



UV 6.1 – Projet au lac de Guerlédan

Bathymétrie autonome par USV

Auteurs :

M Kevin BEDIN - ROB CI2020
M Clément BICHAT - ROB CI2020
Mme Pauline CELTON - HYO CI2020
M Aurélien GRENIER - ROB CI2020
Mme Ombeline LE GALL - HYO CI2020
Mme Aurélie PANETIER - HYO CI2020

Encadrants :

M Rodéric MOITIÉ
M Yoann SOLA

2019-2020

Remerciements

Nous remercions l'ensemble de l'équipe pédagogique de l'ENSTA Bretagne¹ qui a permis la réalisation de ce projet au Lac de Guerlédan. Ainsi, les enseignants des spécialités « Hydrographie Océanographie » et « Robotique » ont su faire face à une logistique et une préparation conséquentes, et nous ont apporté les notions théoriques et pratiques essentielles pour la réalisation de ce projet.

Nous tenons à remercier Fabrice Le Bars et Alain Bertholom pour l'aide qu'ils nous ont apportée lors de la mise en place du réseau, ainsi que pour l'utilisation du logiciel Mission Planner. Nous remercions également l'équipe de Texys Marine qui était présente lors de la première séance au Lac de Guerlédan, et qui nous a été d'une grande aide lors de la calibration des paramètres de l'autopilote.

Nous remercions particulièrement Romain Schwab pour sa patience durant les deux semaines sur le terrain afin de régler les instruments utilisés, ainsi que pour son aide durant l'acquisition, le traitement et l'interprétation des données.

Enfin, nous remercions également nos encadrants pour ce projet, Rodéric Moitié et Yoann Sola, pour leur soutien et leur disponibilité dans toutes les étapes de réflexion et d'implémentation du projet.



1. École Nationale Supérieure de Techniques Avancées Bretagne

Résumé — De nos jours, la demande de levés hydrographiques réalisés par les hydrographes est en hausse. Les étapes de réalisation afin d'obtenir des données bathymétriques exploitables peuvent parfois être longues et fastidieuses. C'est pour répondre à ce type de problématique que de nouvelles méthodes d'acquisition et de validation des données peuvent être mises en place. Dans le cadre du projet se déroulant au Lac de Guerlédan, c'est un robot catamaran, surnommé Ulysse, qui effectue les levés, muni d'un sondeur multifaxceau R2Sonic 2020. Le but du projet est de rendre le robot de surface autonome lors de l'acquisition des données, notamment en effectuant un contrôle qualité en temps réel afin de réaliser un levé exploitable sans intervention humaine, grâce à un retour automatique sur zone lorsque des mesures ne répondant pas aux exigences de l'hydrographe sont détectées. Ainsi, ce rapport présente le développement d'une architecture logicielle grâce au middleware ROS² afin de contrôler la navigation du robot, l'acquisition des données ainsi que le contrôle qualité de ces dernières. Après avoir filtré les données en temps réel à l'aide de plusieurs algorithmes, formatant et vérifiant la cohérence des données, le lot de données de profondeur de la zone sondée par Ulysse est alors validé. Ce levé pourra alors permettre la réalisation d'un MNT³ satisfaisant.

Abstract — Today, the demand for hydrographic surveys by hydrographers is increasing. The steps involved in obtaining usable bathymetric data can sometimes be long and tedious. It is to respond to this type of problem that new methods of data acquisition and validation can be implemented. As part of the project taking place at Guerlédan lake, it is a catamaran robot, nicknamed Ulysse, which carries out the surveys, equipped with an R2Sonic 2020 multibeam sounder. The aim of the project is to make the surface drone autonomous during data acquisition, in particular by carrying out quality control in real time in order to carry out a usable survey without human intervention thanks to an automatic return to the area when measurements not meeting the requirements of the hydrograph are detected. Thus, this report presents the development of a software architecture thanks to the ROS middleware in order to control the navigation of the drone, the acquisition of data as well as the quality control of the latter. After filtering the data in real time using an algorithm, formatting and verifying the consistency of the data, the survey data of the area is then validated. By this way, an DEM⁴.

Mots clés — Ulysse, bathymétrie automatique, USV⁵, filtrage en temps réel, ROS.

ENSTA Bretagne
2, rue François Verny
29200 BREST

-
- 2. Robot Operating System
 - 3. Modèle Numérique de Terrain
 - 4. Digital Elevation Model
 - 5. Unmanned Surface Vehicle

Table des matières

Introduction	1
1 Contexte	2
1.1 Matériel	2
1.2 Enjeux	3
1.3 Objectifs	3
1.4 Stratégie adoptée	4
2 Développement des algorithmes	5
2.1 Réalisation d'un MNT sous ROS	5
2.1.1 Gestion des GNSS	6
2.1.2 Gestion de la SSV	6
2.1.3 Gestion de l'IMU	7
2.1.4 Gestion des données du sondeur multifaisceau	8
2.1.5 Changement de référentiel des données bathymétriques	10
2.2 Contrôle Qualité	11
2.2.1 Détection d'erreurs aberrantes	11
2.2.2 Recouvrement entre les fauchées	13
2.2.3 Cohérence entre les fauchées	15
2.2.4 Densité de sondes	16
2.2.5 Célérité	17
2.2.6 Ordonnancement des algorithmes	17
2.3 Navigation	19
2.3.1 Gestion de la navigation	19
2.3.2 Développement et validation par simulation	21
2.3.3 Commande par waypoint	21
2.3.4 Le contrôleur	23
3 Expérimentation	25
3.1 Préparation de la mission	25
3.1.1 Préparation des lignes de levé	25
3.1.2 Calibration de l'autopilote	28
3.1.3 Alignement de la centrale inertuelle	29
3.1.4 Procédure patch test	30
3.1.5 Configuration de ROS	31
3.2 Exécution de la mission	31
3.3 Résultats et améliorations envisagées	32
3.3.1 Contrôle qualité	32

3.3.2	Modèle numérique de terrain	38
Conclusion		39
Glossaire		i
Bibliographie		ii

Table des figures

2.1	Schéma simplifié de l'architecture des programmes pour l'acquisition d'un MNT	5
2.2	Schéma des différents packages ROS impliqués dans la gestion de l'IMU ⁶	8
2.3	Application du profil de célérité à une mesure du multifaisceau	9
2.4	Schéma des différentes étapes effectuées par le package ROS <i>mbes</i>	10
2.5	Schéma des changements de repères utilisés pour réaliser un MNT	11
2.6	Détection des sondes aberrantes par Moindres Carrés	12
2.7	Schématisation d'un recouvrement de 100% entre les fauchées régulières d'une mission	13
2.8	Étapes du calcul du ratio du recouvrement entre deux fauchées	14
2.9	Application de l'algorithme de recouvrement sur un jeu de données de référence	14
2.10	Illustration de l'analyse de la cohérence entre une fauchée régulière et une traversière	15
2.11	Schéma décrivant l'algorithme du filtre de cohérence	16
2.12	Schéma de fonctionnement de l'algorithme densité	17
2.13	Principe d'ordonnancement du contrôle qualité	19
2.14	Interface du logiciel Mission Planner utilisé lors d'une mission à Guerlédan	20
2.15	Simulation d'une mission à Guerlédan sur le simulateur d'Ardupilot	21
2.16	Angle droit	22
2.17	Angle droit adapté	22
2.18	Demi-tour/Grand angle	22
2.19	Les entrées et les sorties du contrôleur	23
2.20	Gestion des mauvaises mesures	24
2.21	Flowchart du déroulement des missions	24
3.1	Schéma de la largeur de fauchée	26
3.2	Schéma après une approximation de la zone de levé	26
3.3	Lignes de levé sur la zone du lac de Guerlédan	27
3.4	Lignes de levé pour la surface de référence sur la zone du lac de Guerlédan	28
3.5	Aperçu de <i>robot_monitor</i>	31
3.6	Aperçu de <i>robot_monitor</i> lors de la détection d'une erreur	32
3.7	Lignes suivies par Ulysse lors de l'expérimentation	32
3.8	Observation et recherche des paramètres optimaux de l'algorithme de traitement des sondes aberrantes	33
3.9	Vue 3D des lignes régulières acquises lors de l'expérimentation	34
3.10	Différence de profondeur entre les deux lignes	36
3.11	Histogramme des différences des profondeurs	36
3.12	Coupe réalisée sur la surface	37
3.13	Profil vertical des fauchées issues de ROS	37

6. Inertial Measurement Unit

3.14 Profil vertical des fauchées issues de l'acquisition sous Qinsy	37
3.15 MNT réalisé sous ROS visualisé en temps réel via <i>RViz</i>	38

Liste des tableaux

1.1 Capteurs de Ulysse	2
2.1 Résultats obtenus à l'issue de l'algorithme calculant le taux de recouvrement entre les fauchées	15
2.2 Description de l'algorithme de célérité	17
3.1 Largeur de fauchée en fonction de la profondeur au nadir et de la position pour un recouvrement à 100%	27
3.2 Recouvrement obtenu pour l'expérimentation	34

Introduction

Le robot Ulysse est un robot de surface autonome conçu par Texys Marine. Ulysse possède un autopilote qui le rend autonome en navigation. L'ajout sur ce robot d'une centrale inertie et d'un sondeur multifaisceau le rend capable d'effectuer des levés bathymétriques en étant supervisé par un opérateur. Ulysse est très utile pour des levés bathymétriques sur des petits fonds car il possède un faible tirant d'eau. L'objectif est de mettre en place une architecture matérielle et logicielle permettant d'effectuer des levés bathymétriques sur des petits fonds de manière autonome.

La difficulté principale de ce projet réside dans l'acquisition, le contrôle qualité des données, ainsi que la gestion de la navigation, le tout en temps réel. Il a donc été nécessaire d'identifier les différentes tâches du contrôle qualité qui nous permettront d'identifier les erreurs dans les données. L'intérêt de gérer en temps réel la qualité de celles-ci est de pouvoir notifier à l'autopilote la nécessité d'un retour sur une éventuelle fauchée non satisfaisante, avant que la mission ne soit complètement terminée.

Ce rapport vise à détailler la réalisation d'un modèle numérique de terrain, ainsi que les différentes étapes du contrôle qualité et l'impact qu'elles ont sur les algorithmes de navigation. Une attention particulière est apportée à l'utilisation du middleware ROS qui a permis la synergie entre les différents algorithmes que nous avons implémentés.

Chapitre 1

Contexte

1.1 Matériel

Le robot utilisé pour ce projet, dénommé Ulysse, est conçu par Texys Marine. Ce robot, utilisable en environnement côtier comme au large, est équipé d'un multifaisceau, d'une centrale inertie, ainsi que d'un conductimètre. Les principaux capteurs utilisés sont regroupés dans le Tableau 1.1 :

Matériel	Modèle/Référence
Multifaisceau - MBES¹	R2 Sonic 2020 (fréquence et d'autres paramètres sont modifiés directement sur Sonic Control)
Centrale Inertie - IMU (Inertial measurement unit)	Splitbox Ekinox 1
GNSS	Deux boîtiers u-blox ZED-F9H avec correction RTK ²

TABLE 1.1 – Capteurs de Ulysse

Ulysse est doté d'une carte autopilote pour contrôler ses propulseurs. Elle possède :

- différents capteurs en interne (centrale inertie, magnétomètre, GNSS³...).
- plusieurs ports de communication pour interagir notamment avec le planificateur de mission et ROS ainsi que pour commander les actionneurs.

Ulysse dispose également de deux NUC⁴, un avec un système d'exploitation Windows et un avec Linux. Le premier sert à l'utilisation des logiciels hydrographiques comme Qinsy permettant l'acquisition des données hydrographiques mais également pour RTKLIB⁵, une bibliothèque utilisée pour la correction des données GNSS. Le NUC Linux nous a servis à mettre en place l'architecture middleware ROS. L'un de nos objectifs a donc été de transposer toutes les fonctions effectuées sur le NUC Windows vers le NUC Linux afin de limiter l'architecture matérielle

3. Géolocalisation et Navigation par un Système de Satellites

4. Next Unit of Computing

5. Real Time Kinematic Library

et de centraliser l'ensemble des éléments sur un seul système d'exploitation Linux, qui a de plus l'avantage d'être libre de droits.

1.2 Enjeux

L'objectif initial de ce projet est de rendre Ulysse autonome, l'USV, afin d'effectuer la réalisation de levés hydrographiques au niveau du lac de Guerlédan, notre lieu d'étude. L'utilisation d'USV est utile lors de la réalisation de levés sur des zones de petits fonds, endroits où les bateaux ne peuvent sonder à cause de leur tirant d'eau trop important. Depuis 2015, après la vidange du barrage EDF⁶ construit sur le lac, l'entreprise a des demandes hydrographiques au sein de la zone. En effet, de nouveaux modèles numériques de terrain sont nécessaires à cause du mouvement sédimentaire important. Le volume d'eau est une donnée nécessaire à EDF afin de faire fonctionner au mieux les turbines présentes au niveau du barrage. C'est pourquoi la réalisation de modèles numériques de terrain de qualité est primordiale.

1.3 Objectifs

Les avantages de l'utilisation de ce type de système sont la possibilité de sonder les zones en petits fonds, mais aussi du gain de temps. La mobilisation des équipements utilisés lors d'un levé est en effet fastidieuse et nécessite beaucoup d'installation de matériel, comme la centrale inertie, les GNSS et le montage du sondeur multifaisceau. De plus, toute la synchronisation entre les capteurs doit être effectuée avant le début de chaque levé. Le but de cette automatisation est de pouvoir créer un ensemble d'algorithmes permettant de repasser sur les lignes ne répondant pas aux critères exigés par l'hydrographe, c'est-à-dire des lignes dont les données sont acceptables pour être exploitables. Pour ce faire, deux solutions étaient envisageables :

1. Utiliser le logiciel Qinsy et utiliser les fichiers log issus de celui-ci pour faire un contrôle qualité. Utiliser ROS pour la gestion du retour sur zone.
2. Développer à nouveau les fonctionnalités du logiciel Qinsy sous ROS et ainsi s'affranchir de ce logiciel.

La première solution évite un développement ambitieux qui peut être fastidieux, mais celle-ci est moins souple en termes de champ des possibles et de paramétrage. De plus, elle nécessite deux architectures matérielles (Windows et Linux) et logicielles (Qinsy et ROS). La deuxième solution permet quant à elle d'avoir un contrôle de l'ensemble des traitements effectués, et donc une configuration adaptée aux exigences. Cela permet également de n'avoir qu'une seule architecture matérielle et logicielle. En revanche, cela nécessite beaucoup de temps de développement et de gestion de la chaîne d'acquisition des données.

6. Électricité De France

1.4 Stratégie adoptée

Dans le but d'être entièrement maître des opérations et pour simplifier le déroulé du levé automatisé, nous nous sommes penchés vers la solution qui permet de s'affranchir du logiciel Qinsy jusqu'à nécessaire pour l'acquisition des données. En effet, si celui-ci est destiné à faciliter la mise en œuvre manuelle d'un levé, ce logiciel externe peut en réalité être un frein au projet d'automatisation du levé puisque l'automatisation vise à centraliser plusieurs fonctionnalités (contrôle de la trajectoire du robot, gestion des événements lors l'acquisition des données comme la perte GNSS, analyse des sondes acquises...), pas toujours disponibles sur tous les types d'architectures matérielles comme Windows ou Linux.

De plus, toutes les fonctionnalités offertes par Qinsy ne sont pas nécessaires à notre projet : elles alourdissent et ralentissent le processus d'acquisition et de contrôle des données, ce qui est problématique dans l'optique de la réalisation d'un contrôle qualité en temps réel. Nous avons ainsi choisi la seconde solution décrite. L'ensemble des solutions techniques apportées seront donc implémentées sous ROS.

Chapitre 2

Développement des algorithmes

L'implémentation sous ROS permet d'utiliser l'ensemble des capteurs à disposition afin de réaliser un lot de données bathymétriques exploitable en temps réel. ROS permet également d'effectuer le contrôle qualité des données acquises afin de les valider ou non. Enfin, le retour sur zone peut être directement relié au résultat de ces filtres. Ceci fournit donc un ensemble fonctionnel et cohérent au vu de l'objectif de ce projet. Nous allons ainsi décrire dans cette partie l'ensemble des différents algorithmes développés permettant de mener à bien une expérimentation. Enfin, il est important de noter que nous avons développé nos solutions techniques en utilisant la plateforme collaborative GitHub¹.

2.1 Réalisation d'un MNT sous ROS

Cette partie a pour but d'expliquer l'ensemble des programmes nécessaires à la fabrication d'un MNT sous ROS à partir des capteurs installés dans Ulysse.

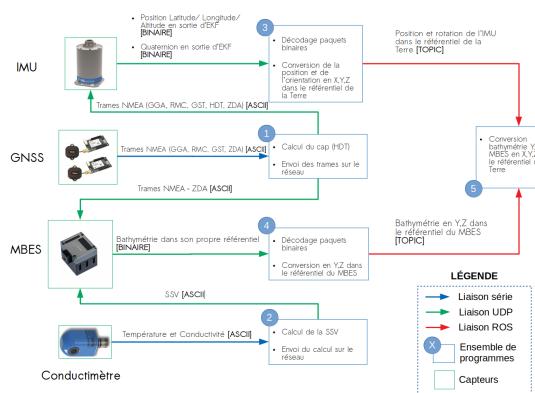


FIGURE 2.1 – Schéma simplifié de l'architecture des programmes pour l'acquisition d'un MNT

1. <https://github.com/KevinBdn/ULYSSE>

La Figure 2.1 schématise l'implémentation d'une architecture simplifiée. Il explicite les différents éléments nécessaires à la création d'un MNT, ainsi que les liens existant entre eux. Ainsi, nous allons décrire les cinq ensembles de programmes présents dans le schéma. Il s'agit de scripts Python implémentés sous ROS.

2.1.1 Gestion des GNSS

Le but de ces capteurs est de fournir des positions Latitude/Longitude/Altitude au robot afin qu'il puisse se localiser dans l'espace. L'une des antennes est positionnée à l'avant du robot, la seconde à l'arrière. Les GNSS fournissent également une base temporelle afin de synchroniser l'ensemble des appareils renvoyant des informations utiles variant dans le temps.

L'utilisation de deux GNSS permet de calculer le cap du robot, essentiel à la centrale inertielle. Pour ce faire, nous avons repris un code Python déjà existant sur Ulysse, que nous avons implémenté sous ROS sous le package *ublox_gps*. Ce code consiste à calculer le cap à partir des deux positions des antennes. Soit (λ_1, ϕ_1) le couple de coordonnées Longitude/Latitude du récepteur arrière et (λ_2, ϕ_2) celui du récepteur avant. On a alors : $\Delta_\lambda = (\lambda_2 - \lambda_1)$ On définit également (x, y) tel quel : $x = \sin(\Delta_\lambda).\cos(\phi_2)$ $y = \cos(\phi_1).\sin(\phi_2) - \sin(\phi_1).\cos(\phi_2).\cos(\Delta_\lambda)$ On obtient ainsi θ le cap du robot : $\theta = \arctan2(x, y)$ On ajuste ensuite cette valeur grâce à un offset désignant l'angle de biais entre l'axe des antennes GNSS et celui de la centrale inertielle.

La donnée une fois calculée est mise sous forme de trame NMEA² de type *HDT*. Les trames *GGA*, *RMC*, *GST*, *ZDA* et *HDT* sont par la suite envoyées sur le réseau en UDP³. Elles pourront alors être traitées par la centrale inertielle et le multifaisceau.

2.1.2 Gestion de la SSV

La célérité du son dans l'eau est un paramètre important pour le positionnement des mesures acquises par le multifaisceau. L'équation de la célérité du son dans l'eau est fonction de trois paramètres : la salinité, la température et la pression. Nous déterminons la valeur de cette célérité en surface qui nous servira à estimer celle de toute la colonne d'eau. Pour la surface de l'eau, la pression est connue car égale à celle de la pression atmosphérique, la température est mesurable et la salinité est obtenue à partir d'une mesure de conductivité. Une approximation de la célérité du son dans l'eau est la suivante :

$$c = 1449.2 + 4.6T - 0.055T^2 + 0.00029T^3 + (1.34 - 0.010T)(S - 35) + 1.58 * 10^{-6}p$$

c est la célérité du son dans l'eau en $m.s^{-1}$, T est la température en $^{\circ}C$, S la salinité en psu et p la pression en pascal. Cette formule est obtenue à partir de l'équation de l'état de l'eau de la mer (IES 80) [3].

La salinité est obtenue par l'échelle pratique de la salinité dont la formule est la suivante :

$$S = 0.0080 - 0.1692K^{1/2} + 25.3853K + 14.0941K^{3/2} - 7.0261K^2 + 2.7081K^{5/2}$$

S est la salinité et K la conductivité en $S.m^{-1}$.

2. National Marine Electronics Association

3. User Datagram Protocol

Le robot est donc doté d'une sonde de température et de conductivité, de laquelle on récupère les valeurs du capteur en ASCII. Ces valeurs nous permettent d'en déduire, par les formules précédentes, la salinité de l'eau puis la célérité du son dans l'eau proche de la surface SSV⁴. La célérité du son est nécessaire par la suite pour la mesure de la bathymétrie à partir des données brutes du multifaisceau. On communique cette valeur de vitesse sur le réseau via un socket UDP. Le script python permettant l'ensemble de ce procédé a été implémenté sous un package ROS nommé *ssv_computing*.

2.1.3 Gestion de l'IMU

La centrale inertie est connectée au réseau d'Ulysse et envoie ses données via un protocole UDP. La récupération et l'interprétation de ces données sur le NUC s'effectue via le package ROS *sbg_ros_driver*, initialement développé par l'ENSTA Bretagne puis repris par l'entreprise SBG Systems.

Ainsi plusieurs types d'information sont accessibles selon le manuel de référence [6] : le temps UTC, la latitude, la longitude, l'altitude par rapport au géoïde, l'ondulation associée, les angles de gîte, d'assiette et de cap, ainsi que tous les écarts-types associés aux différentes mesures obtenus grâce à un filtre de Kalman étendu (**Extended Kalman Filter**).

Voici celles que nous exploitons :

- La position géographique du porteur à l'issue de l'EKF.
- Un quaternion décrivant l'atitude à l'issue de l'EKF.

Ayant à disposition ces données, nous pouvons alors projeter la position du porteur.

Position du porteur

La position du porteur dans un repère plan est effectuée via un second package ROS : *geonav_transform*.

Afin de pouvoir l'utiliser, il nous faut au préalable transformer les messages non standards du package *sbg_ros_driver* en message standard exploitable par *geonav_transform*. Pour cela la première étape importante est la conversion des informations du système NED⁵ de l'IMU en ENU⁶. La seconde consiste à combiner les informations de navigation et le quaternion en un seul message odométrique (*Odometry*). Ces deux étapes sont regroupées au sein d'un package ROS que nous avons développé sous le nom de *sbg_to_odom*.

Une fois la position standardisée, elle peut alors être projetée grâce au package *geonav_transform*. La projection utilisée par le paquet est l'UTM⁷, basée sur le système géodésique de référence WGS84 utilisant l'ellipsoïde IAG-GRS80. Afin de déterminer le fuseau dans lequel sera effectuée la projection, nous utilisons un point de référence (*datum*) qui sera désigné comme le point d'origine $(X, Y) = (0, 0)$ de la projection. La valeur Z caractérise quant à elle la hauteur ellipsoïdale.

-
- 4. Sound Surface Velocity
 - 5. North East Down
 - 6. East North Up
 - 7. Universal Transverse Mercator

Schéma récapitulatif

La figure 2.2 résume sous forme schématique la description ci-dessus.

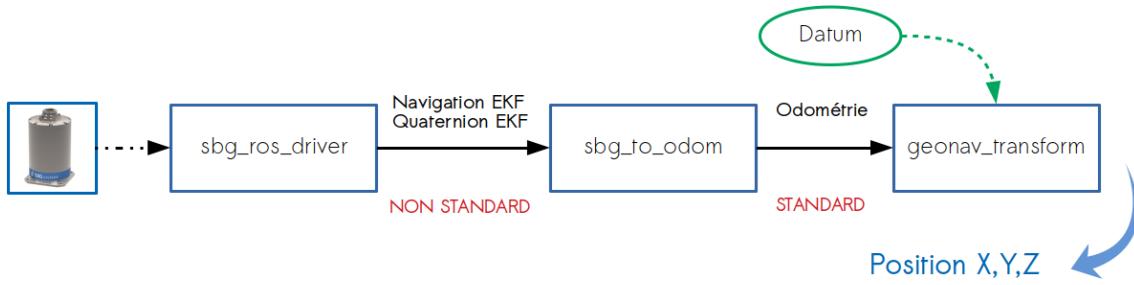


FIGURE 2.2 – Schéma des différents packages ROS impliqués dans la gestion de l'IMU

A l'issue de cette étape, nous disposons donc de la position du porteur projetée dans un référentiel terrestre local.

2.1.4 Gestion des données du sondeur multifaisceau

Le multifaisceau étant relié au réseau d'Ulysse via un protocole UDP, il est alors possible d'exploiter ses données sous ROS. Nous avons donc développé un package nommé *mbes* qui a pour charge de réaliser les différentes opérations décrites ci-dessous, par ordre chronologique d'exécution.

Récupération des données acquises par le sondeur

Il s'agit en premier lieu de récupérer les données brutes du sondeur multifaisceau R2Sonic 2020, sans passer par le logiciel Qinsky. Le sondeur émet continuellement des données binaires au cours de l'acquisition via des paquets UDP. Il est capable d'émettre plusieurs paquets comprenant chacun différents types de données comme des informations sur la colonne d'eau, une image acoustique... Pour notre projet, nous nous intéressons au paquet rassemblant les informations bathymétriques. Nous avons donc du gérer ce flot d'informations en exploitant seulement les données nécessaires à la construction de nos points de sondes. Pour ce faire, nous avons créé un client UDP venant lire ces données binaires lorsqu'elles sont disponibles. Ensuite, nous décodons les paquets reçus en s'appuyant sur le manuel d'opérations officiel du sondeur R2Sonic2020 [4]. Enfin, nous y récupérons les informations concernant les temps aller-retour entre l'émission d'un faisceau et la réception de l'écho, ainsi que les angles qui leurs sont associés. Une fois ces données récupérées, nous pouvons les exploiter.

Application du profil de célérité dans le repère du porteur

Lors de l'acquisition d'un nouveau ping par l'étape précédente, il est nécessaire de lui appliquer le profil de célérité (SVP⁸) de la colonne d'eau correspondante. L'application du profil

8. Sound Velocity Profile

de célérité consiste à une déviation de chacun des faisceaux en fonction de la célérité de surface et de chacune des célérités des différentes couches du profil.

Pour ce faire, on considère i_0 un faisceau incident fourni par le multifaisceau, T le temps de mesure aller-retour du faisceau et v_0 la SSV mesurée. Pour chaque couche k du profil de célérité, de célérité v_k et d'épaisseur D_k , on calcule le rayon diffracté i_{k+1} grâce aux lois de Snell-Descartes :

$$n_k \cdot \sin(i_k) = n_{k+1} \cdot \sin(i_{k+1})$$

$$\Rightarrow i_{k+1} = \arcsin\left(\frac{n_k}{n_{k+1}} \cdot \sin(i_k)\right)$$

Or :

$$\frac{n_k}{n_{k+1}} = \frac{v_{k+1}}{v_k}$$

Donc :

$$i_{k+1} = \arcsin\left(\frac{v_{k+1}}{v_k} \cdot \sin(i_k)\right)$$

On calcule également le temps parcouru dans cette couche t_k :

Soit d_k la distance parcourue par le faisceau dans la couche k , on a alors :

$$d_k = \frac{D_k}{\cos(i_k)}$$

Or :

$$t_k = \frac{d_k}{v_k}$$

Donc :

$$t_k = \frac{D_k}{v_k \cdot \cos(i_k)}$$

Ce procédé s'arrête lorsque les temps cumulés t_k sont égaux à la moitié du temps T .

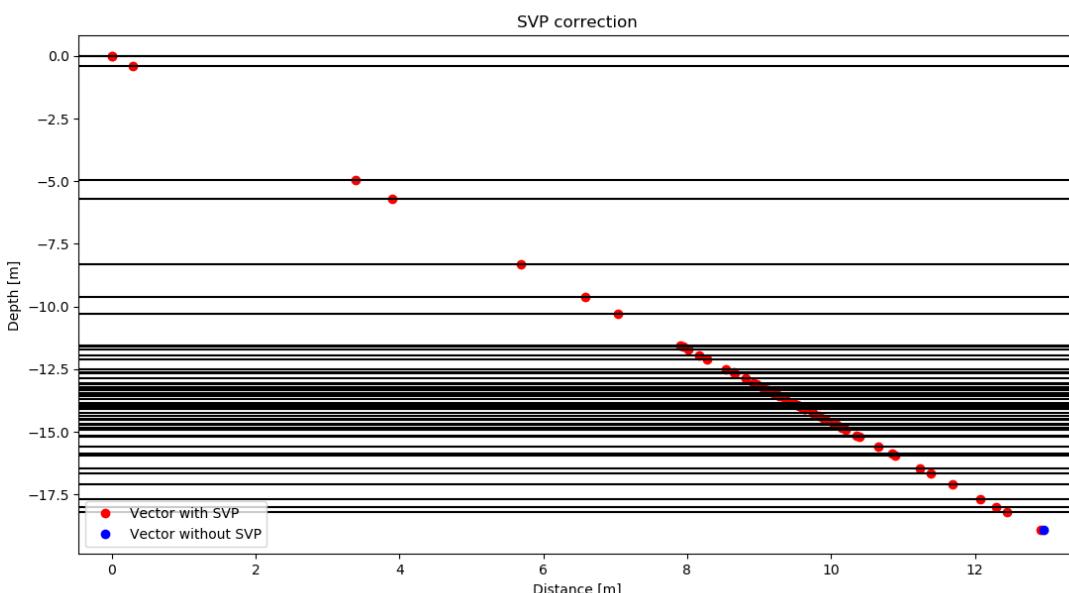


FIGURE 2.3 – Application du profil de célérité à une mesure du multifaisceau

La Figure 2.3 est une illustration de l'algorithme décrit ci-dessus. Pour chaque couche du SVP, le point du faisceau entrant et tracé en rouge. Les lignes noires représentent les différentes couches fournies par la SVP. On observe bien une légère déviation de la position finale par rapport à la position calculée sans l'application de la SVP, illustrée en bleu.

Transmission des données du multifaisceau

Une fois que les données brutes d'un ping sont récupérées sur le réseau et que nous leur avons appliqué le profil de célérité de la colonne d'eau correspondante, elles sont ensuite transmises dans un message ROS sous la forme d'un nuage de points (*PointCloud*). La Figure 2.4 résume sous forme imagée les étapes précédemment décrites.



FIGURE 2.4 – Schéma des différentes étapes effectuées par le package ROS *mbes*

Ce message ROS pourra par la suite être exploité par d'autres algorithmes dont celui de la réalisation du MNT.

Ainsi à l'issu de cette partie nous disposons des sondes dans le référentiel du porteur sous la forme de nuage de points.

2.1.5 Changement de référentiel des données bathymétriques

Le changement de référentiel des sondes se fait exclusivement via ROS. Les données récupérées à partir du message *PointCloud* sont transformées dans le référentiel de l'IMU. Cette transformation utilise le principe de *tf* sous ROS. En décrivant la transformation existante entre le sondeur multifaisceau et la centrale inertie, on peut alors demander à ROS de transformer les sondes dans le référentiel de l'IMU préalablement projetée dans un repère local. Ce processus permet ainsi de réaliser des MNT. La Figure 2.5 représente schématiquement les différentes transformées appliquées pour la réalisation d'un MNT.

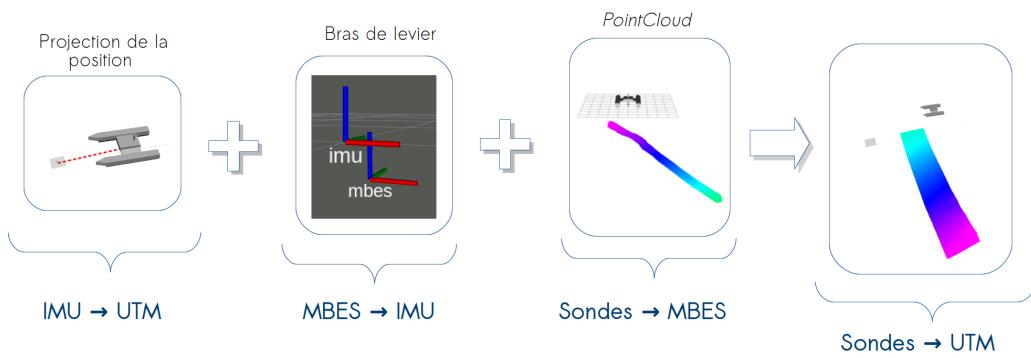


FIGURE 2.5 – Schéma des changements de repères utilisés pour réaliser un MNT

Ainsi les différents algorithmes présentés dans cette partie permettent la réalisation d'un MNT brut en temps réel. Ce dernier est par la suite soumis au contrôle qualité décrit dans la partie suivante.

2.2 Contrôle Qualité

Le but de l'hydrographe est de fournir à l'issue d'un levé un MNT de qualité et exempt d'erreurs. Le MNT brut obtenu à l'issue de l'acquisition des données ne peut donc être le produit final de l'étude. Ainsi, les mesures doivent être traitées. Le contrôle de la qualité des données en temps réel va donc se révéler comme l'un des challenges principaux de l'automatisation du levé hydrographique. Beaucoup de possibilités existent afin de vérifier que les données acquises sont exploitables pour la réalisation d'un MNT valide. Cependant, il ne faut pas oublier que ces contrôles nécessitent des calculs qui peuvent rapidement devenir complexes selon les objectifs fixés : le défi est donc d'implémenter des outils à la fois performants et rapides en terme d'exécution afin de respecter la dimension du temps réel.

Nous choisirons pour ce projet de développer des algorithmes de type filtre. Le principe est de comparer les résultats obtenus à l'issue de ces filtres aux exigences souhaitées par l'opérateur. Cette comparaison permettra alors de lever une alerte si le résultat ne répond pas aux critères exigés, et ainsi de rendre possible un retour sur zone pour refaire les mesures avant la fin de la mission.

2.2.1 Détection d'erreurs aberrantes

Avant de réaliser tout contrôle des mesures, il faut s'affranchir de la majorité des erreurs ponctuelles du lot de données sur lequel les algorithmes sont appliqués, c'est-à-dire invalider les sondes aberrantes qui pourraient impacter la validation d'un ensemble de mesures. Ainsi, un algorithme de détection automatique d'outliers est implémenté.

Cet algorithme repose sur la méthode des moindres carrés [5], dont le modèle implémenté est linéaire :

$$Y + V = A * X$$

Y représente les mesures (soit les valeurs de nos sondes), V correspond aux résidus, A est la matrice modèle et X représente les inconnues (soit les sondes estimées d'après nos mesures).

Le principe est donc de considérer un ensemble de mesures et de détecter celles qui s'éloignent de la tendance. Ainsi, c'est le vecteur des résidus qui nous intéresse.

Implémentée au premier ordre, la technique des moindres carrés ajuste un plan sur l'ensemble des mesures considérées. Les résidus issus de l'algorithme sont ensuite comparés aux exigences, permettant finalement de valider ou non chaque mesure.

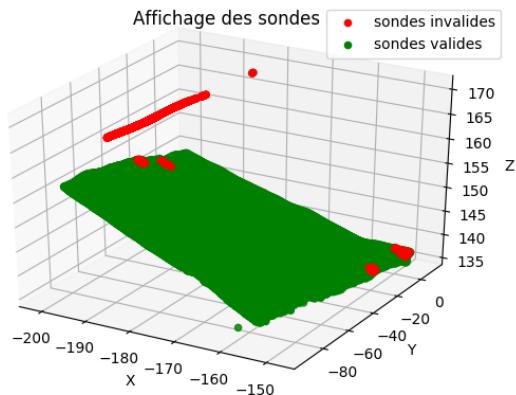


FIGURE 2.6 – Détection des sondes aberrantes par Moindres Carrés

Il existe donc plusieurs paramètres à déterminer. D'abord, le nombre de mesures qu'il faut considérer pour pouvoir y appliquer la méthode des moindres carrés, qui correspond en fait à la taille des paquets de sondes analysées. En effet, il faut trouver la taille la plus adaptée au lot de données reçue. Le modèle implémenté est linéaire, une surface plane est donc considérée : si le lot de données correspond à une fauchée de morphologie complexe, il faudra prendre garde à ne pas choisir un paquet de taille trop élevée qui invaliderait les sondes s'éloignant du plan considéré, bien qu'elles représentent une élévation ou une anfractuosité réelle. A contrario, si la taille du paquet choisi est trop petite, l'algorithme ne détecterait aucune valeur aberrante par manque d'information. Outre la question de la morphologie de la fauchée, il faut de plus considérer le nombre total des sondes acquises pour la fauchée afin de déterminer un ordre de grandeur de la taille du paquet.

Comme il s'agit d'un filtre, il faut également déterminer un seuil régissant les résultats du filtre, qu'on note α . C'est en fait la comparaison des résidus des moindres carrés à ce seuil qui validera ou non la sonde.

Enfin, l'algorithme invalide les sondes une à une. Dans l'optique de l'automatisation du levé, il faudrait détecter l'invalidité de la fauchée lorsqu'elle ne répond pas aux exigences. On intègre donc un nouveau paramètre, correspondant au pourcentage de sondes invalides toléré.

Ainsi, trois paramètres sont nécessaires à l'algorithme, et dans un souci d'adaptabilité de celui-ci à tout type de levé, ces paramètres sont déterminés par l'opérateur au préalable. La taille du paquet de sondes et α , bien que cruciaux vis-à-vis des résultats de filtrage, nécessitent une connaissance a priori de la bathymétrie de la zone étudiée. Le pourcentage de sondes invalides toléré dépend quant lui des exigences associées à la qualité du levé.

2.2.2 Recouvrement entre les fauchées

Le recouvrement entre les fauchées est un élément essentiel à la qualité d'un levé. Tout hydrographe averti planifie sa mission de sorte que la zone étudiée soit sondée plusieurs fois pour obtenir une redondance d'information entre les fauchées régulières. Cette redondance est ensuite utilisée pour effectuer les tests de validation des données. La figure 2.7 montre le principe d'un recouvrement de 100% entre les fauchées :

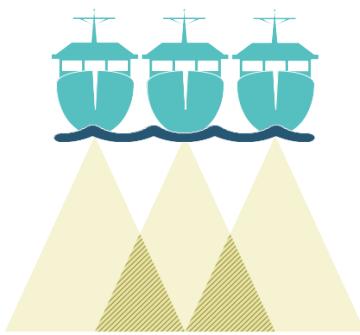


FIGURE 2.7 – Schématisation d'un recouvrement de 100% entre les fauchées régulières d'une mission

Le recouvrement entre les fauchées se calcule, et doit a priori être assuré si le profil de la mission a bien été défini. Cependant, l'emprunte de chaque fauchée diffère selon la bathymétrie du terrain : si celle-ci est méconnue, il est possible que le recouvrement attendu entre les fauchées ne soit pas satisfait. En automatisant le levé, il est donc important de vérifier si ce recouvrement est assuré.

L'algorithme implémenté repose sur une analyse géométrique. Le principe est de détourer l'aire de deux fauchées voisines, et de calculer le ratio de la zone où les deux aires se chevauchent sur la totalité de l'aire d'une des fauchées. Pour arriver à ce résultat, plusieurs étapes sont nécessaires :

1. Détourage des aires des fauchées

Il est réalisé en traçant un polygone convexe englobant l'ensemble des sondes de la fauchée.

2. Rotation des aires

Pour pouvoir comparer les deux fauchées, il est nécessaire de les redresser dans un repère leur étant propre et commun. Pour ce faire, on calcule le cap moyen des premières sondes d'une fauchée, qui sera donc celle de référence pour la rotation, et on redresse les polygones représentant les aires de l'angle trouvé.

3. Intersection des aires

Elle est facilement calculée en compilant l'intersection des deux polygones.

4. Redéfinition de la zone d'étude

Les fauchées comparées ne font pas toujours la même longueur. Ainsi, avant de faire le ratio de l'intersection sur l'aire totale, il faut redéfinir cette aire totale en la découplant selon la longueur de la fauchée à laquelle on la compare. Si cette étape n'est pas réalisée, la comparaison est biaisée car on compte une partie qui n'a pas été sondée, et qui ne doit donc pas être traitée par l'algorithme. Pour redéfinir la zone d'étude, on utilise les bornes du polygone représentant

l'intersection des deux aires, pour ne prendre donc que les parties communes des deux fauchées.

La figure 2.8 illustre les quatre étapes nécessaires pour le calcul du recouvrement de deux fauchées :

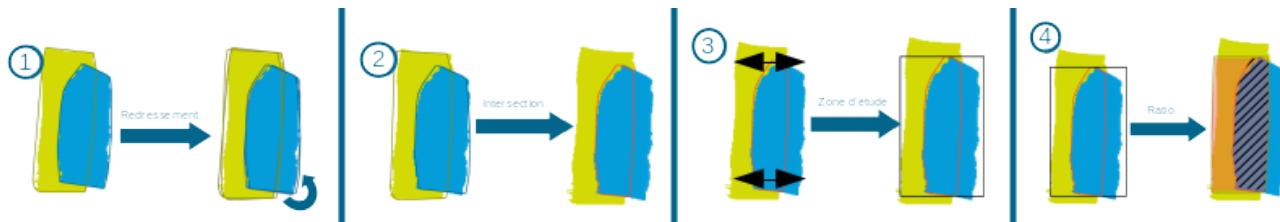


FIGURE 2.8 – Étapes du calcul du ratio du recouvrement entre deux fauchées

Il faut prendre en compte le fait que les fauchées ne représentent pas les mêmes bathymétries du côté bâbord et du côté tribord au sein d'elles mêmes. Or, le recouvrement est calculé sur les côtés latéraux des fauchées : ainsi, le recouvrement d'une fauchée sur une autre n'est pas égal à l'inverse. Il faut donc penser à calculer le recouvrement sur les deux bords et selon chaque fauchée considérée. Ainsi, on vérifie le recouvrement de fauchées régulières deux à deux, mais également le recouvrement total d'une fauchée en additionnant les deux ratios la concernant. Une alerte est levée lorsque le total des ratios pour une fauchée est inférieur au recouvrement souhaité.

Cet algorithme dépend de deux paramètres : l'un représentant le recouvrement souhaité par l'opérateur, l'autre définissant le seuil de tolérance par rapport au résultat espéré. Encore une fois, ces deux paramètres sont adaptables à l'étude et sont de ce fait fixés par l'hydrographe.

Afin de vérifier la cohérence des résultats obtenus, l'algorithme est testé sur un ensemble de fauchées de référence, pour lesquelles le design de la mission est connu. Le profil du levé a été dessiné tel que le recouvrement entre chaque fauchée doit respecter le taux de 100%. Ainsi, notre algorithme devrait nous renvoyer une valeur autour de ce taux. La figure 2.9 présente le contexte du jeu de données de référence sur lequel l'algorithme est appliqué :

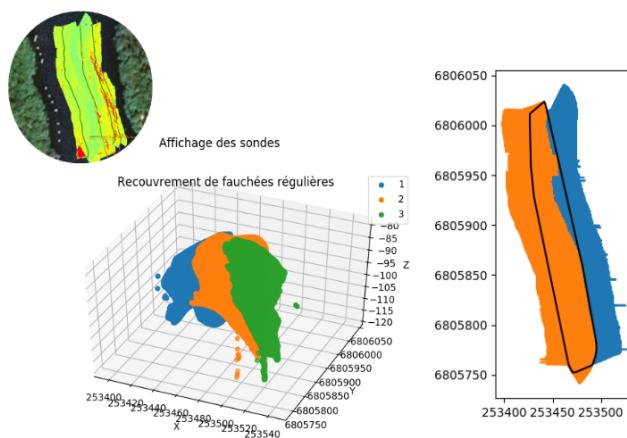


FIGURE 2.9 – Application de l'algorithme de recouvrement sur un jeu de données de référence

Numéro fauchée	1 sur 2	2 sur 1	2 sur 3	3 sur 2	RECOUVREMENT TOTAL 2
Taux de recouvrement	52.13%	55.36%	69.65%	54.04%	106.17%

TABLE 2.1 – Résultats obtenus à l’issue de l’algorithme calculant le taux de recouvrement entre les fauchées

Le tableau 2.1 montre des résultats tout-à-fait satisfaisants puisque l’algorithme calcule un recouvrement de 106%, conformément à ce qui était attendu.

2.2.3 Cohérence entre les fauchées

Le filtre de cohérence a pour objectif d’attribuer un statut invalide à une fauchée si elle n’est pas cohérente avec le reste du jeu de données.

Il établit ses calculs lorsque les données d’une fauchée régulière et d’une traversière sont disponibles. Ainsi, si en comparant les deux fauchées il existe un biais de profondeur ϵ comme illustré par la figure 2.10, il sera possible de déterminer quelle fauchée est incohérente à l’aide d’une autre fauchée, mais cela se fera uniquement en temps différé.

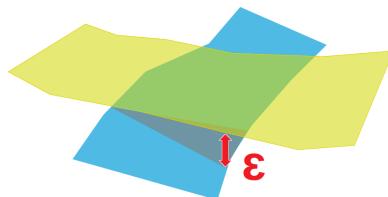


FIGURE 2.10 – Illustration de l’analyse de la cohérence entre une fauchée régulière et une traversière

Le script Python de ce test se décompose en plusieurs étapes et est illustré par le schéma 2.11 :

1. Entrée de l’algorithme

L’algorithme prend en entrée deux fauchées : une régulière et une traversière.

2. Discréétisations

Deux grilles ayant la même origine et de même taille sont créées pour chacune des fauchées.

3. Moyennage des profondeurs

En chaque case de la grille, il y a plusieurs valeurs de profondeurs. Il est donc nécessaire de moyennner.

4. Différenciation

La différence entre les valeurs de profondeur des deux fauchées est effectuée.

5. Seuillage et Alerté

Lorsque la différence de profondeur en une case de la grille est supérieure à un seuil d’invalidation, il faut affecter le statut d’invalidité à la fauchée. Ce seuil dépend de l’exigence du levé demandé par l’hydrographe et également de la topographie du terrain.

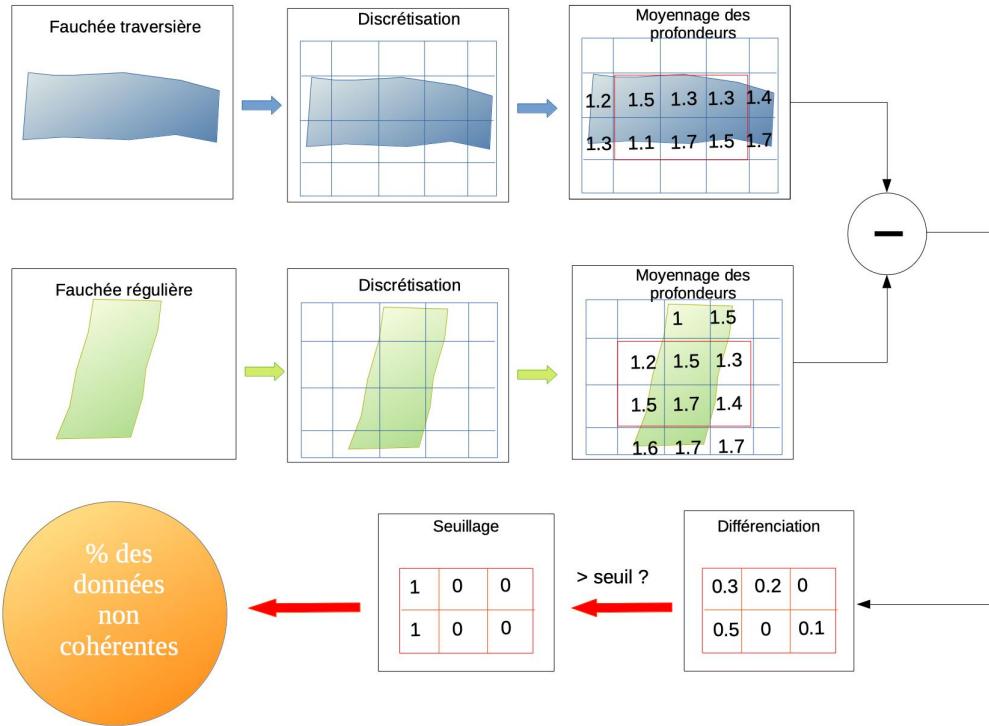


FIGURE 2.11 – Schéma décrivant l'algorithme du filtre de cohérence

Deux paramètres sont importants dans la réalisation du filtre : la résolution et le seuil de rejet. La résolution de grille doit être adaptée à la quantité de données en fonction de l'avancée du porteur. En effet, si la résolution choisie pour la réalisation de matrices contenant les données de sondes est trop faible, il y aura des zones sans information dans le modèle numérique de terrain. Si au contraire la résolution est importante, le test sera moins robuste car il y aura moins de points dont la position dans la grille est commune pour effectuer la différence.

2.2.4 Densité de sondes

L'algorithme implémenté pour le filtrage de la densité travaille sur une fauchée de type quelconque. Son schéma apparaît en Figure 2.12. À partir du cap moyen suivi au long de la fauchée, l'algorithme commence par redresser celle-ci afin de ramener son axe à la verticale. Puis il va créer une grille de résolution choisie par l'utilisateur, s'appuyant sur cette verticale, qui s'étend jusqu'aux valeurs extrêmes de la fauchée en hauteur et en largeur. Il calcule aussi la largeur du pied de faisceau pour chaque sonde en fonction de la profondeur et de la largeur du faisceau. Puis l'algorithme calcule le nombre de sondes qui tombent dans chaque maille de la grille. Enfin, l'algorithme calcule le pourcentage de mailles de densité inférieure à la densité souhaitée par l'utilisateur, et le compare au seuil d'acceptation également défini par ce dernier. Pour cela, nous tenons compte par défaut du fait qu'il y aura un recouvrement de 100% entre les fauchées. Ainsi, seules les mailles de densité supérieure à la moitié de la densité imposée par l'opérateur sont conformes. Par ailleurs, c'est l'algorithme de recouvrement développé en section 2.2.2 qui vérifiera que ce critère est bel et bien respecté.

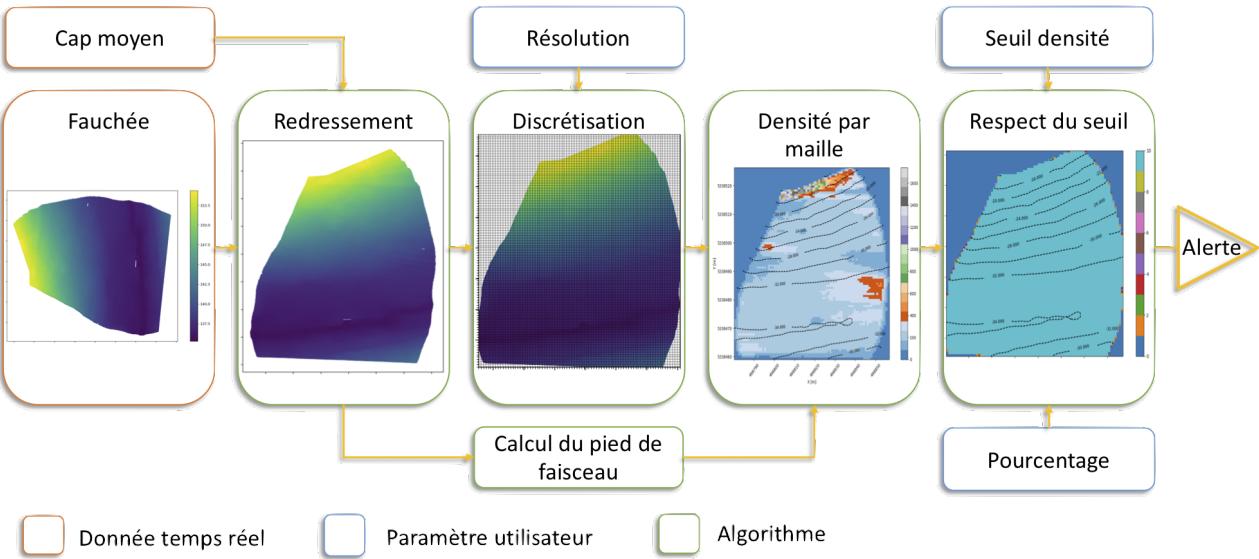


FIGURE 2.12 – Schéma de fonctionnement de l'algorithme densité.

2.2.5 Célérité

L'algorithme implémenté sous Python est décrit dans le tableau 2.2. La valeur de référence utilisée correspond à la valeur du profil de célérité en surface. Elle sera alors comparée à la valeur de la célérité de surface transmise au fur et à mesure du levé par la structure ROS. Ainsi, si la différence entre ces deux valeurs est supérieure à un seuil, une alerte est transmise au contrôleur.

Entrée	SVP : profil de célérité (profondeur en m célérité en m/s) SSV : célérité de surface en m/s
Paramètre	Seuil : déterminé par l'hydrographe, dépend de la différence entre les deux entrées
Sortie	Alerte lorsque la célérité de surface est trop éloigné par rapport à la célérité donnée par le profil.

TABLE 2.2 – Description de l'algorithme de célérité

2.2.6 Ordonnancement des algorithmes

Une fois les algorithmes nécessaires au contrôle qualité choisis et implémentés, il s'agit d'ordonnancer ces différents outils afin de les faire communiquer de manière judicieuse. Il faut donc réfléchir à l'instant et à l'ordre auxquels ils seront appelés afin d'optimiser le contrôle qualité en temps réel.

A l'exception de l'algorithme de célérité, chacun travaille sur une ou plusieurs fauchées complètes. Les sondes acquises par le sondeur sont reçues en un flux continu d'après le lot de données acquis en temps réel. La première étape est donc de stocker ces sondes ligne par ligne. Un algorithme effectuant cette tâche a été implémenté, il est assimilé à un gestionnaire des fauchées. Un contrôleur, explicité dans la partie 2.3.4, permet au gestionnaire de détecter une ligne grâce à un signal de début et de fin de la fauchée. Ainsi, les lignes du levé sont enregistrées sans les sondes acquises pendant les virages, car généralement inexploitables pour la réalisation d'un MNT.

Une fois les sondes stockées fauchée par fauchée, le gestionnaire fait intervenir l'algorithme de détection des mesures aberrantes, présenté en 2.2.1. En effet, tout contrôle qualité doit être effectué sur des fauchées exemptes d'outliers, qui peuvent impacter les résultats des filtrages. Ainsi, à chaque fauchée enregistrée, l'algorithme des outliers est appelé. A l'issue de celui-ci, les sondes traitées sont écrites dans des fichiers texte dans lesquels on retrouve le statut de validité de chaque mesure. Ces fichiers sont de plus nommés de façon à ce qu'apparaisse le type de la fauchée enregistrée, défini au préalable par l'opérateur au début de la mission. Cette convention de nommage est particulièrement utile pour le bon déroulement des autres algorithmes de filtrage.

Lorsque les sondes aberrantes sont traitées, les autres algorithmes peuvent être lancés parallèlement. Cependant, chacun fonctionne avec un type et un nombre de fauchées particuliers. Pour gérer cela, les algorithmes stockent chacun les noms des fichiers dans une liste, et dès que celle-ci contient la configuration nécessaire à un algorithme, celui-ci est lancé.

C'est ensuite au contrôleur de navigation de gérer les différentes alertes éventuellement levées par chaque algorithme. La figure 2.13 résume schématiquement le principe d'ordonnancement du contrôle qualité :

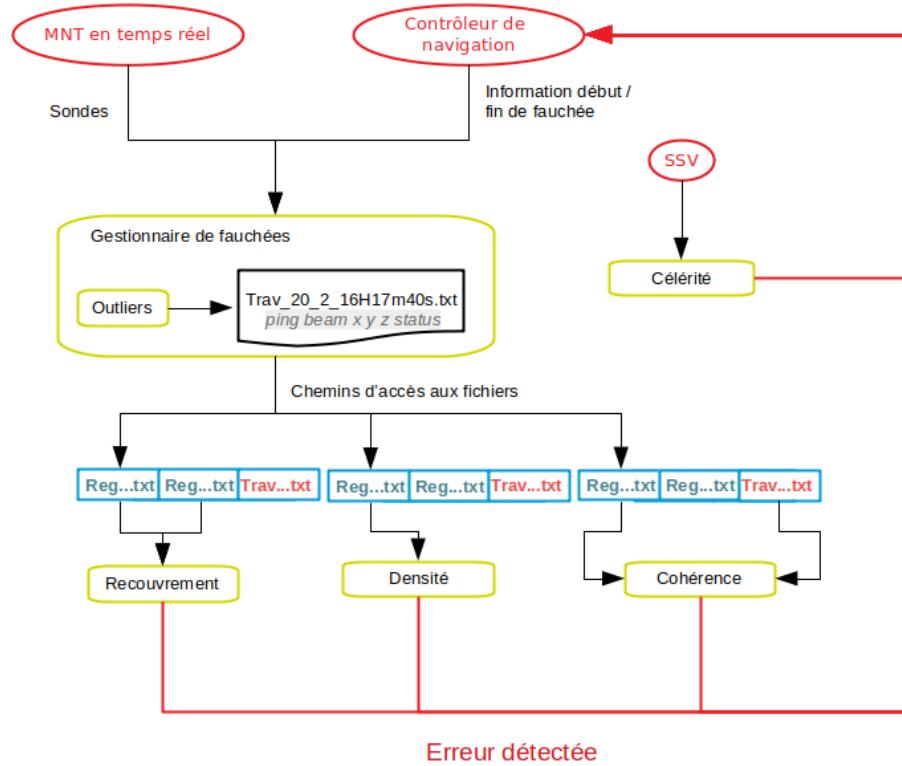


FIGURE 2.13 – Principe d'ordonnancement du contrôle qualité

2.3 Navigation

La partie navigation consiste à expliciter la manière dont le retour sur zone et la gestion de la navigation ont été implémentés à l'aide du package ROS *mavros*, permettant la communication avec l'autopilote via le protocole MAVLink.

2.3.1 Gestion de la navigation

La carte autopilote et le protocole Mavlink

La carte autopilote supervise les missions. Elle gère la navigation par suivi de waypoints. Pour communiquer avec une station de commande, l'autopilote utilise le protocole MAVLink (Micro Aerial Vehicle Link). Ce protocole permet d'échanger des données à des fréquences assez élevées de manière fiable et robuste entre des architectures aux ressources limitées. L'autopilote peut être paramétré lors de la préparation de la mission par un logiciel de station de contrôle ou via ROS par l'intermédiaire du package Mavros.

Gestion des missions sur une station de contrôle

Les logiciels de station de contrôle qui s'interfacent avec le protocole MAVLink permettent

de planifier aisément les missions pour des engins terrestres, aériens et sous-marins. L'opérateur peut entrer les différents waypoints des missions, ainsi que des paramètres relatifs au pilotage, au guidage et à l'état du robot. Cependant, malgré leur simplicité d'utilisation, ces logiciels peuvent avoir certaines limites concernant la flexibilité des missions. La suite logicielle Ardupilot propose plusieurs programmes open source. Les plus utilisés sont Mission Planner, APM Planner et Mavproxy.

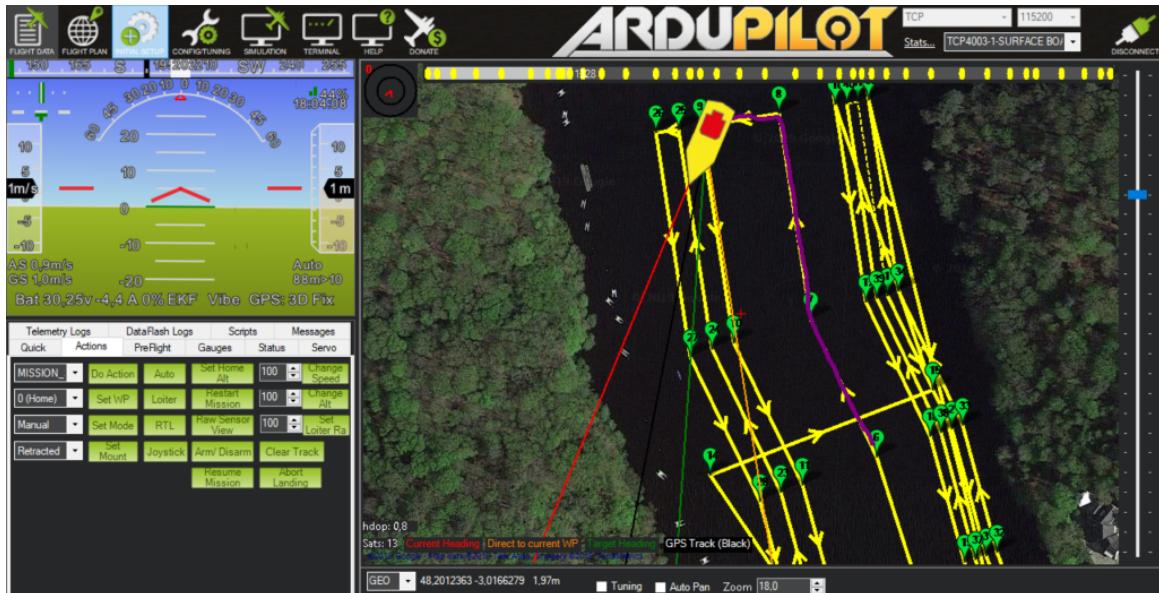


FIGURE 2.14 – Interface du logiciel Mission Planner utilisé lors d'une mission à Guerlédan

Mavproxy

Lors des différentes missions et lors de leur préparation, nous avons également utilisé Mavproxy. Mavproxy est une station de contrôle offrant des fonctionnalités supplémentaires. Elle permet d'utiliser le flux MAVLink sur plusieurs interfaces afin d'avoir le retour d'informations d'Ulysse simultanément sur plusieurs ordinateurs et logiciels. Cela nous donne ainsi la possibilité d'utiliser Mavros et une ou deux station(s) de contrôle simultanément.

Gestion des missions sur ROS

Le package Mavros est un pont de communication entre ROS et le protocole MAVLink. Il est ainsi possible de développer notre propre architecture logicielle de contrôle de l'autopilote. Cela va nous permettre d'automatiser les missions et de contrôler le comportement d'Ulysse plus finement.

Mavros offre via de multiples services et topics, une interface de commande et de réception de données. Cela nous permet notamment d'atteindre :

- l'état du drône : `/mavros/state`, `/mavros/cmd/set_home...`
- la mission : `/mavros/mission`, `/mavros/mission/push...`
- les capteurs de la carte autopilote et les actionneurs : `/mavros/global_position`, `/mavros/imu/data`, `/mavros/battery...`

Ce package permet de se passer complètement des logiciels de station de contrôle. Cependant,

il nous a semblé pertinent de continuer à les utiliser comme outil de visualisation.

2.3.2 Développement et validation par simulation

Les développements des différents algorithmes ainsi que la compréhension de l'architecture ROS ont été facilités par l'utilisation d'un simulateur. Nous avons utilisé celui proposé par Ardupilot. Il offre un modèle dynamique pour différents véhicules. Le choix d'un véhicule de surface nous a permis d'avoir le retour des différents capteurs.

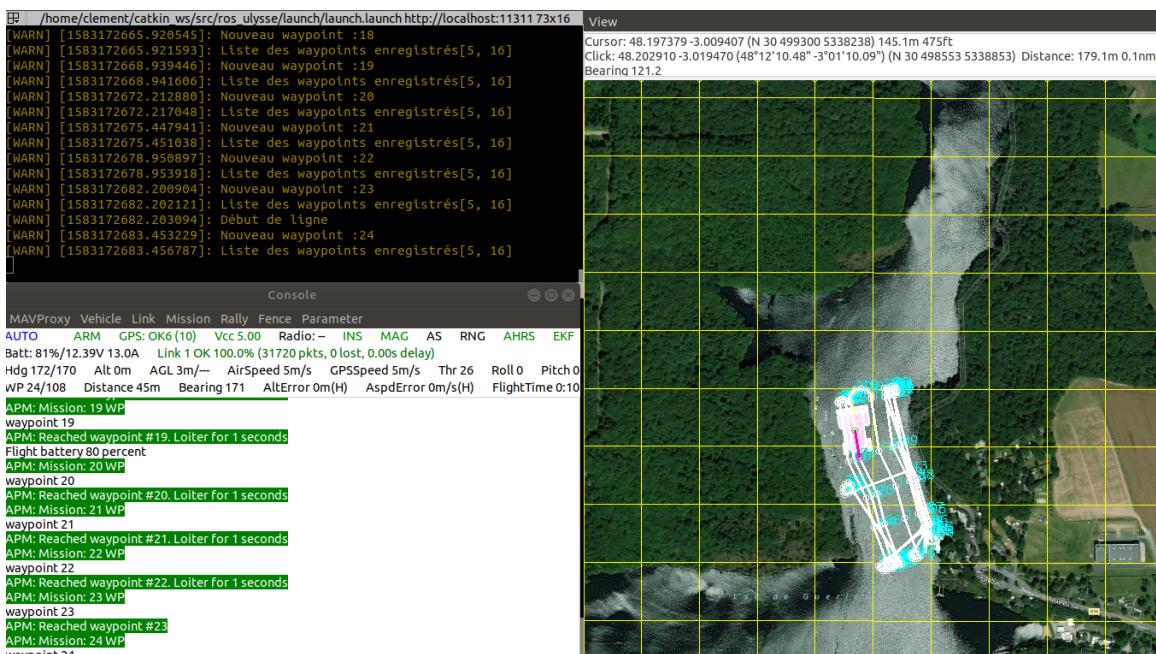


FIGURE 2.15 – Simulation d'une mission à Guerlédan sur le simulateur d'Ardupilot

2.3.3 Commande par waypoint

Afin d'effectuer le levé d'une zone, il faut produire un plan d'inspection de la zone afin de valider certains critères pour la qualité des données acquises (recouvrement des données, éloignement entre les lignes..). Ce plan d'inspection, généré sous le logiciel Qinsy, est composé d'une liste de waypoints définie dans un ordre précis. Ces waypoints décrivent des lignes durant lesquelles il faut acquérir les données du multifaisceau. Ces waypoints décrivent donc des trajectoires généralement à angle droit, demi-tour et angle vifs. Il est nécessaire d'y apporter quelques modifications afin que cette trajectoire puisse être suivie par le robot Ulysse, car il est impératif que la trajectoire des fauchées soit suivie avec la plus grande précision.

Pour ce faire, nous avons développé un algorithme permettant d'adapter la trajectoire :

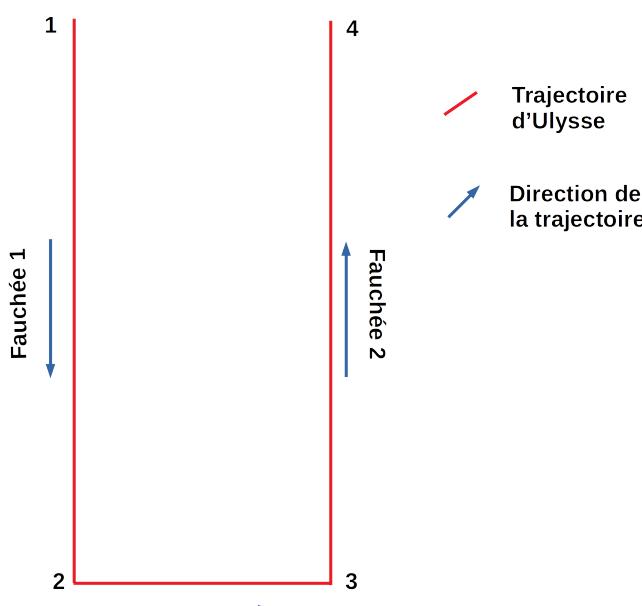


FIGURE 2.16 – Angle droit

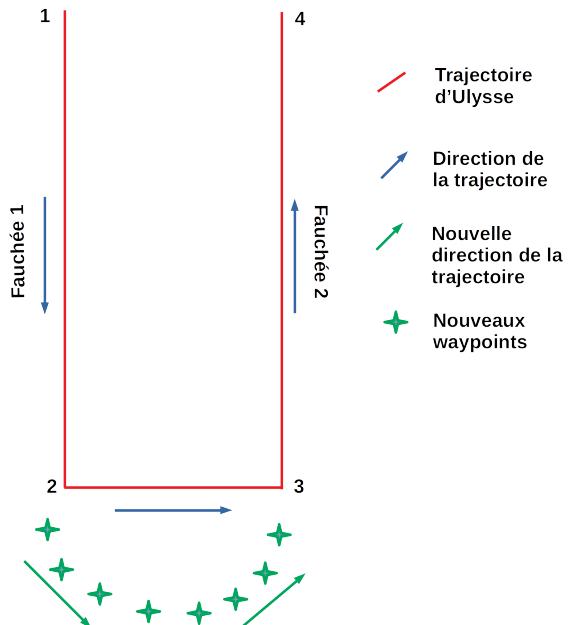


FIGURE 2.17 – Angle droit adapté

La trajectoire ainsi adaptée est alors moins discontinue et permet également de positionner le robot dans l'axe de la fauchée suivante afin d'améliorer d'une part la régulation dès le début de la fauchée, et d'autre part de placer le robot sur la ligne de la fauchée avec précision pour obtenir une bonne qualité de données.

De la même manière on adapte la trajectoire pour des changements de direction supérieure à 90°.

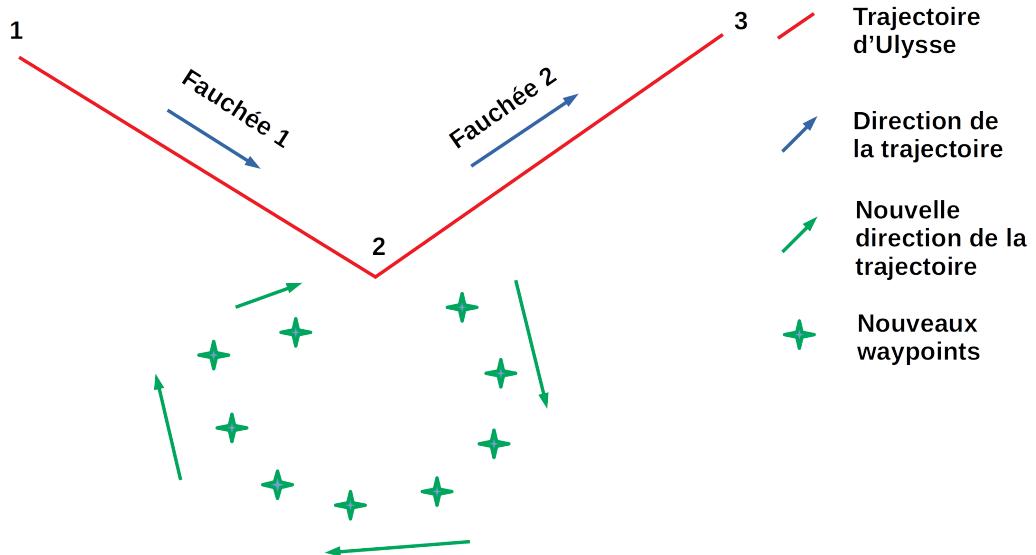


FIGURE 2.18 – Demi-tour/Grand angle

Sur la figure 2.18, on peut observer que l'avant-dernier waypoint est légèrement décalé de l'axe de la fauchée suivante. Ce décalage est prévu pour compenser l'arrêt à un waypoint si l'on

est dans un cercle de 3 mètres autour de celui-ci.

Les waypoints ajoutés doivent être distingués des originaux car ils ne sont utiles qu'à la navigation du robot et pas à l'acquisition des données du sondeur multifaisceau. Les waypoints étant définis par une trame QGC WPL 110, on différencie ces deux types de waypoints avec le paramètre 1 de cette trame, et on enregistre le numéro de la fauchée avec le paramètre 2. Ce dernier paramètre sert lors de la validation des données afin de revenir sur zone lorsque les données ne respectent pas les critères du contrôle qualité.

2.3.4 Le contrôleur

Suivi de waypoints

Nous avons développé un package ROS *ulysse_navigation* intégrant le contrôleur de navigation. Il va connecter la partie navigation du robot avec la partie acquisition des données. Il prend en entrée la liste des waypoints de la mission ainsi que le waypoint courant. Il reçoit également les alertes sur les mauvaises mesures, ainsi que les différents paramètres caractérisant l'état d'Ulysse. Les différentes entrées-sorties sont explicitées dans la figure 2.19.

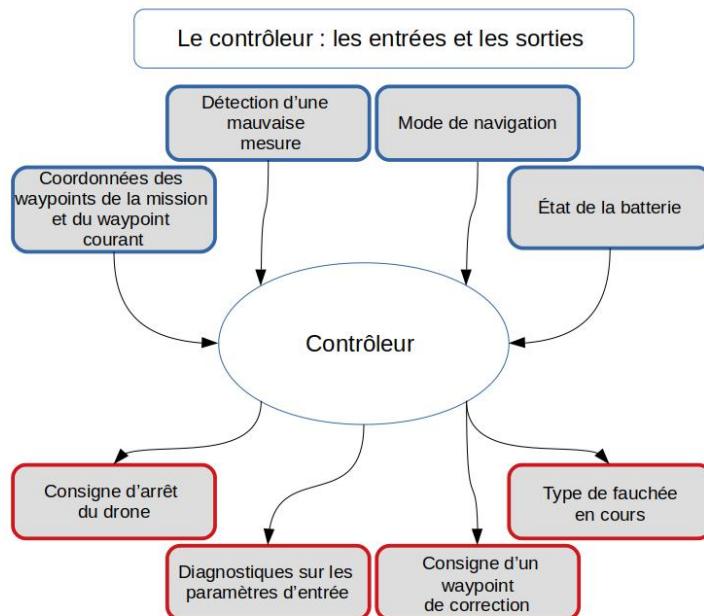


FIGURE 2.19 – Les entrées et les sorties du contrôleur

Lors d'une mission, le contrôleur effectue en temps réel un suivi de l'évolution des waypoints. Lorsqu'une mauvaise mesure est détectée, il est nécessaire de réaliser à nouveau l'acquisition de la fauchée dans son intégralité. Ainsi, si cette mauvaise mesure est remontée jusqu'au contrôleur, la consigne d'un nouveau waypoint de correction est envoyée à la carte autopilote. Afin de laisser du temps au robot pour qu'il se repositionne précisément, le waypoint de correction envoyé est

le dernier de la fauchée précédant l'erreur, comme schématisé sur la figure 2.20.

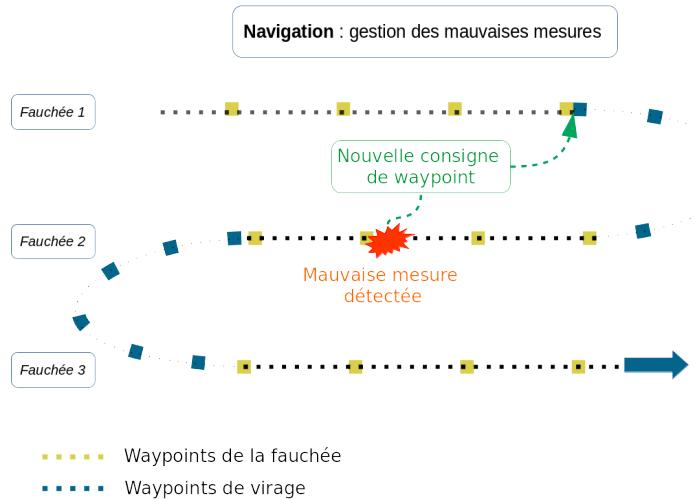


FIGURE 2.20 – Gestion des mauvaises mesures

Ainsi, du point de vue de la navigation, une mission se déroule comme l'illustre la figure 2.21. Suite à une phase d'initialisation où le robot est en guidage manuel ou en mode stationnaire, nous pouvons charger une mission puis lancer Ulysse en mode autonome.

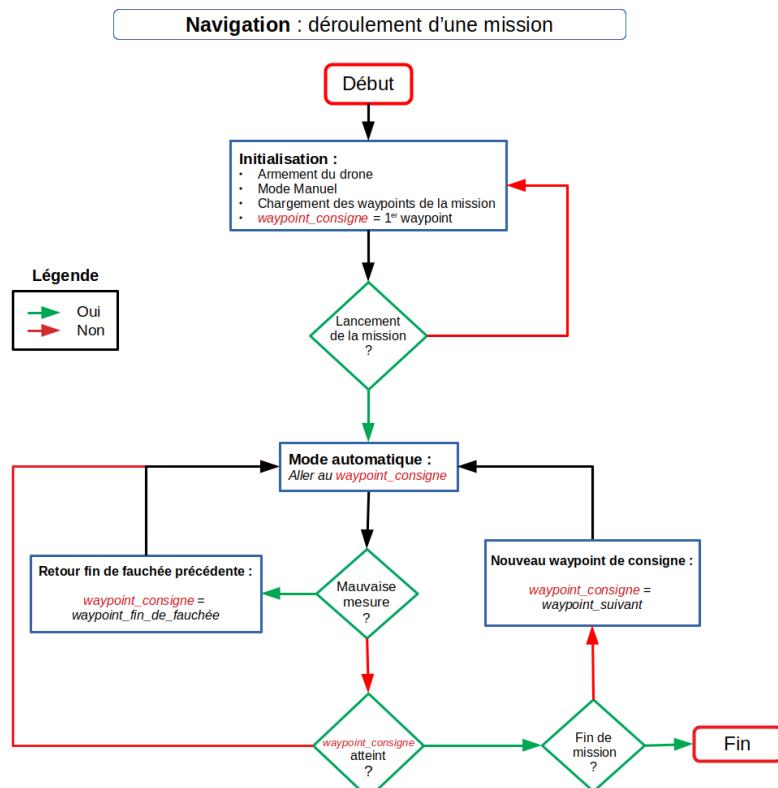


FIGURE 2.21 – Flowchart du déroulement des missions

Chapitre 3

Expérimentation

La partie expérimentation présente le travail effectué lors des deux semaines passées au lac de Guerlédan.

La première semaine avait pour but de prendre en main Ulysse, aussi bien du côté hydrographique que du côté robotique. Nous avons pu effectuer un levé en utilisant la chaîne matérielle et logicielle déjà mise en place au début du projet.

L'objectif de la deuxième semaine consistait également à réaliser un levé, mais en utilisant cette fois-ci la chaîne matérielle et logicielle présentée dans la partie précédente. Ce sont les données de cette seconde expérimentation qui seront par la suite exploitées par les algorithmes de filtrage implémentés sous ROS.

3.1 Préparation de la mission

Il est nécessaire de préparer la mission afin de s'assurer que le levé respecte au mieux les conditions permettant d'obtenir des données valides et qu'il se fasse en toute sécurité. Les quatre étapes nécessaires sont :

1. Préparation des lignes de levé
2. Calibration de l'autopilote
3. Alignement de la centrale inertie
4. Réalisation du patch test

3.1.1 Préparation des lignes de levé

La zone étudiée possède une caractéristique influençant le bon déroulement du levé. En effet, elle comporte deux pentes brutales, étant donné que le milieu du lac correspond au lit d'une ancienne rivière. Il y a donc un gradient important avec de fortes ruptures de lignes. En conséquence, pour couvrir l'intégralité de la zone avec un recouvrement désigné par l'hydrographe, il est nécessaire de calculer la distance entre les lignes en fonction de l'angle d'ouverture 2θ de la pente. Plusieurs hypothèses ont été effectuées pour réaliser les différentes lignes de levé.

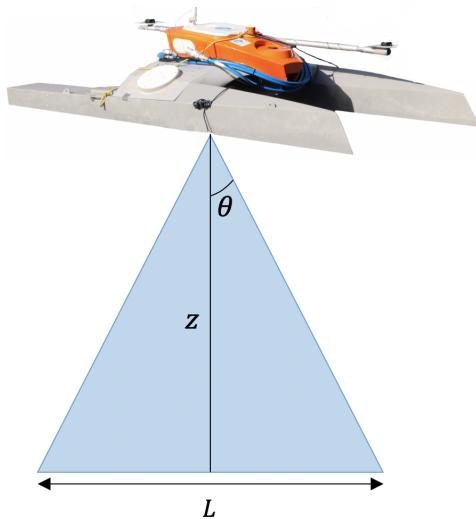


FIGURE 3.1 – Schéma de la largeur de fauchée

D'une part, la largeur de l'empreinte au sol du faisceau L a été calculée en faisant une approximation sur fond plat en considérant la profondeur au nadir z comme illustré par la figure 3.1.

La largeur de fauchée se déduit donc du schéma de la figure 3.1 par :

$$L = 2z * \tan \theta$$

avec θ en radians.

D'autre part, l'évolution de la pente au nadir s'effectue de manière linéaire. Ainsi, au niveau de l'estran, la profondeur au nadir est minimale et vaut 3m, alors qu'au niveau du lit de la rivière, la profondeur est d'environ 40m, comme l'illustre la figure 3.2.

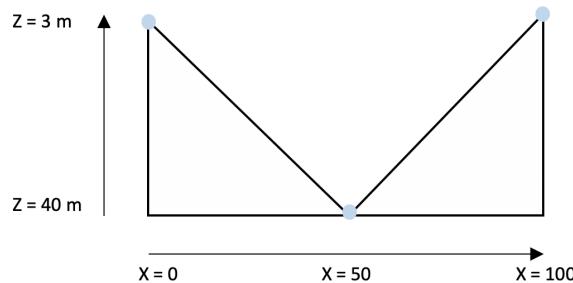


FIGURE 3.2 – Schéma après une approximation de la zone de levé

Ainsi, la relation linéaire entre la position x et la profondeur au nadir z est :

$$z(x) = \frac{37}{50}x + 3$$

car $z(0) = 3$ et $z(50) = 40$.

De cette manière, le tableau 3.1 réunit les numéros de lignes ainsi que leur position sur la zone :

N° ligne	x	z	L
1	0	3	7.17
2	25	21.5	51.39
3	50	40	95.6
4	75	21.5	51.39
5	100	3	7.17

TABLE 3.1 – Largeur de fauchée en fonction de la profondeur au nadir et de la position pour un recouvrement à 100%

Les résultats du tableau 3.1 ont donc permis de calculer la distance horizontale entre chacune des lignes afin de garantir un recouvrement à 100%. Dans le but de pouvoir vérifier la cohérence entre les fauchées, il faut réaliser une traversière, passant perpendiculairement à ces lignes.

Enfin, au delà de la pente, la zone est peu profonde et arrive rapidement au niveau de l'estran. Il est donc nécessaire de réaliser les lignes de levé par rapport à une référence. Cette dernière se trouve être un modèle numérique de terrain effectué en 2018, permettant d'avoir une idée de la profondeur sur la zone.

Première semaine à Guerlédan

Lors de cette semaine, nous avons réalisé un levé sur la zone de Guerlédan suivant les lignes tracées sur la figure 3.3.



FIGURE 3.3 – Lignes de levé sur la zone du lac de Guerlédan

Deuxième semaine à Guerlédan

Lors de cette seconde session, les lignes de levé réalisées pour créer notre surface de référence sont représentées sur la figure 3.4



FIGURE 3.4 – Lignes de levé pour la surface de référence sur la zone du lac de Guerlédan

3.1.2 Calibration de l'autopilote

Avant toute acquisition, il est important de corriger le maximum d'erreurs qui pourraient être dues à la navigation gérée par l'autopilote. Nous devons calibrer les différentes parties de l'autopilote pour que par la suite nos trajectoires soient suivies le plus précisément possible. Les différentes étapes de calibration ont été effectuées à l'aide des logiciels de ArduPilot Development Team and Community, Mission Planner sur Windows et APM Planner sur Linux. Les deux phases de calibration nécessitent de tourner l'autopilote dans différentes directions, il faut donc le retirer d'Ulysse pour ces manipulations. La calibration de l'autopilote se fait en trois phases :

1. Calibration de l'accéléromètre

Dans cette phase, il suffit simplement de tourner la carte autopilote selon différentes positions. Sur le côté gauche, droit, sur le dessus, le dessous, le nez vers le bas et le nez vers le haut. Il est impératif de bien connaître la position de repos du système car toutes les autres sont fonction de cette dernière.

2. Calibration du Compas

Pour cette phase, il faut tourner l'autopilote dans tous les sens en s'assurant d'effectuer au moins 360° de rotation selon chaque axe. Cette partie doit être effectuée à distance d'environnement magnétique et électronique pour diminuer les erreurs lors de la calibration. Elle nous permettra de bien estimer le cap compas, qui est important pour la régulation en cap lors des levés bathymétriques.

3. Réglages des paramètres de régulation

Les levés bathymétriques doivent être effectués à faible vitesse pour obtenir une bonne densité de points acquis, ce qui permettra par la suite en post-traitement d'obtenir une meilleure résolution pour la bathymétrie. Nous avons donc choisi une vitesse de 1.7 m.s^{-1} (6.12 km.h^{-1}). Cette vitesse permet à la fois de valider les critères pour les levés, mais également d'avoir une

vitesse assez conséquente pour que le robot ait l'inertie suffisante pour ne pas être influencé par l'état de l'eau dans le lac de Guerlédan.

Les paramètres de régulation du régulateur PID¹ sont donc proportionnels à cette vitesse choisie. Le calibrage de notre régulateur PID se fait sur le suivi d'une ligne. Il est possible d'estimer les paramètres de la régulation en ne s'intéressant qu'au comportement du robot à travers la trajectoire GNSS retransmise sur la carte de visualisation du logiciel d'Ardupilot. Cette visualisation donne simplement la trajectoire suivie par le robot : pour estimer le paramètre dérivé du PID, il est important de visualiser également le comportement du robot.

Dans notre cas, nous avons choisi un régulateur PD². Le paramètre intégral dans le cas d'un suivi de ligne va nous permettre de suivre la ligne au plus proche. Comme nous avons mis en place des waypoints supplémentaires afin de placer le robot déjà dans l'axe de la fauchée, nous n'avons pas besoin de paramètre intégral. Par ailleurs, le gain choisi en proportionnel permettait déjà d'atteindre les critères souhaités concernant le suivi de ligne.

Le paramètre dérivé est le plus important, car il nous permet de diminuer les dépassemens lors du suivi de ligne, et donc de privilégier le suivi de cap plutôt que celui de la position, le premier étant important lors de l'acquisition des données par le multifaisceau.

Un dernier paramètre à choisir est le "WP_radius". Il s'agit d'un paramètre qui permet de considérer qu'un waypoint est atteint dès lors que la position du robot est dans le cercle qui entoure ce waypoint. WP_radius étant le rayon de ce cercle. Dans le cadre de notre projet, ce paramètre va nous permettre de prendre en compte l'erre du robot. Le robot n'étant pas programmé pour faire marche arrière à l'arrivée sur un waypoint, cette étape est nécessaire pour garder la bonne trajectoire voulue. Ce paramètre dépend également de la vitesse de déplacement que nous avions choisie ($1.7m.s^{-1}$ pour rappel). Nous avons donc fixé ce paramètre à 3 mètres. Celui-ci a été par la suite pris en compte lors de l'adaptation de la trajectoire des waypoints comme illustré dans la section 2.3.3, à la page 22.

3.1.3 Alignement de la centrale inertuelle

Dans le cas de systèmes inertIELS, la phase d'initialisation permettant de calibrer le système se nomme "Alignement de la Centrale", et a pour objectif de déterminer l'attitude du système. Sur la surface terrestre, il existe des informations permettant de connaître l'attitude initiale.

D'une part, la centrale inertuelle est un système comprenant des accéléromètres et des gyromètres. Ces derniers permettent d'estimer respectivement les forces spécifiques du système relatives à la loi fondamentale de la dynamique, ainsi que les vitesses de rotation. Ainsi, si la centrale est suffisamment perfectionnée et les capteurs sont assez sensibles, l'axe de rotation terrestre indique le Nord. D'autre part, le champ de pesanteur permet de déterminer la verticale locale. Enfin, une autre contrainte est que le navire est considéré comme stable sur la surface maritime, permettant d'établir un nouveau plan.

Cette phase d'alignement s'effectue de manière autonome et dure entre un quart d'heure et trente minutes. En effet, nous parlons d'alignement autonome dans le cas où la centrale s'aligne

1. Proportionnel Intégral Dérivé

2. Proportionnel Dérivé

en l'absence de centrale maîtresse, c'est-à-dire une centrale installée sur le même porteur, déjà alignée et initialisée. Sur notre système, les positions sont donc initialisées de manière continue par un système GNSS.

La période d'alignement se décompose en deux étapes : l'alignement grossier et l'alignement fin. L'alignement grossier est réalisé en gardant le robot le plus stable possible, attaché au quai par exemple, afin de maintenir un cap constant. Lors de cette première phase, la centrale cherche son attitude approximativement, avec une précision de l'ordre du degré. Ainsi, une mise à niveau du porteur permet d'estimer les angles de gîte et l'assiette de la centrale par rapport au plan horizontal local. Ensuite, la recherche du cap est effectuée afin de déterminer la direction du Nord géographique. Une fois l'estimation des trois angles d'attitude effectuée, l'alignement fin est réalisé. Les angles d'attitude, vitesse, position ainsi que les erreurs modélisant le modèle précédemment trouvés sont affinés par le biais du filtre de Kalman. Pour se faire, le robot doit avoir une trajectoire la plus variable possible en accélérant et décélérant.

3.1.4 Procédure patch test

La calibration du système multifaisceau est la première étape essentielle avant d'effectuer l'acquisition des données du levé au sondeur multifaisceau. Elle vise à détecter d'une part la désynchronisation entre le capteur de localisation et le sondeur multifaisceau et d'autre part les biais d'alignement en gîte, assiette et cap entre la centrale d'attitude et le sondeur. Nous avons donc mis en œuvre cette calibration par un patch test. Il consiste à traiter les points suivants :

1. Latence : Requiert l'utilisation de deux fauchées orientées dans le même sens, à des vitesses différentes, sur une pente ;
2. Gîte : Requiert l'utilisation de deux fauchées orientées dans des sens opposés sur fond plat ;
3. Assiette : Requiert l'utilisation de deux fauchées orientées dans des sens opposés sur une pente ;
4. Cap : Requiert l'utilisation de deux fauchées orientées dans le même sens sur une pente ;

Dans le cadre du système installé sur Ulysse, il n'y a pas besoin d'estimer la latence. En effet, le sondeur multifaisceau et la centrale inertie sont synchronisés grâce au PPS. Il n'existe donc pas de latence statique entre les deux capteurs. Les angles d'attitude estimés à l'aide du logiciel Qiméra dans le repère (NED) sont :

$$\begin{pmatrix} \varphi \\ \theta \\ \psi \end{pmatrix} = \begin{pmatrix} -0.348 \\ 2.3 \\ 1.9 \end{pmatrix}$$

Afin d'utiliser les différents packages ROS, il est nécessaire de représenter les angles en degrés dans le repère ENU, et également d'effectuer une conversion des angles d'Euler en quaternions. La transformation dans le repère ENU s'effectue par un échange des coordonnées x et y, et par une inversion du signe de z. En effet, le middleware ROS utilise directement les multiplications entre quaternions plutôt que l'utilisation des matrices de rotation. Ainsi, l'ensemble des quaternions permettant de calibrer le système multifaisceau par rapport à la centrale inertie sont :

$$\begin{pmatrix} \omega \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 0.02 \\ -0.003 \\ -0.017 \end{pmatrix}$$

3.1.5 Configuration de ROS

Afin de réaliser une configuration complète, simple et rapide de l'ensemble des algorithmes nécessaires à l'exécution d'une mission, nous avons mis en place un unique fichier de configuration de type YAML nommé *configuration.yaml*. Il permet ainsi le réglage des paramètres tels que : les bras de levier, les angles de calibration, les seuils des filtres, des ports matériels utilisés, le point de référence de la projection *datum* et tous les autres paramètres utilisés.

3.2 Exécution de la mission

Lors de la mission, nous disposons d'un aperçu son exécution. D'une part, l'utilisation de *mavproxy* ou de *Mission Planer* permet de visualiser le parcours effectué par le robot. D'autre part, nous avons implémenté dans tous les packages ROS la publication de messages sur le topic *diagnostics*. Ceci nous permet de moniter via *robot_monitor* le statut de chacun des scripts, dont l'avancée d'exécution des différents filtres. La Figure 3.5 est un aperçu de la visualisation des messages de diagnostique des différents filtres.

All devices	Message
Ulysse	OK
Filters	OK
Density	OK
Files	Trav_3_3_2020-16H25m58s.txt
Result [%]	0.992644261256
Computing time [s]	3.96562504768
State	Not running
Resolution [m]	0.8
MBES frequency [Hz]	400
Percentage threshold [%]	0.95
Density threshold [per mesh]	10
Outlier	OK
Coherence	OK
Swath MANAGER	OK
State	Waiting for a new swath
SSV	OK
Reference [m s]	1442.66
Threshold [m s]	2
Difference [m s]	0
Covering	OK

FIGURE 3.5 – Aperçu de *robot_monitor*

Par exemple, l'algorithme de cohérence a relevé au cours de notre levé un taux supérieur au seuil paramétré. La Figure 3.6 témoigne de cette erreur détectée.

All devices	Message
• Ulysse	Warning
• Filters	OK
• Density	OK
• Files	Trav_3_3_2020-16H25m58s.txt
• Result [%]	0.992644261256
• Computing time [s]	3.96562504768
• State	Not running
• Resolution [m]	0.8
• MBES frequency [Hz]	400
• Percentage threshold [%]	0.95
• Density threshold (per mesh)	10
• Outlier	OK
• Coherence	Warning
• Computing time [s]	5.18134999275
• Acceptable percentage	10
• Coherence threshold [m]	0.1
• State	Not running
• Result [%]	27.0085470085
• Files	['Req_3_3_2020-16H21m3s.txt', 'Trav_3_3_2020-16H25m58s.txt']
• Resolution [m]	1
• Swath MANAGER	OK
• State	Waiting for a new swath
• SSV	OK
• Reference [m s]	1442.66
• Threshold [m s]	2
• Difference [m s]	0
• Covering	OK

FIGURE 3.6 – Aperçu de *robot_monitor* lors de la détection d'une erreur

3.3 Résultats et améliorations envisagées

L'ensemble des messages ROS ayant été échangés lors de l'expérimentation ont été sauvegardés dans un fichier log de type *rosbag*. Comme nous l'avons précisé précédemment, les filtres n'ayant pas pu être implémentés sous ROS lors de cette expérimentation, ce fichier a été essentiel pour le rejeu de la mission en temps réel. Ainsi, une fois les algorithmes mis sous ROS, nous avons pu les tester sur ces lots de données, les évaluer et valider leur comportement ainsi que leur vitesse d'exécution.

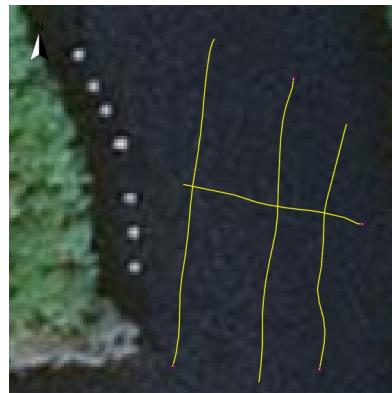


FIGURE 3.7 – Lignes suivies par Ulysse lors de l'expérimentation

3.3.1 Contrôle qualité

Outliers

L'algorithme de détection des sondes aberrantes a été testé selon plusieurs paramètres. Nous avons ainsi pu remarquer l'influence cruciale du choix des paramètres décrits en 2.2.1.

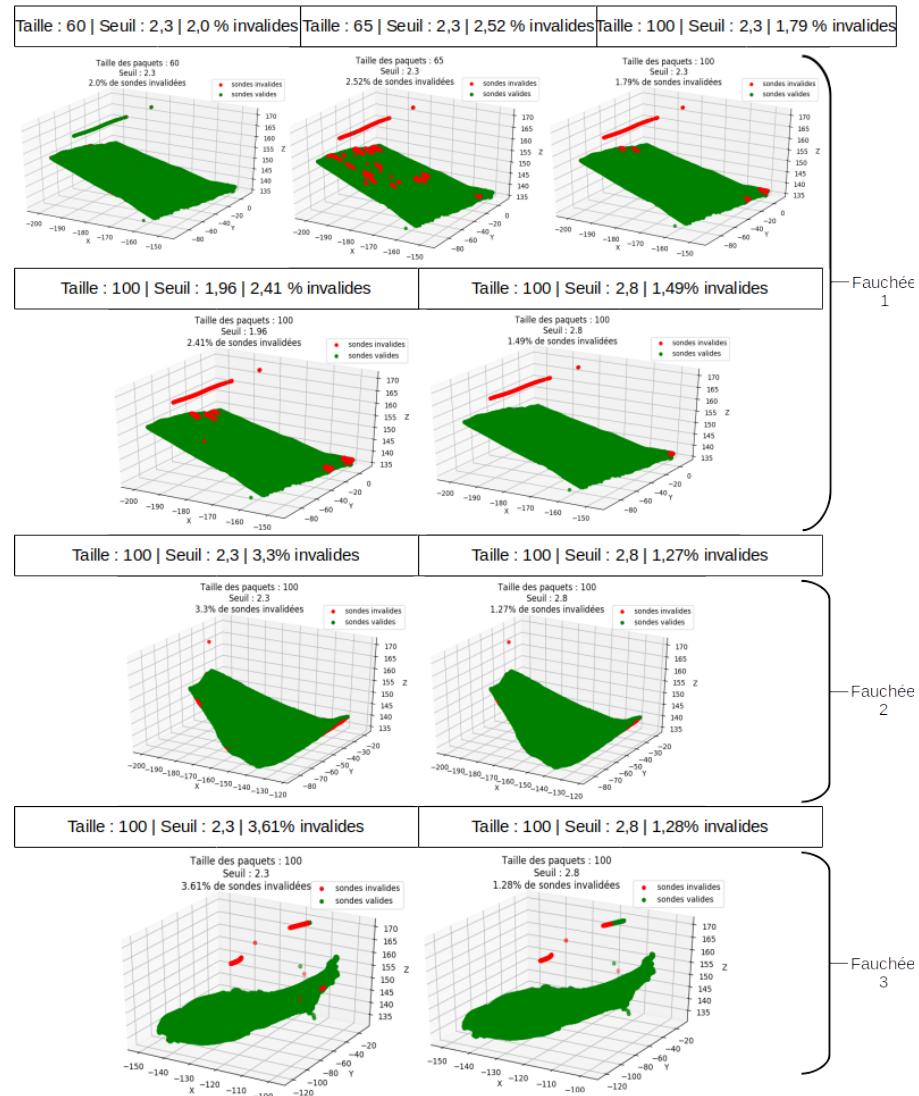


FIGURE 3.8 – Observation et recherche des paramètres optimaux de l’algorithme de traitement des sondes aberrantes

Avec la figure 3.8, nous remarquons que le choix de la taille des paquets de sondes ainsi que le seuil d'éloignement par rapport à la tendance influent de manière significative les résultats de l'algorithme. La figure 3.8 montre ce qui a été obtenu sur les différentes fauchées acquises pendant l'expérimentation. Elle illustre le fait que le choix des paramètres n'est pas trivial, et que l'algorithme fonctionne mieux s'il existe une connaissance a priori du terrain afin d'adapter au mieux ces paramètres.

D'autres algorithmes de détection automatique des sondes aberrantes existent, et peuvent se révéler meilleurs en terme d'adaptabilité à l'étude. En effet, la méthode des moindres carrés considère un plan, ce qui ne reflète pas toujours (voire rarement) la bathymétrie réelle de la zone étudiée. Changer de méthode de filtrage des outliers pourrait contribuer à améliorer l'algorithme, des études pourraient être faites en comparant la technique des moindres carrés avec celle de Hou [2] ou de Du [1], qui prennent en compte plus d'éléments pour invalider une sonde. Cependant, ces autres méthodes sont plus complexes et risquent de ce fait d'impacter le temps de calcul de l'algorithme, qui nous satisfait actuellement car toujours inférieur à 0.7

secondes quelle que soit la fauchée.

Célérité

Nous avons pu constater qu'aucune alerte n'était envoyée suite à l'application du filtre de célérité. Nous savons que la valeur de célérité en surface change au cours du levé. R2Sonic va lui utiliser une moyenne de plusieurs valeurs de célérité en surface afin de calculer les angles et les temps dont il a besoin pour établir les données de profondeur et de position. Les erreurs aberrantes sont donc initialement lissées par cette moyenne, et ce n'est donc pas une valeur de célérité aberrante qui impacte une erreur ponctuelle sur le fond. Nous pouvons alors déduire que les changements de célérité relevés par alarmes sont la conséquence d'un changement de masse d'eau, et non pas d'une erreur capteur. Il n'y a donc dans notre cas aucun changement de masse d'eau au niveau de la zone où le levé a été effectué.

Recouvrement

La figure 3.9 illustre les trois lignes régulières qui ont été acquises pendant l'expérimentation. Ce sont celles-ci qui sont donc traitées par l'algorithme vérifiant le recouvrement entre les fauchées.

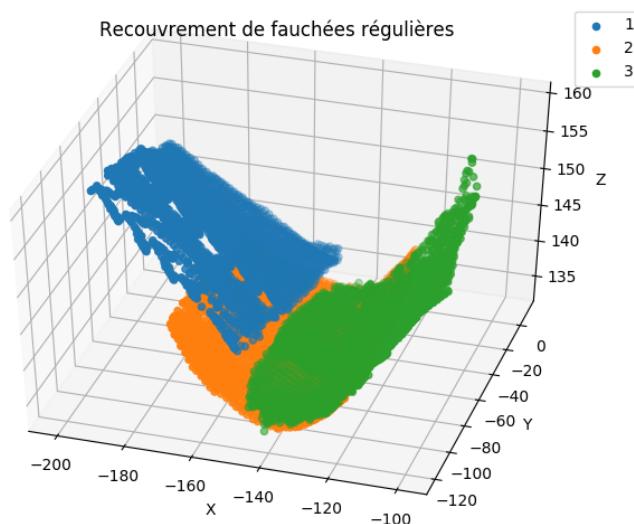


FIGURE 3.9 – Vue 3D des lignes régulières acquises lors de l'expérimentation

Le tableau 3.2 présente le recouvrement obtenu lors de l'expérimentation :

Numéro fauchée	1 sur 2	2 sur 1	2 sur 3	3 sur 2	RECOUVREMENT TOTAL 2
Taux de recouvrement	39.23%	41.51%	61.86%	57.63%	96.86%

TABLE 3.2 – Recouvrement obtenu pour l'expérimentation

Nous attendions un recouvrement de 100% entre les fauchées, nous obtenons un total de 96.86% et aucune alerte n'a été levée par l'algorithme. En fait, nous avions établi un seuil de tolérance de 5% et l'écart observé est de 3.14% : nos critères sont donc bien respectés. De plus, le temps de calcul de l'algorithme est très satisfaisant puisqu'il ne dépasse pas la seconde.

Comme évoqué précédemment, le recouvrement entre les fauchées est tributaire de la bathymétrie du terrain étudié. Ainsi, si l'hydrographe a une connaissance a priori de la zone du levé, il lui est possible d'adapter le profil de la mission pour assurer le recouvrement souhaité. Dans le cas contraire, l'opérateur n'est pas à l'abri d'une surprise bathymétrique comme une élévation, réduisant alors l'emprunte de la fauchée et empêchant le recouvrement exigé. Une alerte serait alors lancée par l'algorithme, et la mission serait à refaire puisque l'hydrographe devra repenser le design de la mission en fonction de l'événement bathymétrique. Pour éviter ces manipulations supplémentaires, il serait intéressant d'améliorer les fonctionnalités de l'USV en lui permettant de planifier lui-même la mission en temps réel en fonction de la bathymétrie et des critères exigés par l'hydrographe.

Densité

L'algorithme implémenté pour filtrer la densité calcule en moins de 4 secondes une densité de plus de 99% pour chacune des fauchées.

Pour calculer l'empreinte du faisceau sur le fond, l'algorithme part du principe que celui-ci est au nadir. Il y a une amélioration à apporter du point de vue de l'algorithme, en récupérant l'information sur l'ouverture du faisceau en temps réel pour la prendre en compte dans ce calcul.

Dans le résultat de la fonction *Densité par maille* de l'algorithme en Figure 2.12, nous observons que la densité de sondes par maille est plus forte pour les faibles profondeurs (représentée par les isolignes). Cela s'explique par le fait que la distance entre les faisceaux est plus faible lorsque la profondeur est moindre, du fait de l'angle d'incidence des faisceaux. Enfin, dans le résultat de la fonction *Respect du seuil* de l'algorithme, nous observons que les mailles ne répondant pas au critère de densité imposé à 10 pour cette expérimentation sont situées principalement au bord de la fauchée. Cela est dû au fait que l'on se situe au niveau des faisceaux latéraux. Ce point de l'algorithme peut également être amélioré en calculant une densité selon le paramètre de recouvrement entré par l'opérateur.

Cohérence

Le temps d'exécution du filtre est inférieur à 5 secondes dans le cas de notre étude. Le résultat des différences des profondeurs au niveau de l'intersection des deux fauchées, l'une régulière et l'autre traversière est représenté sur la figure 3.10.

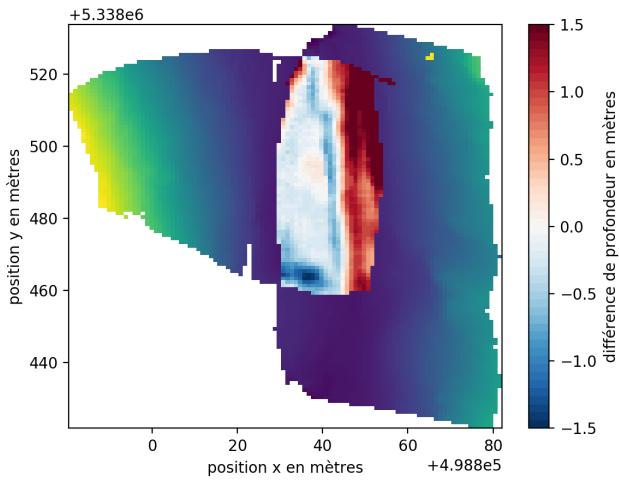


FIGURE 3.10 – Différence de profondeur entre les deux lignes

Nous constatons que les fauchées ne sont pas entièrement cohérentes suite au seuil de 10cm d'écart de profondeur établi précédemment. En effet, la dynamique de l'écart demeure importante lorsque deux points de sondes normalement cohérents sont distants de 1m.

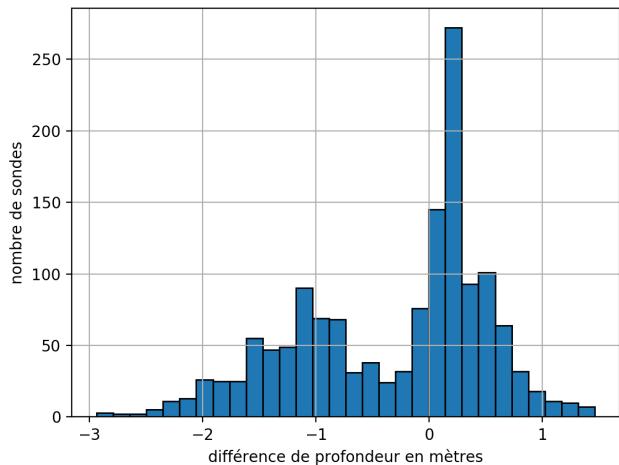


FIGURE 3.11 – Histogramme des différences des profondeurs

L'histogramme de la figure 3.11 met en évidence la dispersion des valeurs de différences, donc la majorité est comprise entre -1 et 1. Nous pouvons supposer ici que cette dispersion est due à la topographie et donc aux reliefs importants. En effet, le seuil imposé est restrictif alors que la pente a un impact direct sur la détection du fond par le sondeur multifaisceau. Les faisceaux latéraux n'arrivent donc pas en incidence normale au niveau du fond, détériorant alors la qualité des données.

Ces données sont directement issues des algorithmes d'automatisation et de la structure effectuée sous ROS, c'est pourquoi, en parallèle, une acquisition a été effectuée sur Qinsy afin de pouvoir faire une comparaison. Il est donc possible de visualiser une coupe de profil comme sur la figure 3.12 des lignes régulières et traversière.

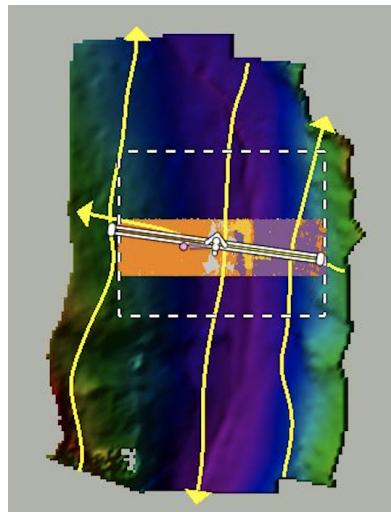


FIGURE 3.12 – Coupe réalisée sur la surface

Ainsi, les allures des profondeurs obtenues sont visualisables en deux dimensions sur les figures 3.13 et 3.14.

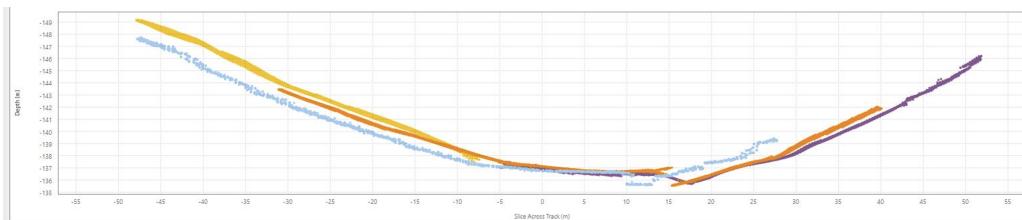


FIGURE 3.13 – Profil vertical des fauchées issues de ROS

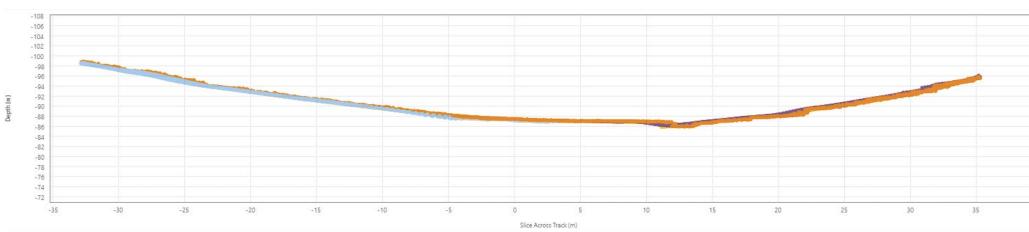


FIGURE 3.14 – Profil vertical des fauchées issues de l'acquisition sous Qinsy

Ainsi, nous constatons que les données issues de l'acquisition sous Qinsy sont cohérentes entre elles, contrairement aux données provenant de l'automatisation sous ROS, ce qui confirme le résultat obtenu suite à l'application du filtre de cohérence avec une forte dispersion des données. Nous pouvons donc supposer que cette incohérence est la conséquence d'une erreur systématique présente lors du paramétrage sous ROS. Ainsi, seul l'algorithme de cohérence a permis de mettre en évidence la présence d'erreur systématique. Deux hypothèses concernant la cause de cette incohérence sont mises en place. D'une part, il est possible que cela vienne d'une mauvaise application des bras de levier calculés entre les antennes des GNSS et la centrale inertielles. D'autre part, cela peut être dû à la conversion des angles d'attitude issus de la calibration au

format utilisé par ROS. Nous observons que les fauchées sont également incohérentes au niveau des faisceaux latéraux. Il est donc envisageable de créer un nouvel algorithme permettant de prendre en compte le poids des faisceaux.

Ainsi l'ensemble des résultats obtenus par les algorithmes de filtre sur le jeu de données sauvegardé et rejoué prouve qu'ils sont fonctionnels et permettent le contrôle qualité du MNT en temps réel.

3.3.2 Modèle numérique de terrain

Le MNT résultant de l'expérimentation menée lors de la deuxième semaine de Guerlédan a permis de tester le bon comportement des algorithmes des filtres développés. Cependant, ne les ayant implémenté sous ROS qu'après l'expérimentation, nous n'avons pas pu effectuer de retour sur zone lors de la détection de mauvaises acquisitions. Ainsi, le MNT présenté dans la Figure 3.15 est brut. De plus, il représente l'ensemble des sondes acquises en temps réel : les virages entre deux fauchées ainsi que les sondes à zéro sont donc également affichés. Enfin, on observe bien le problème de cohérence entre la fauchée traversière et les fauchées régulières, comme l'a soullevé le filtre.

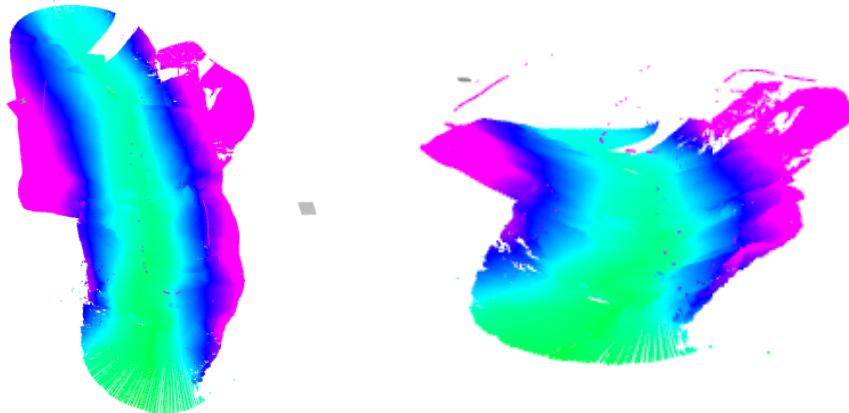


FIGURE 3.15 – MNT réalisé sous ROS visualisé en temps réel via *RViz*

Bien que ce MNT soit brut et qu'il soulève l'existence d'un problème systématique, sa réalisation permet de prouver qu'il est possible d'effectuer un MNT en temps réel en utilisant uniquement une architecture ROS. Par faute de temps, nous n'avons pas corrigé ces erreurs sur le jeu de données enregistré. Mais la souplesse de ROS et du *rosbag* utilisé nous permet de modifier le fichier log et de rejouer la mission corrigée.

Conclusion

Les différentes expérimentations et développements nous ont permis d'améliorer l'automatisation des levés bathymétriques, avec un contrôle qualité des données en temps réel sur de petits fonds.

Suite à la prise en main des différents systèmes composant le robot, il nous a fallu mutualiser les connaissances des spécialités de l'hydrographe et du roboticien pour développer une architecture logicielle sous le middleware ROS, composée de différents programmes.

Ainsi, nous avons utilisé l'ensemble des capteurs à disposition afin d'acquérir des données bathymétriques dans un référentiel projeté. Cela nous a permis de générer un modèle numérique de terrain en temps réel. Pour s'assurer de la qualité des mesures, des filtres ont été développés afin de détecter des erreurs aberrantes, de vérifier le recouvrement et la cohérence entre les fauchées ainsi que la densité des mesures. Pour gérer la navigation, il a fallu s'assurer du bon déroulement des missions et de la réalisation d'un suivi de trajectoires optimal. La possibilité de retour sur zone lors de l'invalidation des données d'une fauchée a été implémentée dans les algorithmes du contrôleur. ROS s'est ainsi distingué comme étant la solution adéquate afin de relier les deux domaines de spécialités exigés pour ce projet.

En somme, le développement de notre système et les expérimentations ont permis de valider le concept initialement défini. Bien que fonctionnel, de nombreuses améliorations peuvent être apportées à ce projet ambitieux comme nous l'avons souligné dans ce rapport. De ce fait, les travaux effectués sont prometteurs quant à une nouvelle façon de réaliser des levés bathymétriques via une solution libre de droits et autonome.

Glossaire

DEM	Digital Elevation Model
EDF	Électricité De France
ENSTA Bretagne	École Nationale Supérieure de Techniques Avancées Bretagne
ENU	East North Up
GNSS	Géolocalisation et Navigation par un Système de Satellites
IMU	Inertial Measurement Unit
MBES	MultiBeam Echo Sounder
MNT	Modèle Numérique de Terrain
NED	North East Down
NMEA	National Marine Electronics Association
NUC	Next Unit of Computing
PD	Proportionnel Dérivé
PID	Proportionnel Intégral Dérivé
ROS	Robot Operating System
RTK	Real Time Kinematic
RTKLIB	Real Time Kinematic Library
SSV	Sound Surface Velocity
SVP	Sound Velocity Profile
UDP	User Datagram Protocol
USV	Unmanned Surface Vehicle
UTM	Universal Transverse Mercator

Bibliographie

- [1] Z Du, D Wells, and L Mayer. An approach to automatic detection of outliers in multibeam echo sounding data. *Oceanographic Literature Review*, 7(43) :737, 1996.
- [2] Tianhang Hou, Lloyd C Huff, and Larry A Mayer. *Automatic detection of outliers in multibeam echo sounding data*. Hydrographic Society of America, 2001.
- [3] Sydney Levitus and Grigory Isayev. Polynomial approximation to the international equation of state for seawater. *Journal of Atmospheric and Oceanic Technology*, 9(5) :705–708, 1992.
- [4] R2Sonic LLC. SONIC 2020 Operation Manual V3.0.
- [5] Patrick Sillard. *Estimation par moindres carrés*. Hermès Science Publications, 2001.
- [6] SBG SYSTEMS. Ellipse ekinox apogee series - firmware manual.