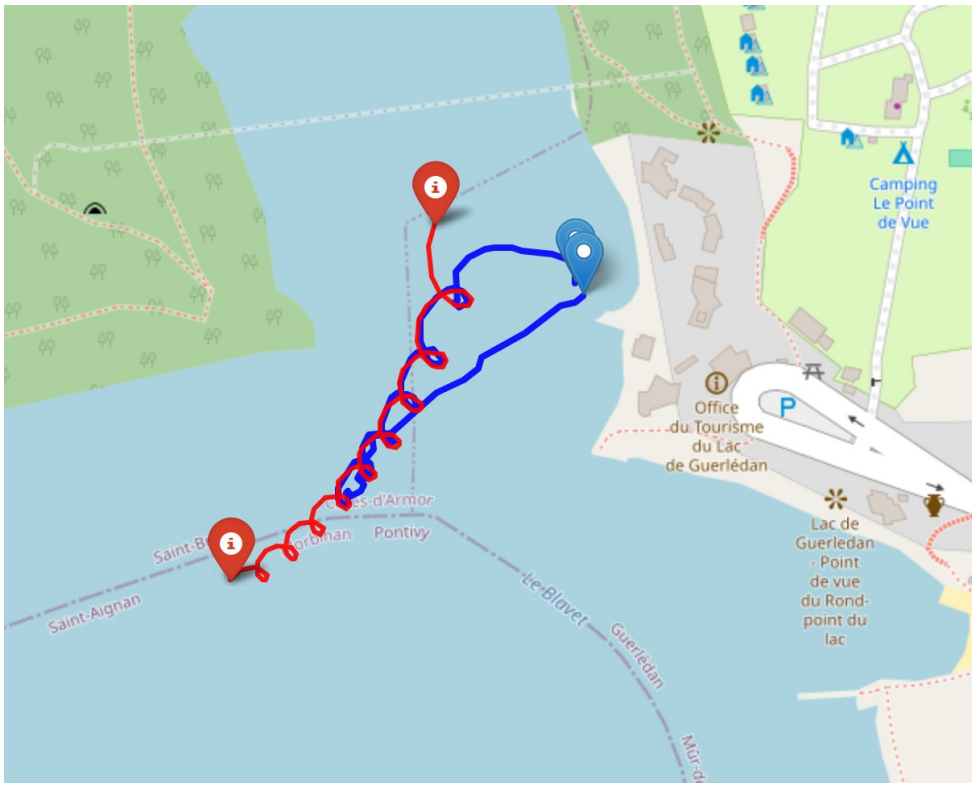


FIGURE 5 – Affichage sur une carte du trajet effectué par le DD-Boat

On observe qu'au bout du temps de trajet réglementaire de 750 s, le DD-Boat a arrêté sa « capture ». Il a donc fallu le faire revenir avec la télécommande, comme on le voit sur la figure 5.

6 Conclusion

Ainsi, nous avons su réussir chaque mission de Guerlédan EXPÉDITION 1, soit le séjour dans son intégralité. Nous savons à présent qu'il faut porter une attention particulière au hardware lorsqu'un problème se pose. La calibration d'un IMU et l'utilisation d'un récepteur GPS n'ont plus de secrets pour nous. En fin de compte, nous pensons à présent que l'essence même de la robotique pratique est l'*adaptation*.

Projet Guerlédan : DDBoats

ROB27

4 novembre 2025

Thibault Lucas
Gaspard DELPIANO–MANFRINI
Matéo BRUN

Sommaire

1 Introduction

Le but de la semaine passée au lac de Guerlédan était de nous faire travailler dans des conditions "réelles" de robotique de terrain. Nous avons pu constater les différences entre théorie et pratique durant ces cinq jours.

Par exemple, la partie "auto-calibration" semblait plutôt rapide à réaliser, et pourtant nous nous sommes confrontés à différents problèmes dus à la version totalement obsolète de python sur les DDBoats, ou à l'absence de certaines bibliothèques qui nous ont forcés à recréer des fonctions (avec plus ou moins de succès).

Ainsi, malgré les consignes en apparence assez simples, ces cinq jours furent très denses, mais aussi instructifs en prévision de nos futurs projets. Bien que le résultat final ne soit pas parfait, nous sommes allés au bout de toutes les missions imposées, ce qui reste satisfaisant.

FIGURE 1 – La fine équipe, avec le bateau 3. Nous avons changé de bateau le dernier jour et avons pris le bateau 10, pour cause de cours-circuits intempestifs et de l'arrachage de notre IMU sur le bateau 3, qui impliquait de toute façon à une recalibration complète.

2 Calibration IMU et Cap

2.1 Théorie pour la calibration

Afin de connaître l'orientation du DDBoat et notamment de repérer la direction du Nord, ce dernier utilise un magnétomètre trois axes présent dans sa centrale inertielle. Cependant, ses mesures peuvent être perturbées par des composants ferromagnétiques. Cet effet s'explique par deux phénomènes: les composants métalliques magnétiques du bateau génèrent un champ magnétique qui s'ajoute au champ magnétique terrestre et donc le parasitent (*hard iron*) et les courants électriques circulant dans le bateau génèrent un champ magnétique dit induit qui parasite également la mesure(*soft iron*). Le champ capté par le magnétomètre est alors une ellipse déformée au lieu d'une sphère normalisée. Nous allons donc commencer par corriger cette erreur par autocalibration.

L'équation d'une ellipsoïde s'écrit :

$$\begin{aligned} f(x) &= p_1x_1^2 + p_2x_2^2 + p_4x_1x_2 + p_5x_1x_3 + p_6x_2x_3 + p_7x_1 + p_8x_2 + p_9x_3 \\ &= 1 \text{ (car on veut une sphère normalisée)} \end{aligned}$$

Le but est donc d'abord de trouver les paramètres de l'ellipse $\vec{p} = (p_1\ p_2\ p_3\ p_4\ p_5\ p_6\ p_7\ p_8\ p_9)^T$
tels que :

$$\begin{bmatrix} x_1^2(1) & x_2^2(1) & x_3^2(1) & x_1(1)x_2(1) & x_1(1)x_3(1) & x_2(1)x_3(1) & x_1(1) & x_2(1) & x_3(1) \\ x_1^2(2) & x_2^2(2) & x_3^2(2) & x_1(2)x_2(2) & x_1(2)x_3(2) & x_2(2)x_3(2) & x_1(2) & x_2(2) & x_3(2) \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ x_1^2(i) & x_2^2(i) & x_3^2(i) & x_1(i)x_2(i) & x_1(i)x_3(i) & x_2(i)x_3(i) & x_1(i) & x_2(i) & x_3(i) \end{bmatrix} \vec{p} = \begin{bmatrix} 1 \\ 1 \\ . \\ . \\ . \\ 1 \end{bmatrix}$$

$<=> M\vec{p} = (1\ 1\ \ldots\ 1)^T$

où x(i) est le i-ème point mesuré. En appliquant la méthode des moindres carrés, on obtient :

$$\vec{p} = (M^TM)^{-1}.M^T.\begin{bmatrix} 1 \\ 1 \\ . \\ . \\ . \\ 1 \end{bmatrix}$$

f(x) s'écrit également $(x-b)^TQ(x-b)$

On veut donc : $(x-b)^TQ(x-b) = 1$

On a : $\begin{aligned} \frac{df}{dx} &= 2(x-b)^TQ = 2x^TQ - 2b^TQ \\ &= \begin{bmatrix} \frac{\partial f}{\partial x_1} & | & \frac{\partial f}{\partial x_2} & | & \frac{\partial f}{\partial x_3} \end{bmatrix} \\ &= \begin{bmatrix} 2p_1x_1 + p_4x_2 + p_5x_3 + p_7 & | & 2p_2x_2 + p_4x_1 + p_6x_3 + p_8 & | & 2p_3x_3 + p_5x_1 + p_6x_2 + p_9 \end{bmatrix} \end{aligned}$

On en déduit :
$Q = \begin{bmatrix} p_1 & \frac{1}{2}p_4 & \frac{1}{2}p_5 \\ \frac{1}{2}p_4 & p_2 & \frac{1}{2}p_6 \\ \frac{1}{2}p_5 & \frac{1}{2}p_6 & p_3 \end{bmatrix}$

et $b = -\frac{1}{2}Q. \begin{bmatrix} p_7 \\ p_8 \\ p_9 \end{bmatrix}$

De plus : $(x-b)^T Q(x-1) = 1 <=> (x-b)^T \sqrt{Q}\sqrt{Q}(x-b) = 1$ (Q est symétrique définie positive)

Donc y = $\sqrt{Q}.(x-b), \qquad ||y|| = 1$

Cependant ces formules font l'hypothèse implicite que ellipsoïde contient le point d'origine ce qui n'est pas forcement le cas.

En posant $y = x - b$
Avec $2Ab + c = 0$

On résoud le calucul suivant :

$x^T Ax + c^T x = (y+b)^T A(y+b) + c^T(y+b) = 1$

$<=> y^T Ay + b^T Ay + y^T AB + b^T Ab - 2(Ab)^T y - 2(Ab)^T b = 1$

$<=> \frac{y^T Ay}{1+b^T Ab} = \frac{1+b^T Ab}{1+b^T Ab}$

On en déduit : $b_{corr} = -\frac{1}{2}A^{-1}c$

$G_{corr} = \frac{Q}{1+b^T Ab}$

2.2 Théorie pour obtenir le cap

On a comme données :
Le vecteur y_1, donné par le magnétomètre (corrigé et normalisé)
Le vecteur a_1, donné par l'accéléromètre (normalisé), par soucis de simplicité on choisira $a_1 = a_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ (le bateau reste horizontal si on néglige l'influence des vagues, absentes sur ce lac, qui reste néanmoins très beau)

On peut calculer la matrice R = $\left[\frac{y_1(y_1^T a_1)a_1}{||y_1(y_1^T a_1)a_1||} \quad | \quad a_1 \times \frac{y_1(y_1^T a_1)a_1}{||y_1(y_1^T a_1)a_1||} \quad | \quad a_1 \right]$

On obtient ainsi les angles d'Euler par :

$\phi = atan2(R_{3,2}, R_{3,3})$

$\theta - arcsin(R_{3,1})$

$\psi = atan2(R_{2,1}, R_{1,1})$

La seule valeur qui nous intéresse alors est le cap ψ dans notre cas, à noter que si d'autres angles nous avaient intéressés, nous aurions dû récupérer la valeur a_1 et non plus a_0.

2.3 Recalage en cap

En fonction de l'orientation de l'IMU au sein de la structure du bateau il peut y avoir un décalage constant entre le cap du bateau et le cap de l'IMU. Pour pallier ce problème, on cherche à calculer la matrice de rotation correspondant à la rotation de l'IMU dans le bateau. Par définition de R_{IMU} la rotation de l'IMU dans le bateau on déduit la relation suivante où Y sont les points sur la sphère associée au cap et Z sont ces mêmes points après recalage:

$$Z = R_{IMU} * Y$$

En posant :

$$Z_{NWU} = \begin{bmatrix} cos(I) & 0 & -sin(I) \\ 0 & -cos(I) & 0 \\ -sin(I) & -sin(I) & cos(I) \end{bmatrix}$$

où I est l'angle entre le plan tangent à la surface de la terre et les lignes de champ magnétique. Au niveau de Guerlédan, on pourra prendre I = 64° Z correspond dans ce cas aux valeurs théoriques pour un cap bien réglé des points associés au caps plein Nord puis plein Ouest et enfin vers le haut. Il ne nous reste plus qu'à mesurer Y_N, Y_W, Y_U puis par la relation précédemment établie on en déduit :

$$R_{IMU} = Z_{NWU} * Y_{NWU}^{-1} \text{ où } Y_{NWU} = \begin{bmatrix} Y_N & | & Y_W & | & Y_U \end{bmatrix}$$

NB: chaque colonne de Z correspond au projeté du champ magnétique sur chacun des axes compte tenu de l'orientation du bateau.

Une amélioration qui a pu ensuite être apportée consiste à projeter la matrice R_{IMU} obtenue, qui dû à des imprécisions lors de l'acquisition des Y_N, Y_W, Y_U n'est pas complètement une matrice de rotation, sur SO(3). Pour cela nous avons utilisé la décomposition QR et son implémentation python. La matrice Q est la matrice de rotation voulue.

2.4 Notre calibration

Le tracé à gauche de la figure 2 expose le nuage de points que nous avons obtenu avant l'auto-calibration. On remarque que les points sont plutôt bien répartis, et donc la "danse du bateau" a bien été réalisée : l'IMU a atteint toutes les positions de manière plutôt équilibrée.

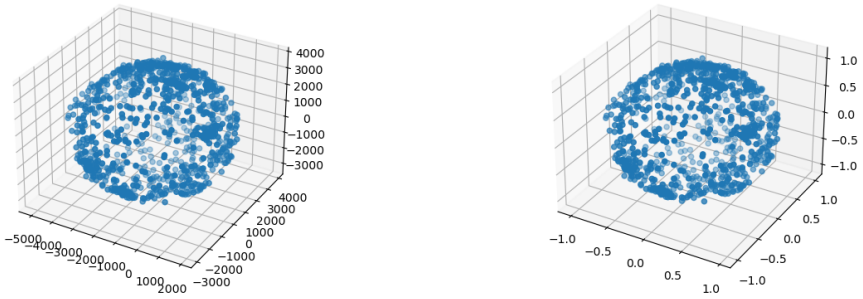


FIGURE 2 – A gauche, nuage de points avant l'autocalibration, formant une ellipsoïde. A gauche, le nuage de point après autocalibration, formant une sphère centrée réduite

Après de nombreux debugs et de nombreuses tentatives pour faire fonctionner notre code, nous avons enfin obtenu la sphère normalisée correspondante, qui est présentée à droite de la figure 2.

2.5 Problèmes résolus

Une mauvaise compréhension de la matrice Z_{NWU} lors de la calibration de la position verticale où l'orientation de la coque vers le Nord n'avait qu'implicitement été mentionnée nous a posé problème. Un recalcul de la matrice nous a permis d'identifier le problème.

Le principal problème que nous avons rencontré durant cette calibration venait du fait que nous faisions tourner nos programmes de calibration directement sur le DDBoat. En effet, la version de python sur ces bateaux est très datée, et certains modules n'y étaient pas installés. Cela a notamment posé problème pour le calcul de la racine carrée de Q (pas de fonction sqrtm sur le DDBoat). Nous avons donc tenté de créer une fonction (par chatGPT pour être honnêtes), sauf que celle-ci ne marchait pas et que nous avions peu de moyens de la tester, vu que ce n'est pas très intuitif de voir si la sortie est bien la racine de la matrice.
Après quelques essais nous avons donc dû changer de méthode et nous servir du DDBoat uniquement pour l'acquisition des points et leur stockage dans des fichiers JSON (plus pratiques que des fichiers TXT). Nous avons ensuite pu calculer les matrices de calibration sur l'ordinateur et les utiliser sur le DDBoat en les retransférant une fois de plus dans des fichiers JSON. Se servir de l'intermédiaire de l'ordinateur dès le départ nous aurait certainement fait gagner beaucoup de temps...

3 Mission suivi de cap

La première mission consistait à faire suivre un cap fixe au bateau (ici aller à l'ouest pendant 15 secondes).

3.1 Théorie : le *feed-forward control*

Une fois la boussole calibrée, nous avons donc asservi le cap du bateau avec la méthode *feed-forward control* (Cf figure 3). Pour cela, nous avons pris comme "perturbation" au signal d'entrée (l'angle -90°) la différence entre le cap mesuré et l'entrée en question. Cela a été possible car nous connaissions entièrement les équations d'état de notre système (sous hypothèses simplificatrices, notamment le fait que le bateau évoluait dans un plan).

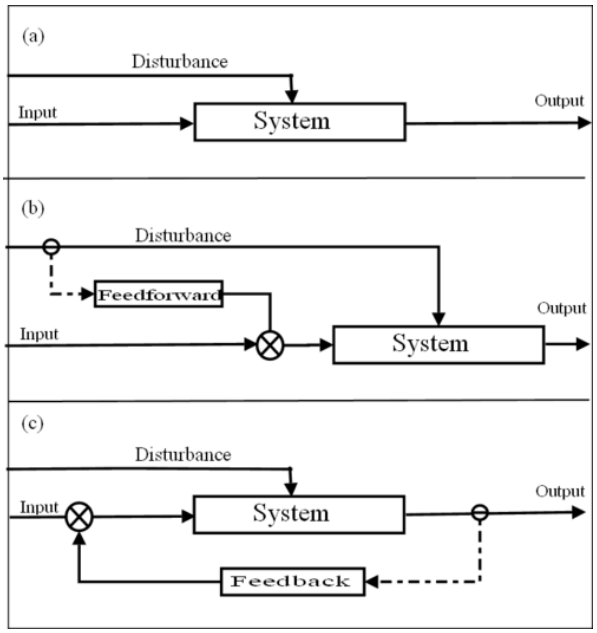


FIGURE 3 – Les trois différents types de contrôle linéaire. Source image : Wikipédia

3.2 Notre implémentation

Cette première mission a été réussie. Le robot a suivi le cap imposé pendant les 15 secondes voulues, malgré le courant du lac. Les logs de la trajectoire sont exposés dans la figure 4.

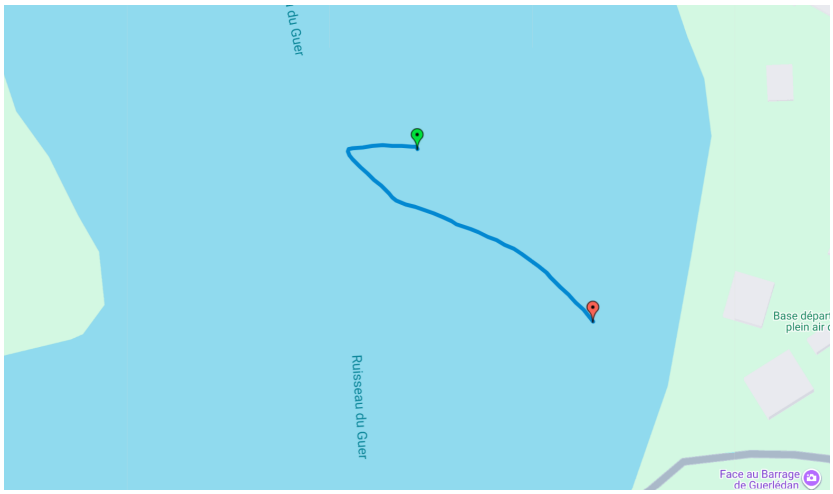


FIGURE 4 – Tracé des logs de la mission "aller à l'ouest pendant 15 secondes" et du retour à la base. Le point de départ est le marqueur vert

3.3 Problèmes résolus

La calibration en rotation n'étant pas toujours optimale, nous avons eu recours à une matrice de rotation codée en dur imposant un offset au cap. Cela nous a permis d'avoir un contrôle correct.

4 Mission suivi de balise

Lors de la deuxième mission, le bateau avait pour objectif d'aller jusqu'à une bouée localisée par ses coordonnées GPS.

4.1 Théorie : projection des coordonnées GPS dans un plan

Pour simplifier le contrôle et se ramener à un problème de contrôle du cap, nous avons projeté les coordonnées GPS de tous les éléments dans un plan dont l'origine était le bout du ponton. Pour cela, nous avons essayé deux méthodes : l'utilisation du module de projection déjà implémenté dans Python (ProjDegree2Meter) et l'implémentation de notre propre fonction de projection.

Prenant l_x, l_y la latitude et la longitude d'un point à projeter, l_x^M, l_y^M la latitude et la longitude de l'origine de notre repère et ρ le rayon du parallèle passant par l'origine du repère, on a que les coordonnées du point dans le plan sont :

$$\begin{cases} x = \rho \cdot cos(l_y^M) \cdot (l_x - l_x^M) \\ y = \rho \cdot (l_y - l_y^M) \\ z = \rho - \rho^M \end{cases}$$

4.2 Notre implémentation

Nous avons finalement opté pour la méthode ProjDegree2Meter, que nous avons trouvé plus précise et qui permettait de mieux se synchroniser avec les autres robots pour la mission finale. La trajectoire aquise est présentée dans la figure 5.

FIGURE 5 – Tracé des logs de la mission d'aller-retour à une bouée

4.3 Problèmes résolus

Mention honorable à la fonction projDegree2Meter, qui est indiquée sur le diapo comme prenant en paramètres (lon,lat) alors que c'est (lat,lon) : 2h perdues (la prochaine fois je demanderai à chatGPT de debug).

5 Mission prise en chasse d'un DDboat "proie"

Cette dernière mission avait pour but de faire suivre à chaque DDboat une trajectoire d'interception d'un bateau "proie". Comme la communication entre les bateaux n'était pas possible, chaque trajectoire devait être fixée à l'avance en tant que phase. Les bateaux étaient synchronisés en se réglant tous sur l'heure GPS.

5.1 Théorie : construction de la trajectoire à suivre

Pour faire suivre une trajectoire au bateau, le principe est de créer une "balise" fictive dont la position évolue au cours du temps en suivant la formule mathématique voulue. Ici celle-ci était :

Trajectoire prédateur : $a(t) = p(t) + R_2(t) \cdot \begin{bmatrix} cos(\omega_2(t) \cdot t) \\ sin(\omega_2(t) \cdot t) \end{bmatrix}$ avec : $p(t) = c + R_1 \cdot \begin{bmatrix} cos(\omega_1 \cdot t) \\ sin(\omega_1 \cdot t) \end{bmatrix}$ (trajectoire proie), $R_2(t) = T_{exp} \cdot \exp \frac{-t}{T_{exp}} + 10$ et $\omega_2(t) = \frac{1}{10R_2(t)}$

Les paramètres étaient ensuite à choisir en commun pour convenir à l'environnement du lac.

FIGURE 6 – Forme de la trajectoire théorique d'un bateau en chasse de la "proie" pour différents paramètres

Le bateau "proie" suivait ainsi une trajectoire en arc de cercle, tandis que les autres bateaux se positionnaient autour de lui, puis lui tournaient autour et se rapprochaient progressivement jusqu'à la fin de la mission. On peut voir dans la figure 6 en bleu la trajectoire théorique d'un bateau prédateur, en forme d'épicycloïde, pour différents paramètres.

5.2 Notre implémentation

Nous avons eu le temps de réaliser deux tentatives "complètes", dont les logs sont présentés dans la figure 7. Lors de la deuxième tentative, le DDBoat a bien suivi le début de la trajectoire, mais est arrivé à court de batterie au milieu de la tentative, ce qui explique la divergence observée dans la deuxième image. Cette deuxième trajectoire est le résultat du lancé synchronisé avec les autres groupes le vendredi matin. Les deux trajectoires obtenues sont cependant très satisfaisantes, et un réglage plus fin des paramètres aurait permis d'obtenir la trajectoire exacte.

FIGURE 7 – Trajectoires réelles de notre DDBoat

5.3 Problèmes résolus

Le bateau proie était trop rapide par rapport à la capacité de nos bateaux, ce qui faisait que notre bateau avait tendance à prendre du retard et donc à couper certaines boucles. Se synchroniser avec tout le monde pour avoir les mêmes paramètres était également un défi.

6 Conclusion

Ces cinq jours de travail intensif ont été particulièrement denses et exigeants. Certaines journées se sont étendues de 9 h à minuit, avec seulement deux pauses pour les repas : un rythme soutenu qui s'est révélé aussi éprouvant que formateur. Malgré la fatigue accumulée, nous gardons de cette expérience un souvenir très positif.

Tout au long du projet, nous avons rencontré de nombreuses difficultés : des erreurs dans la formule d'origine, des détails techniques insignifiants qui nous ont fait perdre des heures, ou encore des ajustements imprévus à effectuer dans l'urgence. Ces moments de frustration ont néanmoins été essentiels à notre progression, car ils nous ont permis de mieux comprendre la logique du système et d'affiner nos méthodes de travail en équipe.

Finalement, nous avons atteint les objectifs fixés, et la satisfaction de voir notre DDBoat fonctionner de manière autonome a largement compensé les efforts fournis. Observer le bateau accomplir ses tâches avec succès — sans que nous ayons à sauter dans le kayak toutes les dix minutes — a été une véritable récompense. Cette aventure, bien que fatigante, a été à la fois formatrice, stimulante et très gratifiante.

7 ANNEXE

Vous pourrez trouver tous nos codes et les fichiers logs originaux dans le répertoire GitLab suivant : Notre GitLab. Les deux principaux programmes sont driver.py et control.py.

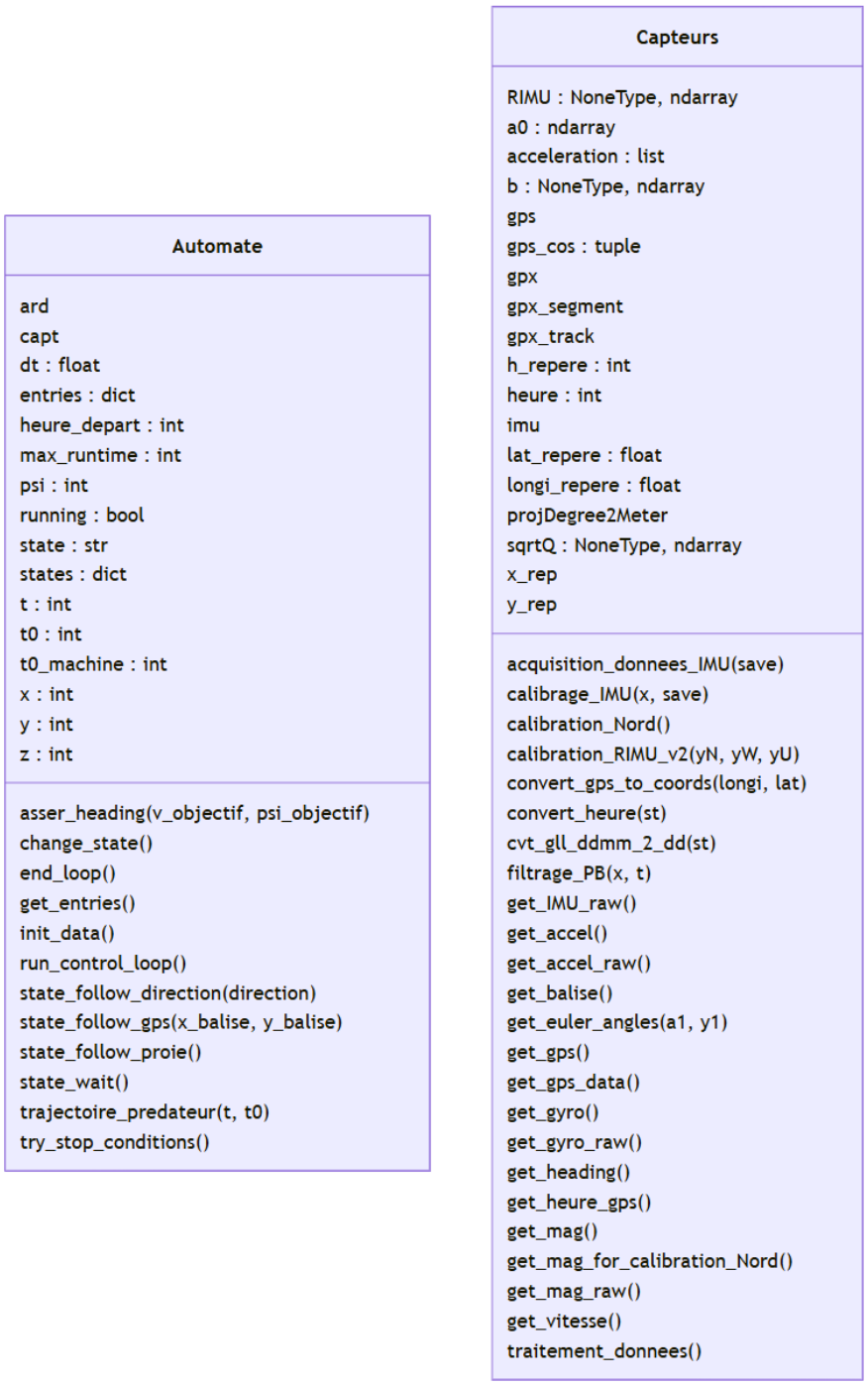


FIGURE 8 – Diagramme de classe de notre architechture logicielle