CORRECTION DE DÉRIVE PAR L'INTÉGRATION D'UN ALGORITHME VSLAM DANS UN SYSTÈME EKF ET MPC

par

Ewan FÉMINIS
Clement IMFURA
Nimai JARIWALA

RAPPORT TECHNIQUE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE DANS LE CADRE DU PROJET DE FIN D'ÉTUDES

MONTRÉAL, LE 12 DÉCEMBRE 2025

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

REMERCIEMENTS

Ce projet a été monté après une longue implication dans les clubs scientifiques de l'École de technologie supérieure. C'est un projet d'une grandeur significative et qui nécessite beaucoup d'engagements pour fournir des résultats concrets et pertinents. Cependant, il n'aurait pas été possible sans l'accompagnement et le soutien des personnes qui nous sont chères. C'est dans ce cadre que nous souhaitons adresser nos sincères remerciements aux personnes suivantes.

Tout d'abord, nous souhaitons remercier notre professeur M. Jean-Philippe Roberge, pour son accompagnement et sa supervision tout le long de la session. Ses idées, son écoute et ses rétroactions nous ont permis d'enrichir nos connaissances et de mener le projet à terme avec succès.

Nous souhaitons également remercier nos amis et membres du club S.O.N.I.A. pour leur soutien moral et présence remarquable pendant les tests en piscine. Leur participation en tant que nageurs lors des essais, la préparation et la gestion des batteries, la réservation des piscines ainsi que l'organisation du matériel de test ont grandement contribué au bon déroulement du projet. De plus, nous remercierons Justin Thériault d'avoir généreusement offert son temps pour assurer le rôle de sauveteur lors des tests en piscine, garantissant ainsi le bon déroulement et la sécurité des phases de tests.

Finalement, nous tenons à remercier le département de génie de systèmes pour l'approbation de ce projet ainsi que pour le soutien financier pour des réservations des piscines nécessaires à la réalisation des tests cruciaux du projet.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AUV	Autonomous Underwater Vehicule
DVL	Doppler Velocity Log
EKF	Extended Kalman Filter
IMU	Inertial Measurement Unit
MPC	Model Predictive Control
PWM	Pulse-Width Modulation
ROS	Robot Operating System
VSLAM	Visual Simultaneous Localization and Mapping
VO	Visual Odometry

INTRODUCTION

Le présent travail porte sur l'amélioration du système de contrôle de mouvement d'un véhicule sous-marin autonome (AUV). Le système actuel repose sur un filtre de Kalman étendu (EKF) combinant les données d'une centrale inertielle (IMU) et d'un Doppler Velocity Log (DVL), comme sur le prototype principal AUV8.1. Dans le cadre du développement du nouveau prototype LITE1, un second IMU a été intégré au filtre de Kalman en remplacement du DVL, une configuration qui nécessite encore des validations expérimentales. Les données issues des capteurs alimentent ensuite un contrôleur prédictif non linéaire adaptatif (MPC) chargé de générer les signaux PWM pour les huit moteurs du véhicule. Malgré la stabilité générale obtenue, le système présente une accumulation progressive d'erreurs au fil du temps, même avec l'ajout de nouveaux capteurs. Afin de pallier ce phénomène, l'objectif de ce projet est de concevoir une méthode rigoureuse et efficace de correction de la dérive, en s'appuyant sur l'intégration d'un système de cartographie basé sur l'algorithme VSLAM (Visual Simultaneous Localization and Mapping). L'analyse des données issues de VSLAM doit permettre la correction en temps réel via le module de commande, afin d'améliorer la précision et la robustesse globale de la plateforme en conditions réelles.

CHAPITRE 1
DÉFINITION DE LA PROBLÉMATIQUE

1.1 Définition de la dérive

La dérive est définie comme la déviation progressive et incontrôlée (Le Robert, [s d]). Dans le cadre de ce projet, la définition utilisée sera : la déviation progressive et incontrôlée du sous-marin, en rotation ou en translation, qui n'est pas détectée ou corrigée. Cette déviation ne peut être mesurée avec les capteurs du sous-marin puisque ceux-ci ne détectent pas les mouvements, autrement la déviation serait corrigée par le contrôle du sous-marin. La dérive doit donc être évaluée par un système externe au sous-marin afin d'obtenir des mesures représentatives. Selon les observations préalables, la dérive actuelle est surtout en rotation sur l'axe z et semble venir des imprécisions de l'IMU. Cependant, dans certaines conditions particulières, d'autres mouvements incontrôlés peuvent apparaitre. Par exemple, lorsque le DVL est incapable de calculer la vitesse de déplacement, le sous-marin ne peut plus calculer sa position, celui-ci commence donc à se déplacer sur le plan horizontal de façon quasi aléatoire. Cela arrive lorsque l'onde envoyée par le DVL n'est pas renvoyée ou est déformé, comme lorsque le sous-marin est retourné ou lorsque le fond de la piscine est trop souple.

1.2 Cas d'utilisation

Ce projet vise à améliorer la stabilité à long terme des sous-marins autonomes en réduisant la dérive en rotation et en position. Pour cela, l'algorithme VSLAM utilise les images et les données de l'IMU fournies par la caméra afin de calculer la vitesse et la position actuelle. Ces informations sont ensuite utilisées pour corriger la dérive générée par les imprécisions des capteurs. Cependant, l'algorithme VSLAM a besoin de repères visuels pour fonctionner. Cela empêche donc l'utilisation en pleine mer, puisqu'aucun élément visible ne serait disponible à proximité. Toutefois, la dérive est moins problématique pour cette utilisation puisqu'il n'y

aurait rien à proximité avec quoi le sous-marin risquerait d'entrer en collision. Ce projet est particulièrement pertinent pour les environnements étroits avec beaucoup d'éléments visuels à éviter, comme des cavernes sous-marines ou des épaves. Les applications de ce projet sont très spécifiques aux sous-marins autonomes, mais pourraient également servir pour des drones aériens dans des environnements qui ne permettent pas l'utilisation de GPS, comme des mines (DroneXperts, 2025) ou des opérations qui demandent de la discrétion. Il est important de noter que ces technologies existent déjà, mais ne sont pas très répandues puisque le GPS est une meilleure solution lorsque disponible et que la plupart utilisent un LIDAR (Aitken, 2025), bien que d'autres technologies existent aussi. Ce projet utilise uniquement une caméra stéréoscopique avec une centrale inertielle intégrée pour le VSLAM, ce qui pourrait possiblement être une autre solution pour les drones aériens.

1.3 Impact environnemental, économique et social

1.3.1 Environnemental

D'un point de vue environnemental, la réduction de la dérive de navigation se traduit directement par une meilleure efficacité énergétique. En conservant davantage d'énergie au fil du temps, le prototype diminue sa consommation globale de batterie, prolongeant ainsi sa durée de vie en réduisant la fréquence des cycles de recharge. Cela permet non seulement de limiter les déchets électroniques, mais aussi de réduire l'empreinte environnementale liée à la production et au recyclage des batteries, ainsi qu'à la production de l'énergie nécessaire à la recharge. Comme l'indiquent Soori et al. (2023), « optimization of energy consumption in […] robots can reduce operating costs, improve performance and increase the lifespan of the robot. » (traduction libre : l'optimisation de la consommation d'énergie dans les robots […] peut réduire les coûts d'exploitation, améliorer les performances et augmenter la durée de vie du robot.)

À l'inverse, la dépendance aux capteurs de navigation à l'estime tels que l'IMU, le DVL et le capteur de profondeur entraîne une dérive cumulative, car ces instruments ne possèdent pas de mécanismes intrinsèques de correction d'erreur de dérive. Lorsqu'ils sont intégrés dans le système de contrôle, cette dérive est amplifiée, générant des inefficacités importantes lors de missions de longue durée ou sur de longues distances. Le prototype peut perdre sa cible et être forcé de réexaminer l'environnement, consommant ainsi plus d'énergie qu'il n'en faudrait et gaspillant un temps précieux. L'utilisation d'un algorithme SLAM corrige activement ce problème en rectifiant en continu la dérive, réduisant ainsi à la fois les pertes de temps et les pertes d'énergie. Cela profite directement à l'environnement en diminuant la consommation énergétique du système et en prolongeant la durée de vie utile de ses batteries.

Il est vrai que l'ajout d'un module SLAM augmente la demande computationnelle. Cependant, les deux prototypes de S.O.N.I.A. alimentent leurs ordinateurs embarqués avec une tension constante, ce qui fixe la limite supérieure de consommation électrique. Même sur des plateformes où la consommation énergétique de l'ordinateur varie, des algorithmes efficaces comme ORB-SLAM et SVO sont connus pour équilibrer performance et faible consommation énergétique. Comme le soulignent Chen et al. (2024), « algorithms like ORB-SLAM and SVO offer a more balanced approach, achieving moderate performance with significantly lower energy consumption. [Especially for drones] where onboard computational power is limited, and power efficiency is critical. These algorithms, with lower energy demands, are well-suited for platforms where sustainable energy usage is prioritized, and continuous operation is needed with minimal energy wastage. » (traduction libre : les algorithmes tels qu'ORB-SLAM et SVO offrent une approche plus équilibrée, atteignant des performances modérées avec une consommation énergétique significativement réduite. [Particulièrement pour les drones] où la puissance de calcul embarquée est limitée et où l'efficacité énergétique est cruciale. Ces algorithmes, avec leurs faibles besoins énergétiques, conviennent parfaitement aux plateformes où l'utilisation durable de l'énergie est prioritaire et où un fonctionnement continu est nécessaire avec un minimum de gaspillage énergétique.) Dans notre cas, l'énergie

économisée en réduisant les temps de recherche inefficaces compense largement les coûts computationnels liés aux corrections de dérive, produisant un bénéfice environnemental net.

1.3.2 Économique

L'intégration d'un algorithme de VSLAM engendre d'abord des coûts initiaux liés au développement logiciel, à l'intégration système et, le cas échéant, à l'ajout de capteurs ou de capacités de calcul supplémentaires. Toutefois, ces coûts fixes peuvent être amortis sur l'ensemble des missions réalisées par le prototype. Sur le plan opérationnel, l'utilisation du VSLAM permet de réduire la dérive associée à la navigation à l'estime, ce qui diminue la durée moyenne des missions, ainsi que les risques d'échec. Cette amélioration de la fiabilité et de la rapidité se traduit par une réduction des coûts liés au temps de mission, à la consommation énergétique et aux interventions humaines nécessaires à la récupération ou à la reconfiguration du système.

De plus, la capacité de fermeture de boucle et de relocalisation offerte par le VSLAM, telle que mise en évidence par Rahman et al. (2019) : « the inclusion of a VSLAM algorithm enables the use of loop-closure and re-localization, both having a significant impact on the precision of current and expected navigation. » (traduction libre : l'inclusion d'un algorithme VSLAM permet l'utilisation de la fermeture de boucle et de la relocalisation, ayant toutes deux un impact significatif sur la précision de la navigation actuelle et prévue.) Ceci contribue à accroître le taux de succès des missions et à limiter le risque de perte du véhicule, lequel représente un coût financier majeur. À moyen et long terme, la diminution des risques opérationnels, de l'usure du matériel et des temps d'arrêt permet d'améliorer la rentabilité globale du système. Ainsi, bien que l'implémentation du VSLAM implique un investissement initial non négligeable, celui-ci est compensé par des économies cumulées et une meilleure prévisibilité des coûts lors des déploiements répétés.

1.3.3 Social

La dimension sociale de ce projet se manifeste principalement dans les domaines de la sécurité, de la confiance et de l'accessibilité. Les véhicules sous-marins autonomes (AUV) qui dérivent ou échouent de façon imprévisible représentent des risques pour les plongeurs humains, les infrastructures environnantes et les véhicules eux-mêmes. Chen et al. (2025) notent, dans une étude de synthèse sur les défaillances d'AUV, que « many of the typical failures are due to some kind of instrument failure such as Gyroscope or accelerometer failure, DVL antenna and sensor failure, and Depth gauge's pressure sensor damage. » (traduction libre : nombre des défaillances typiques sont dues à un type de panne instrumentale, comme une panne de gyroscope ou d'accéléromètre, une défaillance de l'antenne ou du capteur DVL, et des dommages au capteur de pression de la jauge de profondeur.) Ces défaillances représentent non seulement des défis techniques, mais aussi des risques en matière de sécurité.

L'utilisation de l'odométrie visuelle inertielle (VIO), qui combine SLAM et données d'IMU, a démontré des résultats probants pour maintenir la navigation lorsque les capteurs conventionnels ne fonctionnent pas comme prévu. Joshi et al. (2023) observent ainsi que VIO « has shown results in aiding the AUV navigate its environment when standard sensors do not perform as expected. » (traduction libre : a montré des résultats dans l'aide à la navigation de l'AUV dans son environnement lorsque les capteurs standards ne fonctionnent pas comme prévu.) L'intégration de cette approche réduit les risques liés aux pannes instrumentales, limitant les mouvements imprévisibles susceptibles de mettre en danger des plongeurs, d'endommager d'autres véhicules ou de nuire à l'environnement marin.

En améliorant la stabilité et la fiabilité de la navigation, le SLAM et la VIO renforcent la sécurité humaine et les taux de réussite des missions. Au-delà de ces bénéfices immédiats, la diminution des coûts et l'amélioration de l'efficacité rendent les technologies de navigation avancées plus accessibles à un plus grand nombre d'organisations, des groupes de recherche universitaires aux institutions de plus petite taille. Cette accessibilité accrue favorise

l’innovation, la collaboration et la formation, des bénéfices sociaux majeurs qui dépassent le simple cadre technique du projet.

CHAPITRE 2
ÉQUIPEMENT ET FACTEURS EXTERNES

2.1 Prototypes

Le club S.O.N.I.A. possède actuellement deux prototypes actifs et une récemment retraité. Bien que les prototypes soient globalement similaires, chacun possède des éléments et des concepts uniques qui les distinguent. Ces particularités permettent de les combiner pendant les compétitions afin d'accomplir les missions de manière optimale et rapide.

2.1.1 AUV8.1

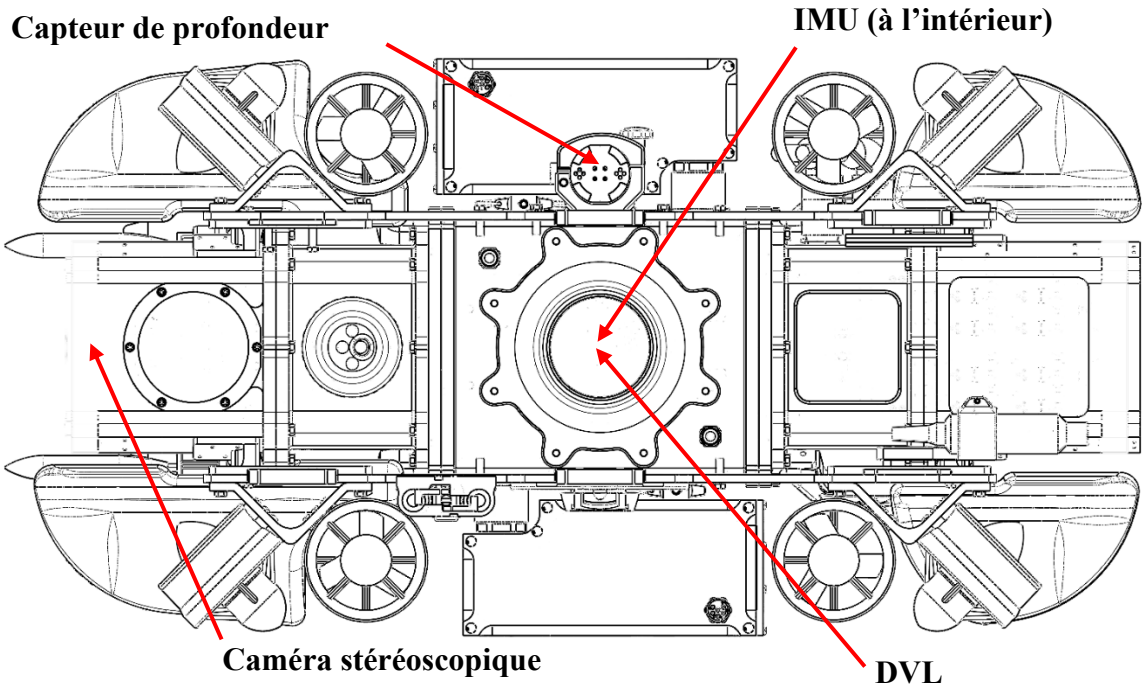

Figure 1 Vue de dessous de l'AUV8.1 avec les capteurs

L'AUV8.1 est le plus vieux des 2 prototypes actifs. Sa conception a commencé en 2019, mais à cause de la pandémie, sa première compétition ne fut qu'en 2021. Il est équipé de huit moteurs, un DVL, un IMU, un capteur de profondeur et une caméra stéréoscopique. C'est le sous-marin principal lors de la compétition.

2.1.2 LITE1

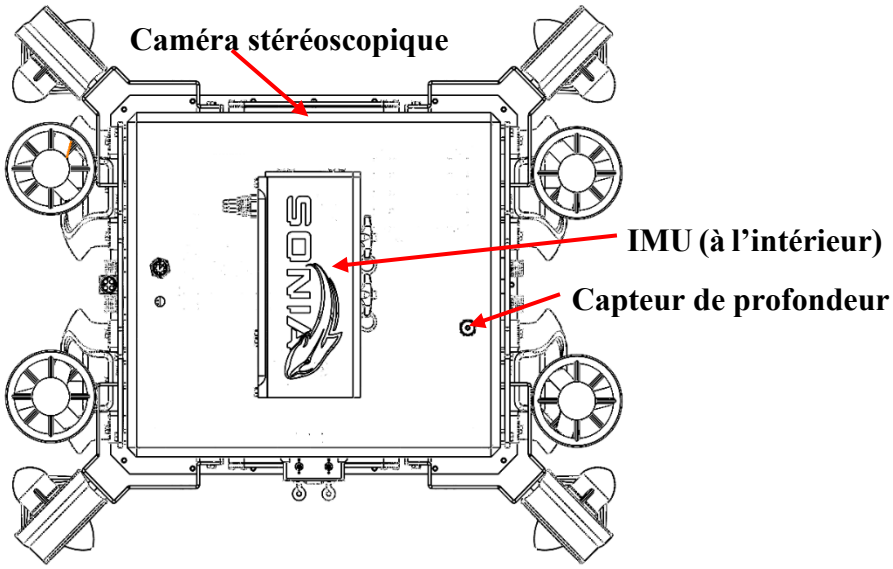

Figure 2 Vue de dessous du LITE1 avec les capteurs

Le LITE1 est le sous-marin le plus récent de S.O.N.I.A. Il a été conçu et fabriqué en 2025 et a participé à sa première compétition la même année. L'idée derrière sa conception était de faire un prototype plus petit et plus léger qui ne serait pas capable de faire la compétition seul, mais qui viendrait soutenir le sous-marin principal en accomplissant les objectifs les plus simples et en lui transmettant des informations. Afin de réduire le poids, certaines fonctionnalités et certains capteurs ne sont pas présents sur le LITE1, notamment le DVL.

2.2 Capteurs

Cette section présente la liste des capteurs utilisés dans les sous-marins qui sont pertinents pour ce projet.

2.2.1 IMU

Les 2 prototypes actuels utilisent le VN-100 (VectorNav, [s d]) de VectorNav. Il s'agit d'une centrale inertielle et d'un système de référence d'attitude et de cap. Celui-ci combine accéléromètres, gyroscopes et magnétomètres triaxiaux afin de fournir des données d'accélération et de rotation en 3 dimensions à haute fréquence. Il communique avec l'ordinateur de bord par le port série. Ce capteur sert de référence pour la vitesse de rotation et l'orientation du sous-marin. Cependant, malgré la grande précision de cet appareil, celui-ci est difficile à calibrer correctement dans ce contexte d'utilisation. En effet, les magnétomètres doivent être recalibrés pour chaque utilisation qui se situe à plus d'une centaine de kilomètres de la calibration précédente. Pour cela, il faut faire tourner l'IMU dans tous les axes, ce qui est difficile dans le cas présent puisque ce capteur ne peut pas être sorti facilement du sous-marin. Il faut donc tourner dans tous les sens les sous-marins qui peuvent faire plus de 40 kilos, ce qui n'est pas évident et qui affecte négativement la qualité de la calibration. Tout cela cause une dérive des données qui proviennent de l'IMU qui impacte la rotation du sous-marin.

2.2.2 DVL

L'AUV8.1 utilise le Pathfinder (Teledyne marine, [s d]) de Teledyne afin de connaitre sa vitesse par rapport au fond de l'eau. Cette information est calculée en utilisant l'effet doppler : le capteur envoie une onde sonore vers le fond, puis mesure la fréquence du rebond. La déformation entre le signal envoyé et celui reçu permet de connaitre la vitesse du sous-marin. Ce capteur transmet ses informations jusqu'à l'ordinateur par Ethernet. Le Pathfinder est également de grande précision, mais le problème vient de l'information fournie. Puisque le

DVL calcule la vitesse et que le contrôle a besoin de connaitre la position, il faut intégrer les données reçues afin d'obtenir la position actuelle du sous-marin, ce qui est fait implicitement par l'EKF. Cela veut donc dire que la moindre erreur cause un décalage permanent dans la position calculée. Il est possible que le DVL soit incapable de calculer la vitesse dans certaines conditions, comme lorsque celui-ci n'est pas orienté vers le fond de la piscine. Il est également possible que la communication entre l'ordinateur et le capteur soit temporairement interrompue. Dans ces cas-là, ou toute autre situation où le système de contrôle n'a pas la vitesse du sous-marin, il y aura un décalage entre la position réelle du sous-marin et celle calculée, puisque les déplacements ne seront pas pris en compte.

Il est important de noter que le LITE1 n'a pas de DVL et se base sur les données d'accélération linéaires afin de connaitre sa position. Puisque pour obtenir cette information à partir de l'accélération il faut faire une double intégration, cela ne fait qu'empirer le problème de dérive de déplacement.

2.2.3 Capteur de profondeur

Les 2 sous-marins utilisent des capteurs de pression afin de connaitre leur profondeur. L'AUV8.1 utilise le ISD4000 (Impact Subsea, [s d]) d'Impact Subsea et le LITE1 utilise le Bar02 (Blue Robotics, [s d]) de Blue Robotics. L'utilisation de 2 capteurs différents s'explique par une différence importante de poids, de taille et de prix. Afin de réduire le poids du LITE1, un capteur plus petit et léger a été choisi. Ces 2 capteurs communiquent à l'ordinateur, soit par le port série et l'autre par I2C, et fonctionnent sur le même principe : la profondeur du sous-marin par rapport à un point de référence est calculée à l'aide de la pression et de la température de l'eau autour. La valeur obtenue n'a pas de dérive puisque la valeur de référence est mise à jour à chaque essai puisqu'il suffit d'enregistrer la pression à la surface avec le capteur. La profondeur calculée est donc utilisée comme une vérité absolue dans le contrôle actuel du sous-marin.

2.2.4 Caméra stéréoscopique

Afin de pouvoir utiliser l'algorithme VSLAM, les 2 sous-marins utilisent une caméra stéréoscopique afin d'obtenir un nuage de points de l'environnement. L'AUV8.1 utilise la ZED Mini (Stereolabs, [s d]) et le LITE1 utilise la ZED 2i (Stereolabs, [s d]), qui proviennent toutes les deux de Stereolabs. L'utilisation de 2 modèles différents est due à la taille des prototypes : le LITE1 a été conçu pour pouvoir utiliser la ZED 2i, mais l'AUV8.1 était déjà construit lorsque la décision d'ajouter des caméras stéréoscopiques a été prise. Malheureusement, ce modèle-ci ne rentrait pas dans la coque, l'équipe a donc décidé de prendre la ZED Mini qui est plus compacte. Bien que les spécifications des 2 caméras ne soient pas identiques, elles communiquent toutes les 2 par USB et accomplissent la même fonction dans le cadre de ce projet, c'est-à-dire de permettre le fonctionnement de l'algorithme SLAM. La perception de profondeur des caméras repose sur le décalage entre les images fournies par les 2 capteurs présents sur l'appareil. En connaissant les spécifications des capteurs ainsi que la distance entre ceux-ci, il est possible de calculer la distance entre la caméra et les objets devant celle-ci.

2.3 Autres technologies des prototypes

Les 2 sous-marins actuels utilisent le Jetson AGX Xavier (NVIDIA, [s d]) de NVIDIA comme ordinateur de bord. C'est lui qui le cerveau du système et qui est responsable de tous les calculs, incluant l'algorithme VSLAM et le Contrôle. Le Xavier est un ordinateur complet dans un format compact. Il possède un CPU ARM 64 bits avec 8 cœurs, un GPU NVIDIA et 64 Go de mémoire DDR4. Il roule le Jetpack 5.1.5 (NVIDIA Developer, [s d]) qui est une version modifiée d'Ubuntu 20.04 fournie par NVIDIA pour ce type d'ordinateur. CUDA 11.4.19 est également inclus, ce qui est essentiel afin de faire fonctionner l'algorithme VSLAM. Finalement, beaucoup de libraires sont directement compatibles avec les ordinateurs Jetson, comme NVIDIA Isaac ROS Visual SLAM, une libraire de VSLAM, ainsi que le kit de développement logiciel des caméras Stereolabs. Il est prévu dans un futur proche que le club

passe au NVIDIA Jetson AGX Orin qui est la génération suivante de ce type d'ordinateur. L'Orin est plus puissant que le Xavier, roule un Jetpack et une version de CUDA plus récents et offre plus de compatibilité. Cependant, tout ce projet sera complété sur les Xavier actuels puisqu'il est impossible de savoir quand le club sera en mesure d'obtenir des Orins.

2.4 Autres équipements

2.4.1 *Tether*

Le *tether* est un rouleau de fil d'Ethernet allant de 50 à 120 mètres de longueur qui permet de lier la communication Duckbox vers les prototypes. À cause des contraintes de communications dans les milieux marins, le moyen de communication avec l'ordinateur de bord est par fil Ethernet. Le câble est hybride comprenant une extrémité RJ45 standard pour connexion à la Duckbox et un connecteur SubConn de MacArtney, conçu pour l'utilisation dans l'eau, pour la connexion au sous-marin. Cette configuration assure une communication fiable entre la Duckbox et le prototype en milieu subaquatique.

2.4.2 Duckbox

La Duckbox est un composant servant de point central de communication avec les prototypes durant toutes les phases opérationnelles. Elle est composée d'un routeur, d'un ordinateur, d'un commutateur réseau (switch) et d'une batterie portable. Cet ensemble met en place un réseau local permettant d'établir une connexion à partir des ordinateurs personnels vers les prototypes parce qu'une communication sans fil est impossible à établir pour un prototype qui est submergé dans l'eau : les ondes radio ne sont pas favorisées pour la communication parce que ces dernières sont absorbées dans l'eau. La Duckbox fait également office de point d'accès à Internet, permettant de récupérer les mises à jour des projets hébergés sur la plateforme GitHub.

2.4.3 Chickenbox

La Chickenbox est utilisée comme source d'alimentation pour le prototype lors des essais réalisés hors de l'eau. Son objectif est de remplacer les batteries et de fournir aux prototypes une tension constante de 16 V. Ce dispositif permet de travailler sur les prototypes sur de longues périodes à l'extérieur de l'eau sans avoir à surveiller en continu le niveau de charge des batteries.

2.4.4 Ordinateurs portables

Le club possède des ordinateurs portables de travail pour la programmation et tout autre travail de club. Ils sont aussi utilisés pour communiquer aux prototypes et de travailler directement sur l'ordinateur de bord. Tous les ordinateurs roulent sur Ubuntu 20.04 ou 22.04.

2.5 Facteurs externes dans les prototypes

Les prototypes AUV8.1 et LITE1 possèdent deux batteries et une batterie respectivement. Ces batteries offrent une autonomie aux sous-marins d'environ deux à trois heures. Il est à noter qu'un faible niveau de charge des batteries peut affecter la performance du système en raison des pics de courant demandés par les moteurs. De plus, les deux sous-marins sont entièrement fabriqués en aluminium et ont été testés pour des opérations inférieures à 10 mètres. Pour finir, le champ magnétique généré par les huit moteurs électriques des sous-marins peut influencer le magnétomètre dans l'IMU et, par conséquent, altérer les mesures.

2.6 Facteurs externes dans l'environnement

Les environnements dans lesquels le prototype se retrouve peuvent affecter la navigation de plusieurs façons. Les capteurs présents sur le prototype et qui contribuent à la navigation ont des limitations au niveau des conditions des eaux et de la localisation. Ces conditions peuvent varier, de la profondeur des milieux, la propreté de l'eau, la forme du fond, des piscines à

l’intérieur, la température et d’autres conditions de plus. Ces limitations peuvent causer des fausses données des capteurs ou des erreurs de transmission de données. Donc, la localisation est un élément à analyser avant une opération pour obtenir les meilleures performances ou d’où moins comprendre l’impact que celui-là possède sur les prototypes.

Pour le DVL, les ondes sonores transmises peuvent être affectées par une concentration significative de bulles présentes dans l’eau causée par des grand vagues. Ces bulles brisent les ondes transmises ou, dans les cas où ces derniers arrivent à pénétrer, modifient leur vitesse sonore. Ce problème affecte le facteur d’échelle de l'effet Doppler. De plus, la présence des algues dans certains milieux aquatiques ne permet pas au DVL de déterminer le fond absolu et peut causer des mesures de vitesse inexactes. Pour finir, dans les eaux salées, soit une concentration de plus de 35 ppt, l’absorption des ondes sonores dans l’eau augmente et réduit la capacité d’altitude du DVL. Donc, le prototype ne pourrait pas opérer à certaines profondeurs dépendant de la salinité (Teledyne marine, [s d]).

Pour l’IMU, le magnétomètre interne est capable de détecter non seulement le champ magnétique terrestre, mais aussi les champs magnétiques générés par des objets autour. En fonctionnement normal, l’IMU s’appuie sur le champ terrestre pour déterminer le cap. Cependant, des perturbations magnétiques peuvent survenir et affecter la précision des mesures. Dans le contexte de prototypes aquatiques, les grandes infrastructures représentent des sources potentielles de perturbations. En effet, pendant les opérations dans les piscines intérieures, il est possible de noter des perturbations magnétiques parce que ces derniers sont dans de grandes infrastructures, habituellement métalliques, et qui possèdent beaucoup d’équipement électrique. Le manuel du modèle de l’IMU utilisé, Vectornav VN-100, indique trois modes de gestion du cap permettant de réduire l’effet des perturbations et minimiser les erreurs dans les données. Il s’agit d’utiliser le cap absolu lorsque le champ magnétique mesuré est presque entièrement le champ terrestre, utiliser le cap relatif lorsque le cap absolu devient

peu fiable en raison des perturbations et utiliser le cap intérieur dans les environnements fortement perturbés comme les piscines intérieures. Ces trois modes permettent à l'IMU de fournir des données fiables en fonction de l'environnement d'opérationnel (VectorNav, [s d]).

CHAPITRE 3
MÉTHODOLOGIE DE TEST

3.1 Types de tests

3.1.1 Test à sec

Les tests à sec sont essentiels, car ils permettent de récolter des données sans avoir accès à une piscine. En effet, la location d'un bassin suffisamment grand et profond pour les tests est couteuse. De plus, il faut planifier plusieurs jours à l'avance afin que le bassin soit libre et aussi que suffisamment de membres de S.O.N.I.A. soient disponibles afin de rendre le test possible.

Ces tests permettent d'obtenir toutes les données reliées au VSLAM ainsi qu'à l'IMU. Cependant, le DVL ne fonctionne que dans l'eau, il est donc impossible d'avoir des données de vitesse, et puisque le contrôle actuel se base uniquement sur la vitesse mesurée par le DVL pour calculer la position, ces données ne sont également pas disponibles. Il est aussi impossible de vérifier la stabilité du sous-marin puisque celui-ci est incapable de se déplacer hors de l'eau.

Afin de réaliser ces tests, le sous-marin est placé sur un chariot avec tout le matériel nécessaire à son fonctionnement, comme la Duckbox et la Chickenbox. Le chariot est ensuite déplacé à la main en suivant un itinéraire prédéterminé et les données sont enregistrées dans un ou plusieurs *ROS Bag*.

3.1.2 Test en piscine

Les tests en piscines permettent de confronter directement la problématique, de recueillir les données de dérive et de valider l'implémentation de VSLAM dans le système de contrôle dans le but de corriger la dérive. Comme le prototype est conçu pour le milieu aquatique, les conclusions seront plus basées sur les tests en piscines. Contrairement au test à sec, tous les

capteurs sont fonctionnels, incluant le DVL. Le système de contrôle est donc très utilisé pendant ces tests afin d'analyser le comportement du prototype et d'évaluer la performance de la navigation lors de l'intégration du VSLAM. Les données enregistrées pendant ces tests sont celles de l'ancien et nouveau système pour pouvoir effectuer une comparaison. Ces tests permettent aussi de vérifier les conditions requises pour confirmer la validité de VSLAM dans l'eau puisque la vision de la caméra stéréoscopique est affectée par l'eau ainsi que par les variations de luminosité, ce qui cause des erreurs comme la distorsion des images. Cependant, pour la réalisation des tests avec le prototype principal AUV8.1, certaines contraintes doivent être respectées concernant le type de piscine utilisé. La qualité de la piscine influence fortement les performances du DVL sur le prototype, un capteur conçu pour opérer en milieu océanique. Les piscines idéales sont celles creusées avec un fond en béton, qui offrent des surfaces dures et stables permettant une propagation adéquate des ondes acoustiques. À l'inverse, les piscines hors terre ou celle dont les parois contiennent un revêtement mou sont à éviter. Dans ces environnements, les ondes sonores émises par le DVL sont perturbées, ce qui dégrade significativement la qualité des mesures et peut rendre le prototype instable. Pour le prototype LITE1, l'absence de DVL élimine cette contrainte : les piscines hors terre ou à parois souples peuvent donc être utilisées sans impact majeur sur la performance. Cependant, lorsque des tests doivent être réalisés simultanément ou comparativement avec les deux prototypes, il est nécessaire d'utiliser une piscine répondant aux exigences du AUV8.1.

3.2 Procédure de test

Lors des tests, les données sont enregistrées à l'aide du système d'enregistrement de ROS2, les *ROS Bags*. Ce sont des fichiers sur lesquels sont enregistrées toutes les données envoyées sur les *topics* choisis. Ces fichiers peuvent ensuite être rejoués afin d'extraire les données voulues dans des fichiers CSV qui sont ensuite intégrées dans des tableaux Excel pour être analysées. De plus, les tests ont été filmés afin de permettre une comparaison entre les résultats des *Bags* et la réalité.

CHAPITRE 4
SLAM

4.1 Définition et types

SLAM – Simultaneous Localization And Mapping

Dans le monde actuel de la robotique, les systèmes dotés de capacités de navigation autonome gagnent en popularité. L'une des exigences essentielles pour les robots autonomes est la capacité de naviguer dans un environnement inconnu tout en évitant les obstacles et en atteignant leur destination en toute sécurité (Qiao, Guo, & Li, 2024). Bien que ce soit un vaste sujet, c'est l'aspect de la navigation qui suscite ici l'intérêt. La localisation et la cartographie sont les éléments clés qui permettent aux robots de comprendre leur environnement et de connaître leur propre position (Qiao, Guo, & Li, 2024). L'algorithme SLAM est une approche largement utilisée pour construire une carte d'un environnement et estimer la position du robot à l'intérieur de celui-ci (Qiao, Guo, & Li, 2024) (Das, 2020) (Yan, Guorong, Shenghua, & Lian, 2009).

Les algorithmes de SLAM se composent généralement de deux éléments : la cartographie et la localisation. Le composant de cartographie construit une carte de l'environnement à partir des données des capteurs, tandis que le composant de localisation estime la position du robot à l'intérieur de cet environnement (Qiao, Guo, & Li, 2024). Il existe plusieurs types d'approches en matière de localisation, telles que les approches basées sur les filtres, les approches basées sur les graphes et les filtres à particules. Ces trois types seront explorés plus en détail dans le présent document.

En lien avec le projet S.O.N.I.A., le cas d'utilisation idéal pour ces implémentations se situe dans des environnements sans accès GPS et dépourvus de tout contexte environnemental.

4.1.1 SLAM basé sur les filtres

Les approches basées sur les filtres font référence à l'utilisation des filtres de Kalman (KF). Un KF est un algorithme mathématique utilisé pour l'estimation d'état dans les systèmes linéaires. L'algorithme fonctionne en deux étapes : la prédiction et la mise à jour. Durant la phase de prédiction, le filtre estime le prochain état du système à partir des données passées. La phase de mise à jour corrige ensuite cette estimation à l'aide des nouvelles données fournies par les capteurs. Ce processus est récursif et s'exécute de manière continue (Das, 2020). Une version de ce filtre exploite le filtre de Kalman étendu (EKF), qui excelle dans les estimations linéaires appliquées à des systèmes non linéaires (Yan, Guorong, Shenghua, & Lian, 2009). Le filtre utilisant l'EKF est également appelé un programme à covariance complète, car il emploie un vecteur d'état augmenté et une matrice de covariance pour déterminer les corrélations entre les états et leurs caractéristiques. Le compromis de l'utilisation d'un EKF ou de filtres similaires réside dans le fait que leur conception d'estimation linéaire les rend adaptés uniquement aux systèmes faiblement non linéaires. La marge d'erreur augmente de façon exponentielle dans le cas de systèmes fortement non linéaires. Malgré ce compromis, cet algorithme demeure largement utilisé (Yan, Guorong, Shenghua, & Lian, 2009).

Il est important de mentionner que, dans la plupart des cas généraux — particulièrement dans le cadre du projet AUV de S.O.N.I.A. — le problème à traiter est non linéaire. Cela découle du fait que l'AUV opère avec six degrés de liberté (6-DOF) et qu'il navigue en milieu aquatique. Lorsqu'on travaille sur un plan 2D, il est possible d'approximer la linéarité. Malheureusement, puisque l'AUV évolue dans l'eau, il est impossible de garantir des mouvements strictement bidimensionnels. De plus, en raison de l'absence de GPS, il n'existe aucune référence externe.

Un exemple d'algorithme, le MSCKF, illustre comment un SLAM basé sur les filtres peut être mis en œuvre pour des systèmes non linéaires, tout en intégrant potentiellement une composante visuelle (Mourikis & Roumeliotis, 2007). Bien que ce système soit performant,

l'aspect visuel présente des limites dans des environnements instables, notamment à cause des variations d'intensité lumineuse. La solution efficace consiste à intégrer un capteur visuel-inertiel, c'est-à-dire une caméra stéréo combinée à une IMU. Ce type de capteur permet de rendre la composante visuelle de l'algorithme SLAM beaucoup plus stable, améliorant ainsi la qualité des résultats (Qiao, Guo, & Li, 2024)

Les deux principales limites des algorithmes SLAM basé sur les filtres sont : le coût computationnel et la forte incertitude due au biais des capteurs. Concernant le coût computationnel, on peut estimer la complexité du EKF-SLAM à $O(N^2)$, où N représente le nombre d'observations. La forte incertitude provient du fait que toutes les estimations dépendent de la précision des capteurs utilisés. En tenant compte du biais inhérent que peuvent présenter ces capteurs, on risque d'obtenir un biais final exponentiellement amplifié, puisqu'il se cumule sur l'ensemble des capteurs employés (Yan, Guorong, Shenghua, & Lian, 2009). Cet aspect d'incertitude élevée peut toutefois être réduit par une calibration rigoureuse et l'utilisation de techniques de vision odométrique (Visual Odometry) permettant d'obtenir un référentiel local plus stable (Qiao, Guo, & Li, 2024).

4.1.2 SLAM basé sur les filtres à particules

L'approche basée sur les filtres à particules repose sur une hypothèse de base selon laquelle le modèle de prédiction est erroné. Les filtres à particules adoptent une approche probabiliste pour déterminer la localisation, en intégrant des modèles imparfaits et des capteurs imparfaits au moyen de lois probabilistes telles que la règle de Bayes. En tant qu'une des approches les plus anciennes en matière de localisation, elle a contribué de manière importante à faire progresser la résolution du problème SLAM. En particulier, elle a permis de traiter le problème du robot kidnappé, où le robot doit retrouver sa position dans un contexte d'incertitude globale (Thrun, 2002). Bien que cela puisse sembler impressionnant, cette approche présente une limite importante : l'environnement est supposé entièrement connu (c'est-à-dire qu'une carte est déjà donnée). De plus, elle est toujours « fausse » dans la mesure où elle repose sur des modèles

probabilistes supposés erronés dès le départ. Ainsi, elle est excellente pour l'approximation et la prédiction, mais peu fiable pour la certitude absolue.

Un algorithme emblématique utilisant les filtres à particules est FastSLAM, dont le nom est assez explicite. La complexité de calcul de cet algorithme est la plus faible parmi ceux mentionnés dans ce document, soit O(M log n), où M représente le nombre de particules et n le nombre de repères (Thrun, 2002). Dans le cas d'utilisation du projet, M croît généralement avec n, de sorte que la complexité peut être réécrite sous la forme O(n log n) (Thrun, 2002) (Hanenko, Storchak, Shlianchak, Vorohob, & Pitaichuk, 2025). Il convient également de noter que cette méthode est affectée par la dimensionnalité : elle a du mal à traiter les problèmes à haute dimension, en raison du nombre de particules qui augmente de façon exponentielle pour représenter correctement un état (Thrun, 2002). Le projet S.O.N.I.A. est considéré comme un système à haute dimensionnalité avec ses 13 états.

Contrairement au SLAM basé sur les filtres de Kalman, cette approche peut être modulaire. L'algorithme SLAM peut être divisé en deux parties : le *front-end* et le *back-end*. Le *front-end* consiste à collecter les données des capteurs (extraction de caractéristiques) et à établir les associations de données entre les différentes observations (suivi de caractéristiques à court terme, bouclage à long terme). Le *back-end*, quant à lui, traite les observations associées pour générer les corrélations nécessaires à la cartographie et à la localisation (Cadena et al., 2016). Les algorithmes de particules modernes, tels que FastSLAM, tirent parti de cette structure modulaire en utilisant des filtres EKF dans le *front-end*, afin d'améliorer la précision des estimations. Cette approche permet également de reconstruire l'environnement de manière plus efficace (Thrun, 2002).

4.1.3 SLAM basé sur les graphes

L'approche basée sur les graphes consiste en la « [construction d'un] graphe dont les nœuds représentent les poses du robot ou des repères, et dans lequel une arête entre deux nœuds

encode une mesure de capteur qui contraint les poses connectées » (Grisetti, Kümmerle, Stachniss, & Burgard, 2010). Le cœur du problème consiste à trouver une configuration des nœuds la plus cohérente possible avec les mesures, ce qui revient à résoudre un grand problème de minimisation d'erreur (Grisetti, Kümmerle, Stachniss, & Burgard, 2010). À l'instar des filtres à particules, cette approche tire également parti de la séparation entre le *front-end* et le *back-end* du SLAM. Cette distinction est particulièrement importante ici, car le SLAM basé sur les graphes se concentre principalement sur la partie *back-end* du processus (Grisetti, Kümmerle, Stachniss, & Burgard, 2010).

Un avantage majeur de cet algorithme est qu'il est conçu pour s'appuyer entièrement sur l'environnement observé et qu'il fonctionne efficacement dans des systèmes non linéaires. Le compromis réside dans le fait qu'il s'agit de l'algorithme ayant la complexité la plus élevée parmi toutes les approches SLAM mentionnées, soit $O(n^3)$ (Hanenko, Storchak, Shlianchak, Vorohob, & 11 Pitaichuk, 2025). Bien que cette complexité soit importante, la précision temporelle du système est également l'une des meilleures, car une fois l'origine du graphe définie, toutes les références qui y sont reliées deviennent extrêmement précises (Durrant-Whyte & Bailey, 2006). Un facteur clé contribuant à cette précision est l'étape de fermeture de boucle (loop closure), absente des approches basées sur les filtres. La fermeture de boucle fait référence à la capacité de l'algorithme à reconnaître un repère déjà visité, même si celui-ci n'apparaît pas exactement dans la même position qu'auparavant. Cela permet à l'algorithme d'optimiser l'ensemble du graphe afin que les nouvelles informations demeurent cohérentes avec les données existantes (Hanenko, Storchak, Shlianchak, Vorohob, & Pitaichuk, 2025).

Un exemple bien connu de cette implémentation est ORB-SLAM, qui intègre des techniques de vision odométrique (Visual Odometry) ainsi que des capteurs visio-inertiels (Visual-Inertial Sensors) (Grisetti, Kümmerle, Stachniss, & Burgard, 2010).

4.2 Comparaison et solution potentielle

Lors de l'analyse de ces méthodes, plusieurs points de comparaison peuvent être relevés, comme le mentionnent (Hanenko, Storchak, Shlianchak, Vorohob, & Pitaichuk, 2025). En comparant le SLAM basé sur les filtres, le SLAM à particules et le SLAM basé sur les graphes, on observe divers compromis. Dans tous les cas, le EKF-SLAM fonctionne mieux pour des systèmes linéaires avec un bruit gaussien, mais cette situation est rarement (voire jamais) rencontrée en pratique. Même dans ce cas, le EKF-SLAM demeure généralement le meilleur choix pour les systèmes de petite échelle, en raison de son faible coût computationnel. Toutefois, à mesure que le système grandit, il devient moins efficace que les autres options. Pour les systèmes de grande échelle, les approches basées sur les graphes (Hanenko, Storchak, Shlianchak, Vorohob, & Pitaichuk, 2025) et celles à particules offrent de meilleures performances en termes de précision (Hanenko, Storchak, Shlianchak, Vorohob, & Pitaichuk, 2025) (Das, 2020).

En observant l'échelle et la complexité des systèmes, le SLAM basé sur les filtres tend à mieux fonctionner au départ, mais perd rapidement en efficacité à mesure que la complexité des calculs et la taille du modèle augmentent. Les approches basées sur les graphes et à particules sont beaucoup mieux adaptées aux systèmes à grande échelle. Bien que les filtres à particules soient techniquement plus rapides, avec une complexité d'O(n log n), leur précision demeure inférieure à celle du SLAM basé sur les graphes. C'est à ce stade que les avantages et les compromis entre les différentes approches deviennent apparents. Comme la majorité des systèmes modernes sont de grande taille et que l'objectif principal est la précision, la combinaison EKF + SLAM basé sur les graphes tend à être l'approche privilégiée dans la plupart des cas d'utilisation.

Le système idéal, sans limites de puissance de calcul ni de mémoire, combinerait les trois approches. Il utiliserait des filtres EKF dans le *front-end* du SLAM — ce que font déjà ORBSLAM et FastSLAM — et une combinaison de SLAM basé sur les graphes et de SLAM

à particules dans le *back-end*. L'idée serait d'employer un EKF pour fusionner les données capteurs et effectuer l'association de données, de laisser le SLAM basé sur les graphes, cartographier les environnements inconnus et suivre les repères, et d'utiliser le SLAM à particules dans les environnements connus afin de filtrer la précision des prédictions, tout en revenant au graphe en cas d'incertitude.

En ce qui concerne le projet S.O.N.I.A., la meilleure approche, compte tenu des informations disponibles et des contraintes de ressources, serait d'utiliser une solution EKF + graphe. D'une part, comme un EKF fait déjà partie du système de contrôle, il n'est pas nécessaire d'en concevoir un nouveau. D'autre part, le choix du SLAM basé sur les graphes s'explique par la volonté de réduire la dérive au fil du temps, rendant ainsi la précision des données absolument essentielles.

Il n'est pas possible de développer une solution personnalisée dans les limites de temps et de ressources imposées au projet, et il est donc nécessaire de recourir à des cadres existants. L'objectif est de se rapprocher du meilleur scénario théorique décrit précédemment. La première option est ORB-SLAM3, particulièrement intéressant pour ses aspects visuo-inertiels et sa réputation dans l'industrie (Campos, Elvira, Rodríguez, Montiel, & Tardós, 2021). Ensuite, Nav2 est une option intéressante grâce à son intégration directe avec ROS2 (« Nav2 — Nav2 1.0.0 documentation », s.d.). Enfin, NVIDIA_ISAAC_VISUAL_SLAM est conçu pour fonctionner sur le GPU plutôt que sur le CPU, ce qui est avantageux puisque les prototypes utilisent des ordinateurs de bord NVIDIA (« Isaac ROS Visual SLAM — isaac_ros_docs documentation », s.d.). D'autres solutions pertinentes sont mentionnées dans (Merzlyakov & Macenski, 2021), notamment OpenVSLAM, qui présente également un certain intérêt.

Cependant, bien que Nav2 soit convivial et facile à intégrer, il ne correspond pas au cas d'utilisation pour deux raisons principales : premièrement, il est conçu pour des environnements 2D (« Nav2 — Nav2 1.0.0 documentation », s.d.) ; deuxièmement, il repose

sur un SLAM purement basé sur l'EKF pour la localisation (« robot_localization wiki — robot_localization 2.7.7 documentation », s.d.), alors qu'ORB-SLAM3 et Isaac Visual SLAM sont basés sur les graphes. Bien que ces deux derniers soient comparables, une étude menée par NVIDIA démontre qu'Isaac est légèrement supérieur sur les systèmes NVIDIA (« Isaac ROS Visual SLAM — isaac_ros_docs documentation », s.d.). De plus, même si ORB-SLAM3 dispose d'une implémentation ROS2, il n'a pas été testé sur ROS2 Humble, la version utilisée par le projet (Jung, 4 août 2022/2025). Le même argument s'applique à OpenVSLAM, dont la compatibilité avec ROS2 n'a pas encore été solidement démontrée (« ROS Package — OpenVSLAM documentation », s.d.). Ainsi, la solution optimale serait d'utiliser Isaac Visual SLAM, car il existe de solides preuves de compatibilité et une intégration fluide avec l'infrastructure déjà en place.

4.3 Impact sur le système actuel

L'intégration de l'algorithme de VSLAM n'a pas démontré d'impact visible sur le système de l'AUV8.1 parce que ce dernier répond aux exigences minimales de la libraire de NVIDIA_ISAAC_VISUAL_SLAM. En effet, après avoir effectué l'implémentation et des phases de tests, aucun impact négatif n'a été remarqué sur le système. Cependant, il serait intéressant d'observer la réaction du système si l'algorithme tournait en parallèle avec un modèle d'IA nécessaire aux opérations du prototype lors de la compétition, étant donné que les deux utilisent la carte graphique du système. Ce test, qui a pour but d'analyser les limites de l'usage de l'algorithme en parallèle avec d'autres ressources du système, n'a pas été réalisé parce qu'il n'était pas possible d'obtenir une phase de test avec un modèle d'IA pendant la réalisation du projet.

4.4 Critère d'analyse

Les critères qui vont permettre de confirmer la conformité de l'algorithme reposent sur la fiabilité de l'odométrie fournie par VSLAM ainsi que la précision du nuage des points généré

à partir des images de la caméra. Le premier critère permet d'évaluer dans quelle mesure l'implémentation de cet algorithme améliore l'état actuel du système. En effet, si l'odométrie produite par l'algorithme n'est pas suffisamment fiable, il devient peu pertinent de considérer VSLAM comme une solution efficace pour la correction de la dérive.

Par ailleurs, si le nuage de points, par rapport au sous-marin, n'est pas bien défini, c'est-à-dire si les corrections des points résultant de la fermeture de boucle sont de faible qualité. Alors, VSLAM ne fonctionne pas de façon optimale. Cette dégradation peut être causée, entre autres, par une qualité insuffisante des images ou des mauvaises conditions environnementales.

4.5 Données en entrée

4.5.1 Caméra stéréo

Les lentilles droite et gauche fournissent des paires d'images permettant d'identifier des points clés correspondants. De plus, grâce à la capacité de la caméra stéréo à fournir des informations de profondeur, il est possible d'estimer la distance entre la source et chaque point clé, ce qui permet ensuite de déterminer la position de ce point dans un espace 2D ou 3D. L'ensemble de ces points clés constitue la cartographie et permet de localiser le prototype dans son environnement. (MathWorks, [s d] ; NVIDIA Isaac ROS, [s d])

4.5.2 IMU

Le système SLAM peut fonctionner entièrement à partir des caméras stéréo, en s'appuyant sur l'odométrie visuelle (VO) générée par le flux vidéo des lentilles droite et gauche. Cependant, dans les conditions où ces données ne sont pas suffisamment précises pour estimer la pose, comme en cas de distorsions d'image, de mauvaise luminosité ou de surfaces aux propriétés

optiques défavorables, les données issues de l'IMU permettent d'obtenir une estimation de pose plus précise et plus stable. (MathWorks, [s d] ; NVIDIA Isaac ROS, [s d])

4.6 Données en sortie

À partir des entrées, l'algorithme de VSLAM est capable de produire une estimation d'odométrie et une représentation de l'environnement par des points en 3D autour de la caméra.

4.6.1 Odométrie visuelle

L'odométrie visuelle est le résultat de l'analyse des entrées de l'algorithme de VSLAM : les images des caméras et les données de l'IMU. L'algorithme utilise une libraire cuVSLAM de NVIDIA pour traiter les images stéréo directement sur le GPU du système. Le cuVSLAM détecte des points distinctifs dans les images pour générer des points correspondants dans l'espace 3D. Ces points visuels sont intégrés dans une carte interne qui permet d'estimer le mouvement dans l'espace de la caméra dans l'espace. Concrètement, il calcule le mouvement en comparant les points qu'il a déjà observés avec ceux détectés en temps réel. L'ensemble de ces étapes permet de produire l'odométrie visuelle qui fournit la position et l'orientation de l'observateur par rapport à son environnement (Anon, [s d]).

4.6.2 Nuage de points

Le nuage de points représente une disposition des points générée par l'algorithme. La distribution de points reflète la structure de l'environnement observé par la caméra, chaque point étant défini par des coordonnées en trois dimensions. Figure 3 illustre un exemple de propagation des points dans l'espace de l'atelier du club S.O.N.I.A, tel que montré dans la Figure 4. Les points en blanc représentent un historique des points déjà observé par l'algorithme tandis que les points rouges indiquent la fermeture de boucle (*loop closure*).

La fermeture de boucle arrive lorsque la caméra revient dans un environnement déjà visité et reconnait les points déjà existants dans la carte. Lorsque ce processus survient, le cuVSLAM corrige les erreurs accumulées au cours de l'odométrie visuelle et réajuste la position de certains points récents en fonction des points précédemment observés. Ce processus améliore la cohérence de la carte et permet d'obtenir une trajectoire globale plus précise.

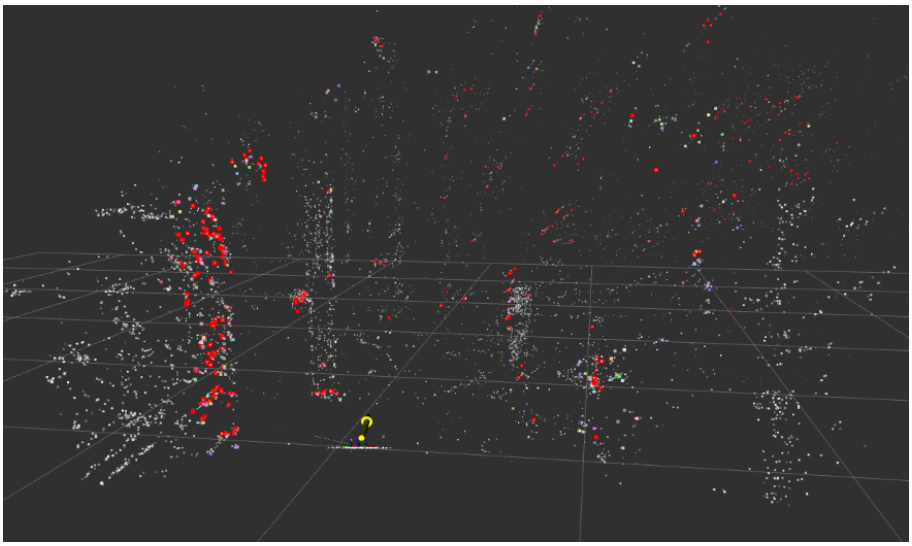

Figure 3 Point cloud de l'espace des ateliers

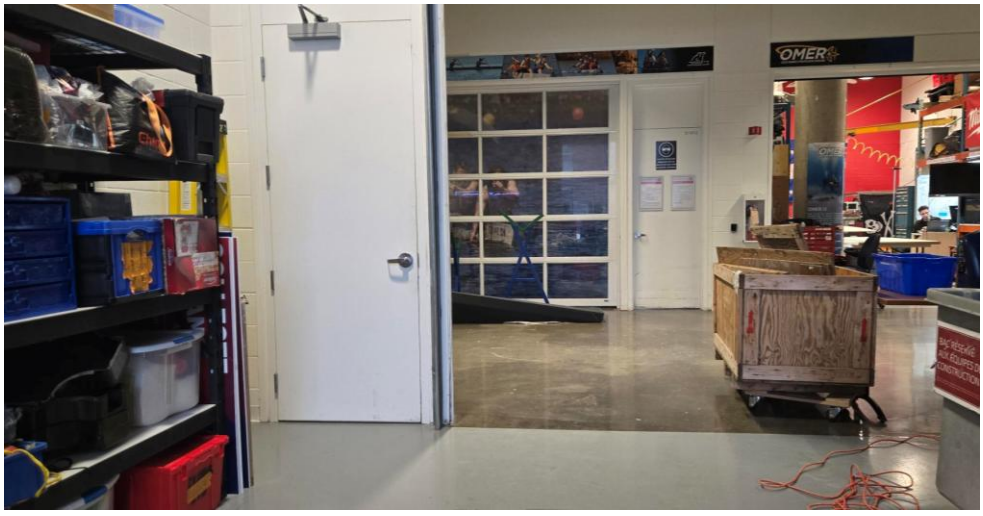

Figure 4 Vue de l'intérieur de l'atelier du club de S.O.N.I.A.

CHAPITRE 5
SYSTÈME DE CONTRÔLE

5.1 Définition

Le système de contrôle du sous-marin comprend un filtre de karman étendu (EKF) et un MPC adaptatif. La combinaison des deux approches est conçue pour des systèmes non linéaires et qui sont soumis aux variations et perturbations de l'environnement. Avec un tel système, le sous-marin est capable d'évoluer dans les 6 dégrées de liberté, définis par la position et l'orientation. Le contrôle repose sur un estimateur d'état généré par la fusion de capteurs de pression, d'IMU et du DVL du sous-marin dans l'EKF. En tenant compte du centre de gravité du sous-marin, le MPC utilise ensuite l'état estimé pour calculer et envoyer les PWM aux huit moteurs pour orienter et positionner le sous-marin vers l'état désiré.

5.2 Implémentation actuelle de proc_nav (EKF)

Pour l'implémentation actuelle de proc_nav, celui-ci possède comme sortie treize états du sous-marin : trois positions, quatre orientations dans un quaternion, trois vitesses linéaires et trois vitesses angulaires. Ces états proviennent des trois capteurs du sous-marin, comme montré dans le Tableau 1, et sont envoyés au MPC. Ce dernier traite les états reçus pour, ensuite, envoyer des signaux aux moteurs pour le déplacement à effectuer et retourner les forces en newtons des moteurs au proc_nav pour la prochaine itération. Le présent projet permet d'analyser trois cas d'utilisation distincts grâce à l'intégration de nouveaux concepts et équipements, notamment l'algorithme VSLAM :

1. Un cas d'utilisation IMU + DVL : VectorNav VN-100 avec le DVL Pathfinder de Teledyne.

2. Un cas d'utilisation IMU + IMU : VectorNav VN-100 combiné avec l'IMU intégré à la caméra stéréoscopique.
3. Un cas d'utilisation IMU + IMU + DVL : les deux IMUs ainsi que le DVL.

Tableau 1: Les types d'états du filtre de Kalman

Type d'état	Source(s)
Orientation	IMU
Position	Capteur de pression, Calculé
Vitesse linéaire	DVL
Vitesse angulaire	IMU

Dans le système de contrôle, il y a trois parties concernant l'obtention des treize états, comme montré dans la Figure 5 : il y a le prétraitement des mesures venant des capteurs, le filtre de Kalman étendu et le rassemblement des données d'états dans un objet bus de Simulink. En ce qui concerne les états, le système se fie entièrement sur l'orientation provenant de l'IMU pour sa précision importante grâce au magnétomètre. Cependant, l'orientation en z peut acquérir de la dérive dépendamment des conditions de l'environnement. Pour la position, le système favorise le capteur de pression pour les déplacements dans l'axe de z et, pour les deux autres, ils sont déterminés en effectuant des calculs d'intégrale des vitesses linéaires sur les trois axes mesurés par le DVL.

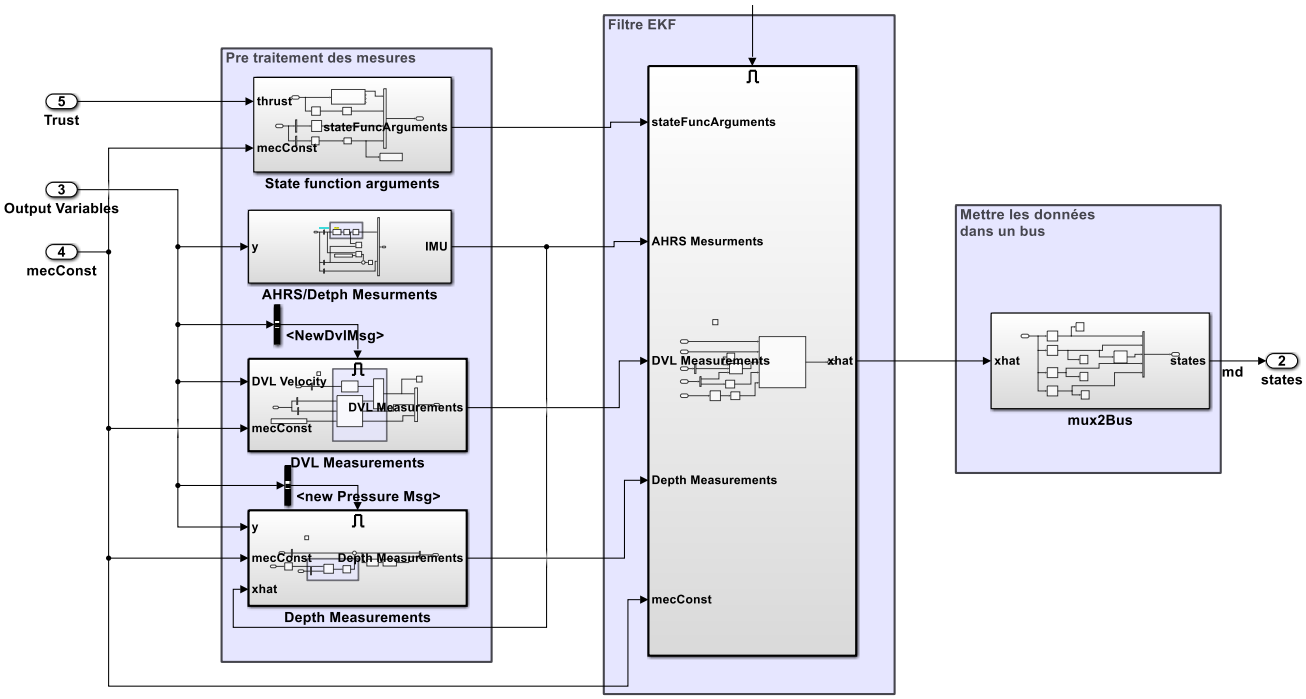

Figure 5 Présentation des entrées du proc_nav dans Simulink

5.2.1 Cas d'utilisation IMU + DVL

Le principe le de ce cas d'utilisation, qui est le cas d'utilisation actuel pour l'AUV8.1, est de montrer une importance significative envers les mesures des capteurs IMU et DVL dans l'EFK parce que ces derniers sont les sources d'une bonne partie des treize états du sous-marin sortie par l'EFK. Cette approche est sur lequel le système initial a été conçu pour et il est fiable en temps d'opérations normal, d'où moins pour la compétition auquel le sous-marin participe.

5.2.2 Cas d'utilisation IMU + IMU

Le cas d'utilisation IMU + IMU représente une nouvelle approche intéressante pour le système. Il a été introduit après l'acquisition de caméras stéréoscopiques de Stereolabs : la ZED Mini pour l'AUV8.1 et la ZED2i pour le LITE1. Les deux caméras possèdent des IMUs intégrés dont le système de contrôle pourrait bénéficier. Elles offrent une source supplémentaire pour les états mesurés par l'IMU principale, permettant ainsi d'améliorer la précision et, indirectement, la performance globale du système. Ce cas d'utilisation vise à démontrer le fonctionnement du système de contrôle en absence du DVL à cause des deux raisons suivantes : la fréquence de données provenant du DVL n'est pas consistante et le nouveau prototype, le LITE1, ne possède pas de DVL. Donc, la source des vitesses linéaires proviendrait des accélérations linéaires fournies par les deux IMUs, plutôt que par le DVL, en effectuant un calcul d'intégration sur les mesures de l'accéléromètre du capteur et une double intégration pour en déduire la position par rapport au temps.

5.2.3 Cas d'utilisation IMU + IMU + DVL

Ce cas d'utilisation est la combinaison des deux cas d'utilisation en utilisant les 3 capteurs rigoureusement dans le filtre de Kalman étendu. C'est une approche seulement possible avec l'AUV8.1 parce qu'il possède les trois composants nécessaires. Elle est intéressante comme approche parce que, en théorie, la possibilité d'avoir des redondances dans le système rendrait le système de contrôle beaucoup plus robuste que le système actuel. En effet, comparé aux sources des mesures définies dans le Tableau 1, l'implémentation suivante vient ajouter une source additionnelle de données pour chaque type d'état du sous-marin.

5.3 Implémentation actuelle de proc_control (MPC)

Actuellement, le contrôle reçoit en entrée 13 états qui correspondent à un objectif à atteindre. Cet état est généré par un autre programme qui est responsable de diviser chaque mouvement en plusieurs mouvements plus courts afin de tracer une trajectoire. Ce sont ces plus petits déplacements qui sont ensuite envoyés au contrôle. Une fois que l'EKF a calculé les 13 états actuels, ceux-ci sont envoyés au MPC. Celui-ci compare ensuite ces données à l'objectif, puis calcule les prochains déplacements nécessaires pour atteindre l'objectif. Le MPC prend également en entrée les constantes physiques du sous-marin, comme la position des moteurs, le centre de masse et de poussée et l'inertie entre autres. Ces constantes sont utilisées afin de prédire la réaction du sous-marin afin de prédire les déplacements futurs. Les forces que doit appliquer chaque moteur sont également calculées, puis transformées en signal à envoyer aux moteurs. Les forces calculées sont également renvoyées au EKF afin qu'elles soient incluses dans le prochain calcul des états actuels.

5.4 Intégration idéale du VSLAM dans l'EKF

5.4.1 Données idéales du VSLAM

Idéalement, l'algorithme VSLAM fonctionne parfaitement et retourne les 13 états du contrôle avec une très grande précision et aucune erreur. Ces données peuvent donc directement être utilisées dans le filtre de Kalman étendu avec des covariances très faibles, car elles seraient plus précises que les capteurs actuels. Les données les plus importantes sont la position en X et Y ainsi que la rotation en Z, car c'est sur ces états que la dérive est présente. Avec un VSLAM idéal, ces données seraient corrigées, car l'algorithme fourni des données de position et d'orientation qui permettent de corriger la dérive déjà présente.

5.4.2 Modifications idéales de l'EKF pour intégrer le VSLAM

Avec un algorithme VSLAM idéal, l'intégration dans le filtre de Kalman étendu est très simple. Il faut simplement ajouter les 13 états retournés par VSLAM dans une entrée de l'EKF après avoir ajusté le référentiel de la caméra afin que les résultats soient sur le référentiel du contrôle.

Pour cela, il faut d'abord ajuster les données de position en soustrayant les coordonnées de la caméra par rapport au centre du sous-marin aux données du VSLAM après avoir appliqué la rotation retournée aux coordonnées. Puisque le référentiel du VSLAM place l'axe Z vers le haut, il faut également inverser les rotations sur les axes Z et Y. Cela est fait en multipliant les valeurs associées du quaternion par -1. Pour les vitesses linéaires, en plus d'appliquer la même rotation, il faut également retirer les vitesses linéaires qui sont générées par les rotations et l'effet de levier à cause de la distance au centre du sous-marin. Ces vitesses sont calculées en fonction des vitesses de rotation autour des axes ainsi que des coordonnées de la caméra par rapport au centre du sous-marin. Les vitesses de rotation doivent uniquement être inversées en Y et en Z afin d'utiliser le référentiel du contrôle.

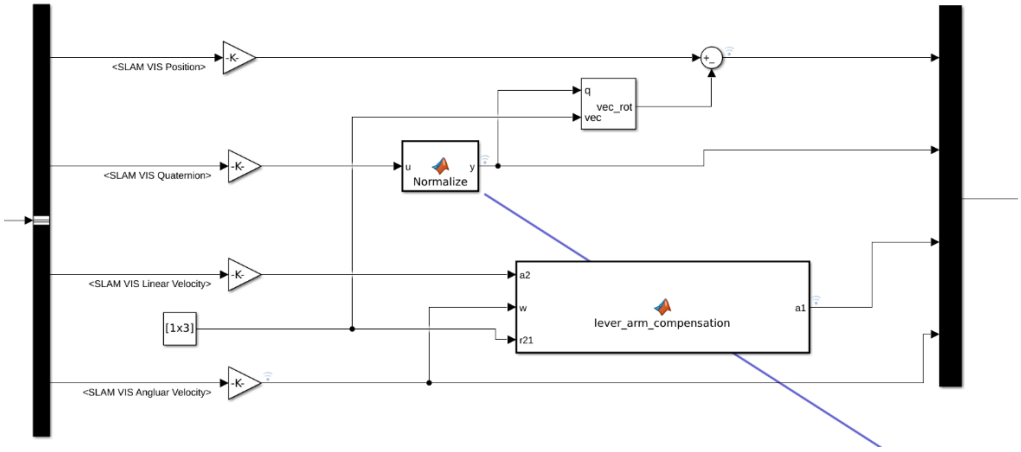

Figure 6 Implémentation de l'ajustement du référentiel

Puisque les données de l'algorithme seraient parfaites, les covariances associées seraient très basses afin qu'elles aient un grand impact sur la sortie de l'EKF.

5.4.3 Coût des modifications idéales

Les couts d'une telle intégration sont très faibles, puisque le système EKF-MPC est déjà en place. L'ajout des données du VSLAM dans le filtre de Kalman étendu est très simple et puisque les données sont parfaites, les covariances peuvent être très basses sans avoir besoin de faire plusieurs tests afin de déterminer les bonnes valeurs.

5.5 Intégration réaliste du VSLAM dans l'EKF

5.5.1 Données réelles sélectionnées du VSLAM

En réalité, les données provenant du VSLAM ne peuvent pas être considérées comme parfaites. En effet, l'algorithme se base sur les changements détectés par la caméra. Dans des conditions réelles, il est possible qu'aucun changement significatif ne soit enregistré par le VSLAM, ce qui entraine des erreurs dans les données envoyées au filtre de Kalman.

5.5.2 Changement réaliste de l'EKF pour intégrer le VSLAM

Intégration réaliste est très proche de l'intégration idéale, puisqu'il suffit également d'ajouter les données du VSLAM en entrée au EKF après avoir ajusté le référentiel. Cependant, puisque les données sont imparfaites, les covariances doivent être ajustées afin de prendre en compte l'imprécision de chacun des 13 états qui sont renvoyés par l'algorithme. L'ajustement de ces covariances est difficile, car il n'est pas possible d'obtenir une valeur de façon mathématique. La meilleure façon de faire consiste à essayer différentes valeurs afin de trouver celles qui donnent les meilleurs résultats.

5.5.3 Risques possibles et calibrations nécessaires

Ces essais peuvent être longue et l'accès à une piscine étant difficile, il est possible que le temps de test disponible ne soit pas suffisant pour déterminer les meilleures covariances

possibles. Il faut aussi potentiellement calibrer le VSLAM dépendant des performances observées. Ces ajustements prennent également du précieux temps de test, il faut donc limiter le plus possible les pertes de temps.

5.6 Impacts sur le MPC

Ce projet n'a aucun impact sur la partie MPC du contrôle. En effet, seul le filtre de Kalman est impacté. Après que celui-ci a calculé les 13 états actuels, ceux-ci sont envoyés au MPC afin de calculer les déplacements futurs. Puisque les données du VSLAM vont dans l'EKF, elles ont uniquement un impact sur le calcul de l'état actuel, ce qui n'impacte aucunement le MPC.

5.7 Impacts attendus sur la dérive

Puisque l'algorithme VSLAM retourne les 13 états du contrôle, toutes les valeurs de celui-ci devraient être corrigées en cas de dérive. En supposant que VSLAM soit en mesure de suivre les déplacements et les rotations du sous-marin, toute la dérive générée par les pertes de données du DVL sera corrigée par les données de vitesse linéaire et de position de VSLAM et la dérive causée par l'IMU sera corrigée par la rotation fournie par l'algorithme.

CHAPITRE 6
RÉSULTATS

6.1 Tests à sec

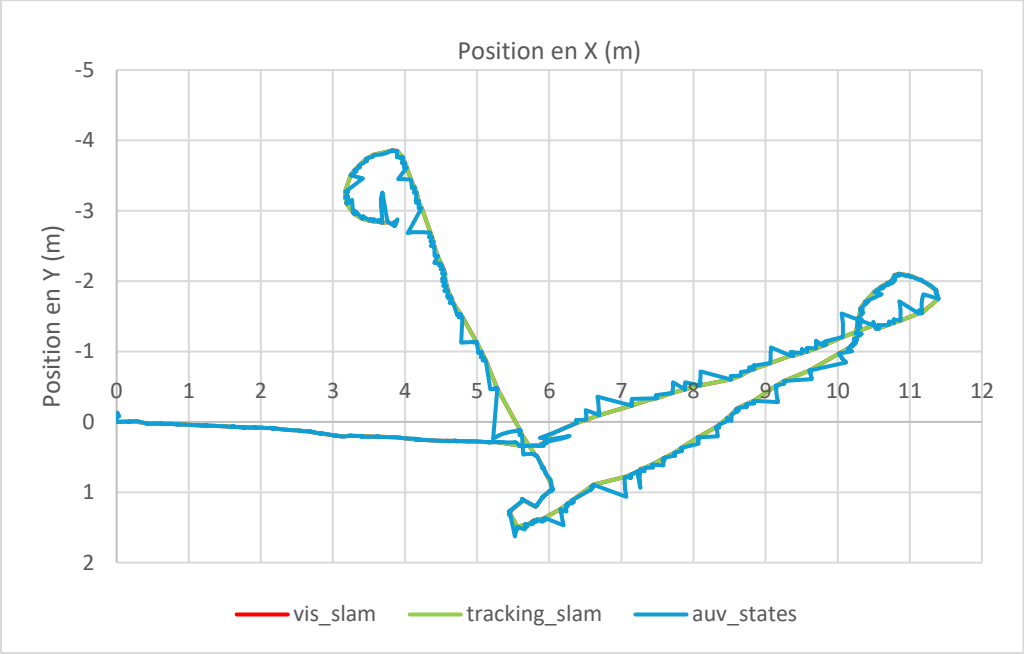

Figure 7 Position du sous-marin vue du dessus lors du test à sec 1

La première Figure 7 montre la position retournée par les deux sorties du VSLAM, *vis_slam* et *tracking_slam*, ainsi que la position calculée par le contrôle *auv_states* durant le premier test à sec. Le mouvement effectué est un déplacement vers l'avant de 6m, une rotation de 90 degrés vers la gauche, un mouvement de 8m vers l'avant, une rotation de 180 degrés dans le sens antihoraire, un autre déplacement de 8m, une rotation vers la droite de 90 degrés, un mouvement de 6m vers l'avant et une rotation de 180 degrés dans le sens antihoraire. Les données montrent que le premier déplacement a bien été mesuré par le VSLAM. La première rotation est visible sur le graphique, mais n'a pas le même angle que la réalité. Le déplacement suivant ainsi que le demi-tour et le mouvement vers le point de la première rotation sont

relativement bien mesurés, mais la position du sous-marin est décalée d'environ un mètre par rapport à la position de la première rotation, alors que ce n'est pas le cas en réalité. La rotation de 90 degrés vers la droite, le mouvement de 6m vers l'avant ainsi que la rotation finale sont également bien détectés. Il est important de noter que l'axe y est inversé, car le référentiel du sous-marin est inversé. Ceci s'appliquera à plusieurs graphiques dans ce chapitre.

À partir de la première rotation, des pics sont visibles dans les données d'*auv_states*. Ceux-ci sont causés par un décalage entre l'orientation de l'IMU et du VSLAM. Puisque les données du DVL ne sont pas disponibles lors des tests à sec, le calcul de position se base entièrement sur la vitesse calculée par le VSLAM. Cependant, ce calcul s'effectue à plus haute fréquence que le VSLAM, ce qui oblige le contrôle à utiliser les dernières données en attendant les nouvelles données. Bien que la vitesse vienne uniquement du VSLAM, l'orientation est également fournie par l'IMU. En absence de données du slam, le contrôle se base entièrement sur ce capteur afin de connaitre son orientation, mais utilise les dernières données du VSLAM pour les autres informations, comme la position et la vitesse. Ces pics pointent donc dans la vraie orientation du sous-marin, puis sont ramenés à la position du VSLAM lorsque de nouvelles données sont reçues. Leur taille et position aléatoire sont dues au fait que ce phénomène n'arrive que lorsque le VSLAM ne fournit pas d'informations, ce qui est dépendant des éléments visibles par la caméra.

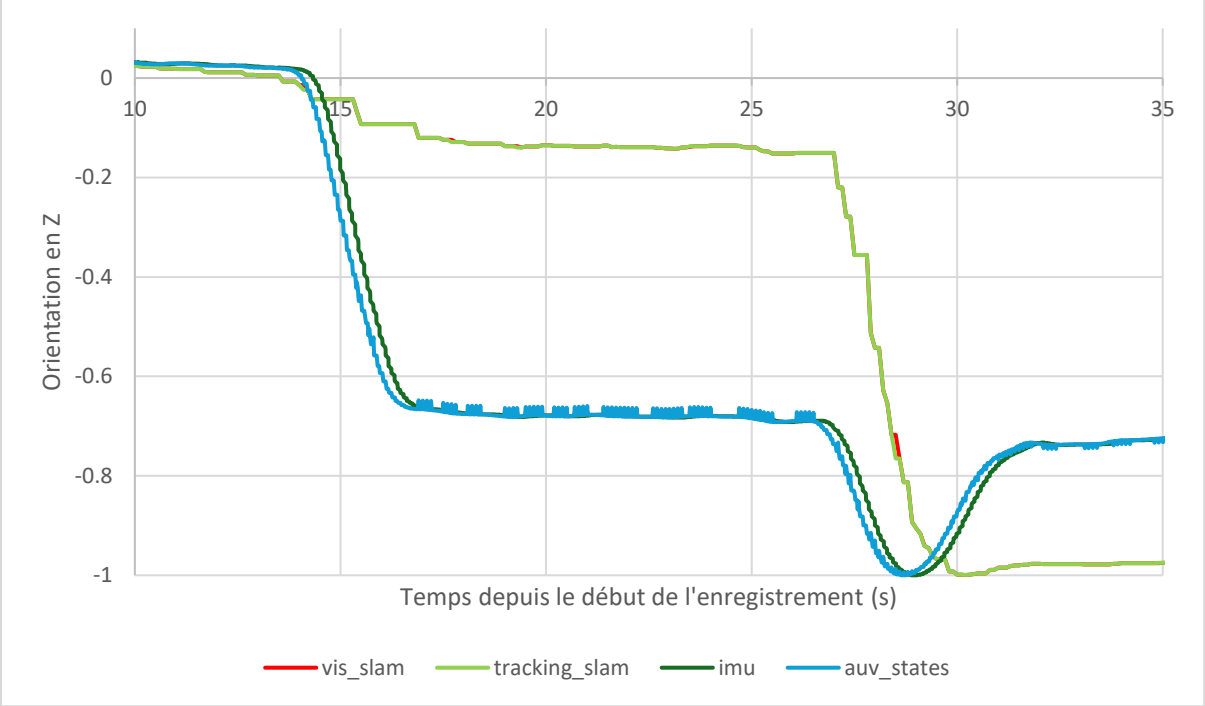

Figure 8 Orientation en Z en fonction du temps lors du test à sec 1

La Figure 8 montre l'orientation du sous-marin pendant le test. Au début, toutes les données sont proches de zéro. À environ 15 secondes, le sous-marin fait une rotation de 90 degrés. L'IMU ainsi que *auv_states* mesure correctement celle-ci, puisque les deux se rapprochent de 0.707 qui est la valeur d'une rotation de 90 degrés dans un quaternion. Cependant, le VSLAM ne mesure pas correctement le changement d'orientation, ce qui cause une différence entre les deux valeurs. À partie de ce moment, des pics verticaux sont visibles dans les données d'*auv_states*. Ceux-ci sont causés par la réception des données du SLAM par le contrôle. Celui-ci tente d'inclure ces informations très différentes dans le calcul de la rotation, mais, puisque les covariances du VSLAM sont beaucoup plus élevées que celles de l'IMU, la rotation retourne rapidement à la valeur de la centrale inertielle. Cela peut également être vu dans les données au-delà de 30 secondes. Après une rotation de 180 degrés, la valeur du VSLAM devient plus basse et plus proche de celle de l'IMU. Cela a pour effet de réduire la taille des pics et d'inverser leur sens.

Figure 9 Position du sous-marin retournée par VSLAM vue du dessus lors du test à sec 2

L'objectif du deuxième test à sec était d'observer l'impact d'un environnement connu par rapport à un environnement inconnu. Pour cela, le sous-marin a effectué le même trajet deux fois. Celui-ci consistait en un rectangle de 5.5 mètres de long sur 2.5 mètres de large. Lors du premier trajet, le sous-marin a tout d'abord été déplacé autour du parcours afin d'avoir un nuage point initial avant de commencer le trajet. Les résultats montrent que, lors de ce déplacement, l'algorithme VSLAM a bien réussi à suivre les déplacements du sous-marin avec peu d'erreurs. Lors du deuxième trajet, le nuage de point a été réinitialisé juste avant de commencer le déplacement. Les données montrent que celui-ci n'a pas bien été évalué par le VSLAM. Les données diffèrent à partir de la première rotation et la position finale est à plusieurs mètres de la position initiale.

6.2 Tests en piscine

Tous les tests en piscine ont été réalisé au Complexe aquatique Michel-Leduc (Aquadôme).

Figure 10 Piscine de test à Aquadome (https://inscriptionsaquadome.ca/bain-libre)

Le bassin utilisé mesure 25 mètres de long sur 12 mètres de large et mesure 3,4 mètres de profondeur avec une pente d'un côté. Lors des tests, le bassin est entièrement réservé par S.O.N.I.A., il n'y a donc personne d'autre dans la piscine. Lorsque le sous-marin est dans l'eau, il y a toujours un membre du club qui nage à côté afin de s'assurer qu'il ne rentre pas en collision avec un mur en cas de problème.

6.2.1 Sans marqueurs visuels

Tous les résultats présentés dans cette catégorie ont été obtenus lors de tests dans une piscine vide, c'est-à-dire sans objets supplémentaires dans le bassin. Les seuls éléments visibles lors de ce test étaient les murs et le fond ainsi que les lignes de natation flottantes à la surface. Ce manque de marqueurs visuels rend le suivi de la position difficile pour VSLAM, ce qui explique l'instabilité des résultats.

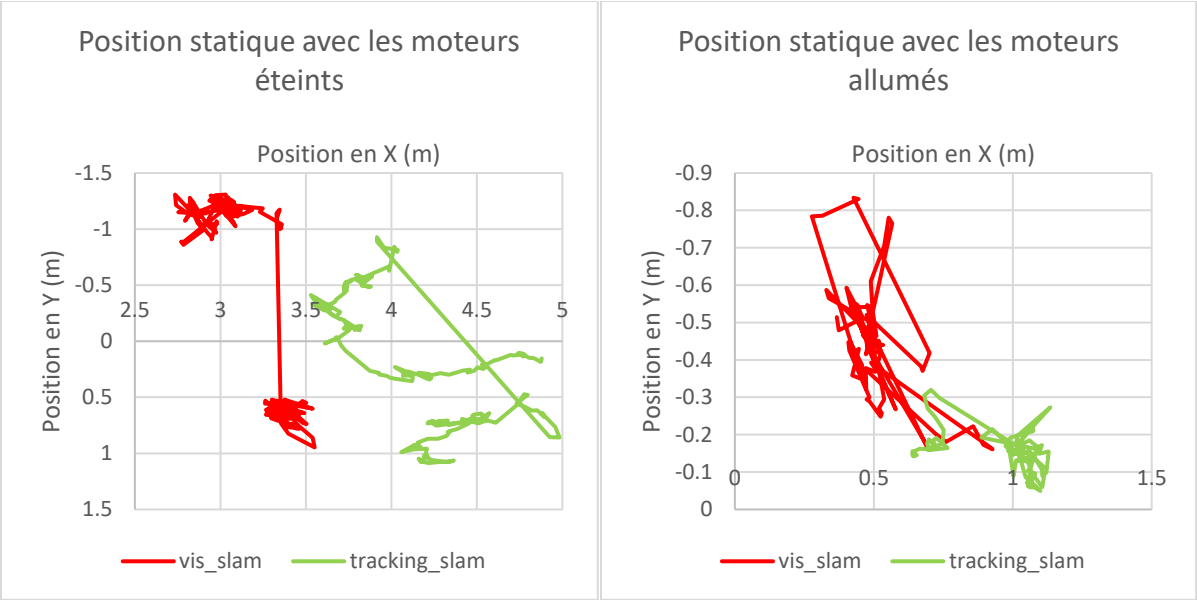

Figure 11 Positions retournées par VSLAM avec le sous-marin statique

Le premier test effectué lors du premier essai en piscine fut d'enregistrer les données envoyées par VSLAM sans qu'il soit connecté au contrôle. Deux enregistrements ont été pris, le premier avec les moteurs éteints et le sous-marin flottant immobile à la surface et le deuxième avec les moteurs actifs et le contrôle essayant de conserver sa position actuelle. Dans les deux cas, aucun mouvement significatif n'a été observé durant les enregistrements. Pourtant, les données montrent des déplacements chaotiques qui font parfois plusieurs mètres de long, ce qui montre que VSLAM ne semble être capable de suivre les mouvements du sous-marin. Un nouveau phénomène qui n'était pas visible dans les tests à sec est aussi très visible dans ces deux graphiques : *vis_slam* et *tracking_slam* ne retournent pas les mêmes résultats. Lors des tests à sec, ces 2 sorties étaient toujours très proches l'une de l'autre, au point que seulement *tracking_slam* est visible sur les graphiques précédents. Mais dans les données de ce test, il est très clair que les deux sorites du VSLAM retournent des résultats complètement différents. La position initiale et finale est différente et il n'y a pas de corrélation entre les mouvements des 2 sorties.

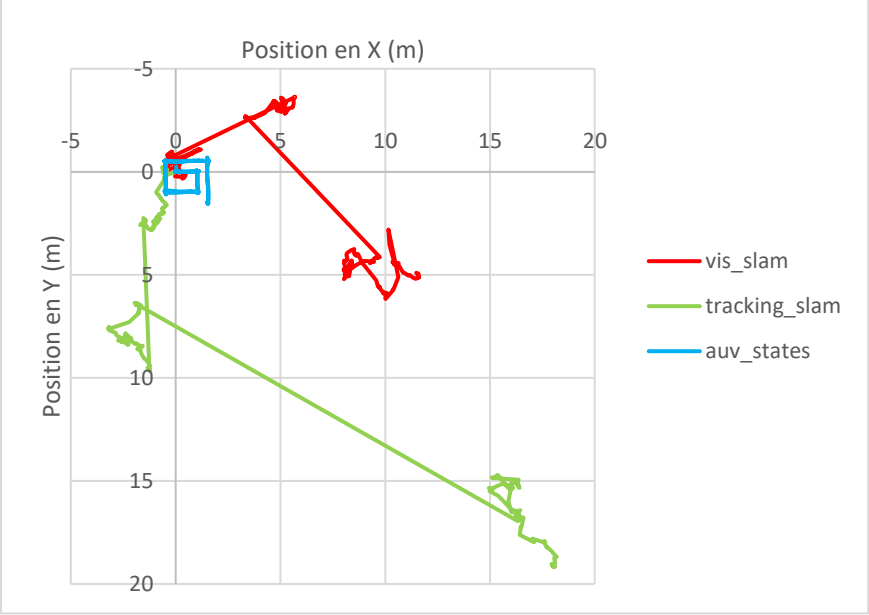

Figure 12 Position du sous-marin vue du dessus lors du test en piscine 1

Ce graphique présentes les résultats du test en mouvement. Comme pour les données précédentes, l'algorithme VSLAM n'est pas connecté au contrôle, il n'a donc pas d'impact sur la position du sous-marin. Pour ce test, le sous-marin devait effectuer une spirale avec des angles de 90 degrés afin d'observer la position enregistrée par le VSLAM lors des déplacements. Ces mouvements sont bien visibles dans les données d'*auv_states*, mais comme lors du test précédent, VSLAM est incapable de suivre les déplacements du sous-marin. Ces résultats montrent l'importance d'avoir des marqueurs visuels supplémentaires dans la piscine lors des tests.

6.2.2 Avec des marqueurs visuels

Lors de ces tests, des obstacles ont été ajoutés dans la piscine afin d'améliorer les performances du VSLAM. Ces obstacles étaient statiques et très visibles sur les parois de la piscine. Ceux-ci sont également les obstacles utilisés lors de la compétition à laquelle S.O.N.I.A. participe chaque année et représente donc bien les conditions réelles d'utilisation.

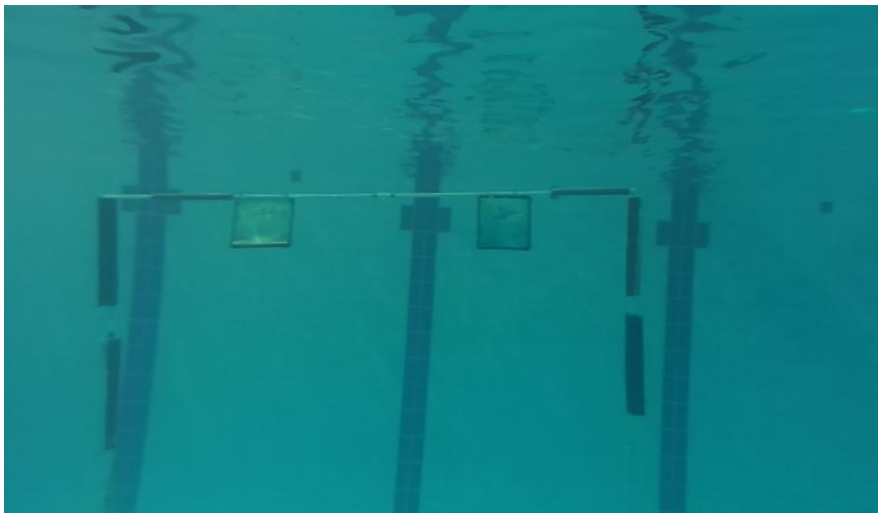

Figure 13 Image d'un obstacle dans la piscine prise depuis la caméra du sous-marin

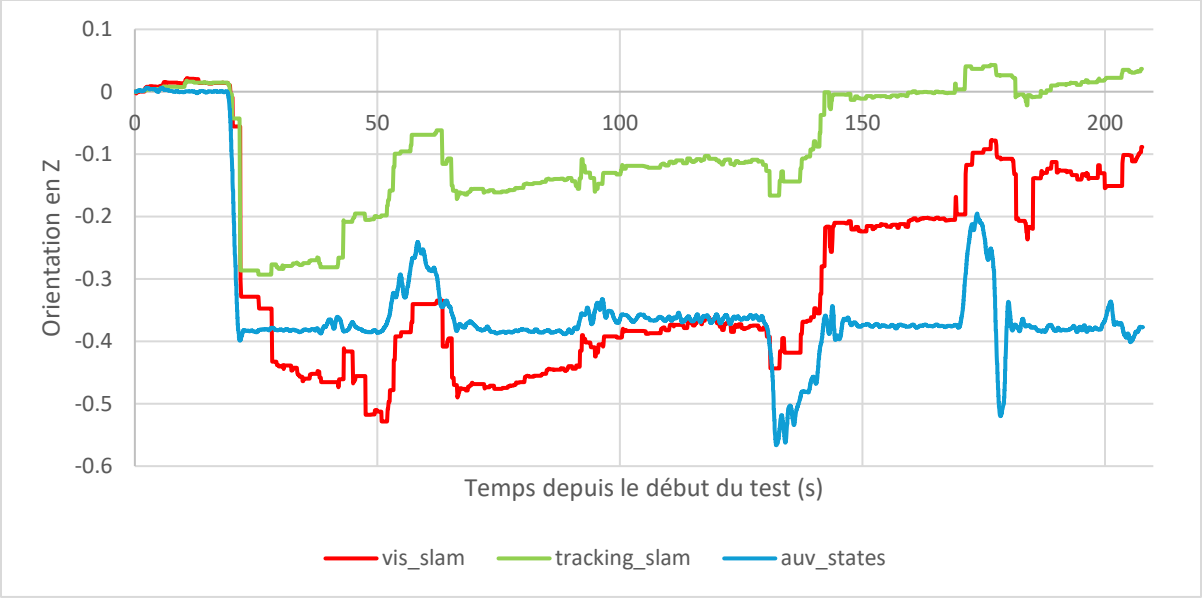

Figure 14 Orientation en Z en fonction du temps lors du test en piscine 2

Ce graphique présente les données d'orientation du sous-marin sur l'axe Z. Au départ, lors des tests, le sous-marin se trouve face à l'obstacle mentionné précédemment. Après quelques secondes, il effectue une rotation de 45 degrés sur l'axe Z, ce qui correspond à une valeur de -0.383 pour l'axe Z du quaternion. Ce mouvement change le champ de vision de la caméra,

mais l'obstacle reste visible, ce qui permet à VSLAM de suivre relativement bien la rotation. Par la suite, le nageur vient perturber le sous-marin en poussant dessus, ce qui vient causer des rotations involontaires. Celles-ci sont clairement visibles dans les données par les pics qui apparaissent dans toutes les données.

Bien qu'il y ait clairement une corrélation entre les trois sources de données, une divergence entre *tracking_slam* et les autres valeurs sont également visibles. Celle-ci commence dès la première rotation du test, mais s'amplifie beaucoup après les mouvements. Selon les résultats obtenus, il est rapidement ressenti que *vis_slam* fournit des données plus précises à court terme et envoie des informations plus fréquemment que *tracking_slam*. Pour ces raisons, c'est *vis_slam* qui a été choisi pour être intégré dans le contrôle. C'est pour cette raison que *tracking_slam* ne sera pas présenté dans les prochains graphiques.

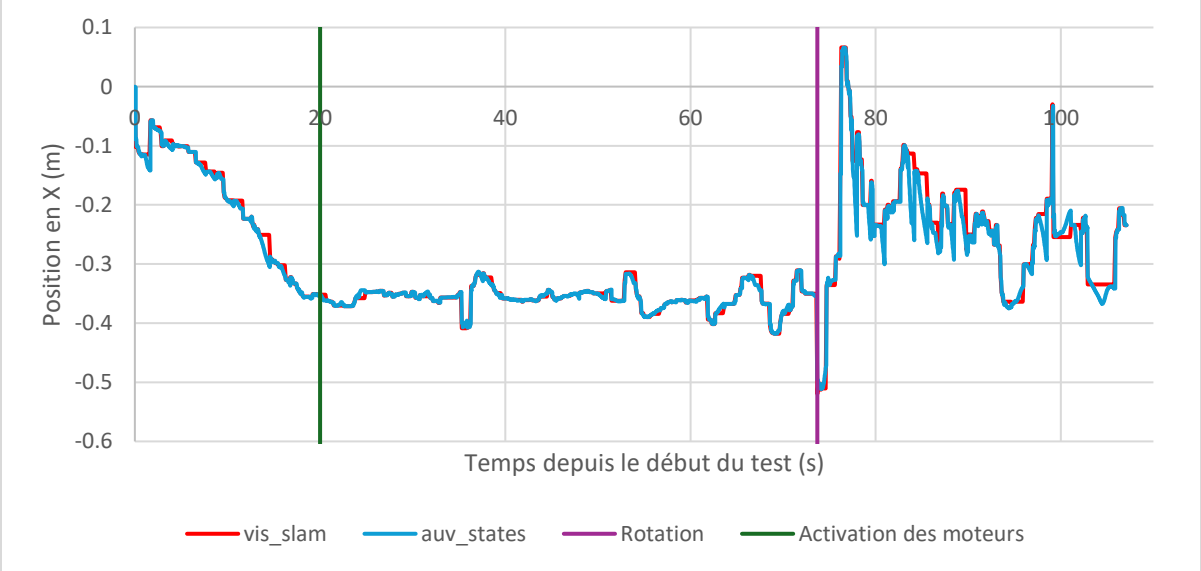

Figure 15 Position en X en fonction du temps pendant le test en piscine 3

Des résultats similaires peuvent être observés pour la position sur l'axe X lors du test suivant qui a eu lieu dans des conditions similaires. Au début de ce test, le sous-marin regarde l'obstacle présenté plus haut. Dans les 20 premières secondes, les moteurs ne sont pas actifs et c'est le nageur qui déplace le sous-marin. À 20 secondes, les moteurs sont allumés et le sous-

marin se stabilise. Puis, 74 secondes après le début du test, le sous-marin fait une rotation de 90 degrés, ce qui fait sortir l'obstacle du champ de vision de la caméra. À partir de ce moment, des pics sont visibles dans les données d'*auv_states*, comme pendant le test à sec 1. La cause est la même, c'est-à-dire que VSLAM n'a pas mesuré correctement la rotation et que la valeur d'orientation en Z de *vis_slam* est différente de celle de la centrale inertielle.

CHAPITRE 7
ANALYSE

7.1 NVIDIA Isaac ROS Visual SLAM

7.1.1 Intégration de l'IMU

Comme mentionné précédemment dans ce rapport, l'utilisation de la bibliothèque *NVIDIA Isaac ROS Visual SLAM* représentait le meilleur scénario pour l'implémentation du VSLAM dans le cadre de ce projet. Il semble toutefois qu'après les tests réalisés, ce n'ait pas été le meilleur choix. La description de configuration indique que l'activation du paramètre *enable_imu_fusion* devrait informer la bibliothèque d'utiliser les données de l'IMU intégré dans la caméra lors du calcul de l'odométrie en sortie. Toutefois, comme le montre l'équation suivante, l'algorithme ne semble pas tenir compte de l'IMU. Cela modifie fondamentalement les attentes envers le système, puisqu'il repose alors exclusivement sur les données stéréoscopiques visuelles. Ce point est davantage expliqué dans Korovko et al. (2025), où l'équation suivante illustre la manière dont l'estimation de pose visuo-inertielle est réalisée :

$$S_{i-1}, S_i =$$

$$\arg\min \left[\left|\left| r^{imu}(S_{i-1}, S_i) \right|\right|^2_{\sum IMU} + \sum_{j=0}^{1} \left|\left| r^{repr}(S_{i-1}, S_i) \right|\right|^2_{\sum VIS} + \left|\left| r^{prior}(S_{i-1}, S_i) \right|\right|^2_{\sum p} \right]$$

Équation 1

Si $r^{\text{repr}} \approx 0$, ce qui représenterait une perte du flux visuel, nous devrions alors nous appuyer uniquement sur les données de l'IMU. En utilisant cette équation, l'algorithme corrigerait les données d'odométrie afin de minimiser l'erreur détectée par l'IMU selon la matrice de covariance Σ_{IMU}. Si l'équation se réduit à n'utiliser que $\| r^{\text{imu}}(S_{i-1}, S_i) \|^2_{\Sigma_{\text{IMU}}}$, l'erreur

s'accumule très rapidement, ce qui amène l'algorithme à ignorer complètement les données et à cesser de publier.

En fin de compte, l'algorithme nécessite un flux visuel pour fonctionner, celui-ci servant de lien entre les poses calculées et le monde réel. Le problème fondamental rencontré est davantage lié à la qualité du flux visuel qu'à sa disponibilité.

7.1.2 Tracking ou Vis

Lors de l'utilisation de la bibliothèque NVIDIA Isaac ROS Visual SLAM, la documentation en ligne pour la version 2.1 ne définit que les topics ROS liés à *tracking_slam* et non ceux liés à *vis_slam*. En examinant le code source ainsi que la version la plus récente de la documentation, il a été déterminé que tous les *topics* associés à *vis_slam* étaient considérés comme des topics de visualisation. Cela est suggéré dans la documentation en ligne d'Isaac ROS cuVSLAM (« cuVSLAM — Isaac ROS », s.d.) et défini de manière plus explicite dans (Mur-Artal & Tardos, 2017). Ce comportement correspond bien à celui qui a été observé lors des tests.

Tracking

Tout ce qui provient du *tracking* contient les données brutes calculées en arrière-plan du VSLAM, ce qui inclut la fusion de capteurs et la fermeture de boucle (*loop closure*). La principale raison pour laquelle l'information provenant de cette source est lente et semble « en retard » par rapport au reste est qu'un volume de traitement beaucoup plus important est nécessaire pour la générer. En fin de compte, ces informations ne sont pas adaptées à ce projet en raison de leur faible fréquence et, puisqu'elles se corrigent continuellement via la fermeture de boucle, elles sont sujettes à des sauts dans les données.

Visualiseur

Tout ce qui provient de *vis* est considéré comme de l'odométrie visuelle. Ces informations sont générées par le *front-end* de l'algorithme SLAM et sont conçues pour l'évaluation visuelle. Cela signifie qu'elles n'appliquent pas en continu les optimisations que l'on retrouve dans le *tracking*, bien qu'elles intègrent la fermeture de boucle et certaines corrections de manière asynchrone. Étant donné que ces données sont conçues pour l'évaluation visuelle, leur fréquence est plus élevée que celle du *tracking*, et les corrections y sont intégrées progressivement au fil du temps plutôt que de manière directe. Ce comportement les rend idéales pour le projet, où l'intégration avec un EKF est nécessaire.

Les informations définissant ces deux sources concordent également avec les résultats observés lors des différents tests.

7.2 Impact des données visuelles répétitives et de la qualité des images

Lors de l'analyse des résultats obtenus au fil de différents tests, les données visuelles se sont révélées systématiquement incohérentes dans l'eau en l'absence d'obstacles, tandis qu'elles étaient nettement plus stables sur terre. L'hypothèse initiale supposait des problèmes liés à l'intégration de l'IMU dans l'algorithme VSLAM, mais cela a été infirmé à la section 7.1.1. En ce qui concerne l'algorithme, la conclusion a été que la qualité du flux visuel entrant dans le VSLAM avait un impact significatif.

Durant les tests terrestres, l'environnement présentait des structures variées, ce qui permettait au mécanisme de fermeture de boucle de fonctionner comme prévu, menant à une localisation adéquate. Plus précisément, les observations montrent que la qualité de la localisation lors d'une boucle est liée à la qualité de la cartographie initiale de l'espace avant l'exécution de la boucle, comme observé lors du test à sec 2.

Au départ, il était difficile de comprendre pourquoi le prototype refusait d'être stable dans l'eau. En utilisant des outils comme RVIZ2 pour visualiser le nuage de points généré par le

module de cartographie du VSLAM, les données montraient qu'aucune position ou localisation cohérente ne pouvait être extraite à partir des observations. En d'autres termes, l'algorithme VSLAM éprouvait des difficultés à définir son environnement de manière stable.

Bien que cette conclusion ait été atteinte, deux facteurs en sont à l'origine : la répétition de motifs et la qualité de la vidéo.

7.2.1 Motifs dans les données visuelles

L'un des problèmes liés à la localisation, particulièrement dans les systèmes SLAM basés sur des graphes, provient de la manière dont ils utilisent les *keyframes* observées et comparent les positions 3D enregistrées dans leur mémoire. Lorsqu'il s'agit de données complexes ou présentant des variations uniques, ce mécanisme fonctionne très bien. Même dans des zones qui se répètent, si cette répétition ne concerne qu'une petite région d'une carte plus vaste, l'algorithme parvient généralement à se localiser correctement. Les difficultés apparaissent lorsque la zone répétitive constitue une portion significative de la carte existante. Dans ce cas, le comportement observé est que l'algorithme tente de se localiser là où il croit se trouver et commence à réécrire la carte en conséquence.

Le test à sec 2 illustre parfaitement ce phénomène, car la zone utilisée est un corridor bordé de grandes portes de garage. En avançant dans ce corridor, les parois se répètent approximativement tous les cinq mètres. Sans prendre le temps de cartographier en détail la zone de départ, l'algorithme perd sa position dès que le prototype se tourne vers l'une des portes de garage.

Cet effet est amplifié lors des essais en piscine. Le sol et les parois présentent des lignes répétitives, et les murs sont entièrement carrelés de blanc. Ainsi, chaque fois que le prototype tente de cartographier son environnement, le VSLAM éprouve de grandes difficultés à déterminer sa position dans la carte. Même la tâche de tracer manuellement la carte de

l'environnement en incluant au moins deux murs n'est pas utile, car le bruit visuel empêche la détection des *keyframes* significatives. La seule situation où la cartographie devient semi-stable est lorsqu'un obstacle contrastant est ajouté, c'est-à-dire lorsqu'un objet est placé de manière à offrir un arrière-plan clairement distinct en couleur, comme lors du test en piscine 2.

Lors de l'un des derniers essais, le prototype s'est montré beaucoup plus stable lorsqu'il était orienté directement vers l'obstacle (dans ce cas, la porte présentée dans la Figure 13 Image d'un obstacle dans la piscine prise depuis la caméra du sous-marinFigure). Il faut souligner que l'algorithme VSLAM fonctionnait mieux lorsque l'environnement local n'avait pas été largement cartographié. Cela s'explique par le fait que, durant la cartographie, dès que le prototype ne regardait plus l'obstacle, il perdait son point de référence dans un environnement ambigu. En revenant vers l'obstacle, il le cartographie à nouveau au lieu de se relocaliser, ce qui entrainait la présence de plusieurs obstacles sur la carte mémoire, ce qui cause une confusion lorsque VSLAM tente de positionner le sous-marin.

Globalement, cela démontre que le VSLAM dépend fortement de la présence de plusieurs repères uniques qu'il peut distinguer de l'environnement, particulièrement lorsqu'il repose uniquement sur les données visuelles. Malheureusement, ce n'était pas le cas pour l'ensemble des tests réalisés en milieu aquatique, mais il était essentiel de mettre en évidence cette limite, car le cas d'usage final (la compétition à laquelle participe le club étudiant S.O.N.I.A.) se déroule également dans une piscine présentant un environnement répétitif. Cela signifie que l'implémentation finale doit être capable de fonctionner même lorsque les *keyframes* sont ambiguës.

7.2.2 Qualité des images

Bien que les motifs observés aient un impact important sur les résultats, la qualité des données joue également un rôle majeur. Cela est suspecté d'être le principal facteur contribuant au bruit

dans le nuage de points, ainsi qu'au fait que le VSLAM ne détecte pas davantage de détails ou, dans certains cas, génère des données erronées. Cette situation est due au fait que le flux visuel provient d'un milieu aquatique, ce qui rend les images légèrement brumeuses et crée un effet de brouillard lorsque l'on observe des éléments situés à plus de quelques mètres. L'éclairage de l'environnement a également un impact, tant sur la visibilité sous l'eau que sur la création de réflexions dès que la caméra est orientée vers la surface. La couleur, à la fois celle de l'environnement et celle captée par la caméra, constitue un autre facteur qui influence la qualité des résultats.

Étant donné que le flux visuel n'est pas aussi clair que dans l'air, l'algorithme peine à définir des *keyframes* et, même lorsqu'il y parvient, il a du mal à les re-identifier, car leur qualité varie selon la distance et l'éclairage. De plus, il est connu qu'il peut confondre des *keyframes* avec leurs reflets; un exemple notable est celui de l'obstacle en forme de porte, lors duquel le prototype oscillait en roulis. L'hypothèse avancée est que le VSLAM alternait continuellement entre l'obstacle réel et son reflet comme référence visuelle, mais les données enregistrées ne permettent pas de confirmer ou infirmer cette hypothèse.

Cela démontre que l'environnement, autant à l'intérieur qu'à l'extérieur, exerce une influence significative sur la capacité de l'algorithme à comprendre son environnement. Il est encore une fois essentiel de considérer cette problématique dans le contexte de S.O.N.I.A. et de leur compétition, qui se déroule à la fois dans une piscine standard et en extérieur. Cela signifie que la qualité de l'eau, le moment de la journée et la couverture nuageuse influencent tous les trois le processus.

7.3 Impact de l'odométrie visuelle sur le contrôle du mouvement avec EKF

Bien que la capacité à cartographier et mémoriser l'environnement soit importante, son impact sur le contrôle est tout aussi crucial, voire plus. L'objectif de ce projet est de réduire l'accumulation de dérive observée dans le contrôle existant. Les résultats montrent toutefois

que cela n'est pas aussi simple que ce qui avait été initialement hypothèse. Plusieurs facteurs liés à l'EKF dans le contrôle influencent les résultats, en particulier dans le contexte aquatique.

Parmi les 13 états définis, les entrées originales n'affectent que 11 d'entre eux. Les valeurs de position en x et y sont inférées. De plus, les valeurs de vitesse linéaire provenant du DVL ne sont pas fournies de manière constante en raison des limitations du capteur. Cela indique que, pour l'EKF avant l'intégration du VSLAM, l'orientation et les vitesses angulaires reposent sur l'IMU, les vitesses linéaires proviennent du DVL lorsque les données sont disponibles, et la position en z est basée sur le capteur de profondeur. Deux facteurs contribuent à l'accumulation de dérive : l'intermittence du DVL et, puisque l'IMU dépend d'un magnétomètre, un léger décalage de lacet (*yaw*) apparaissent avec le temps (variant selon les conditions environnementales).

L'introduction des 13 états complets issus de l'odométrie visuelle a un impact majeur sur les valeurs de position en x et y, puisqu'elles deviennent désormais la référence (*ground truth*) dans l'EKF pour ces valeurs. Pour toutes les entrées de capteurs, l'impact des covariances définies est important, car il détermine dans quelle mesure l'EKF fait confiance à ces valeurs de référence. Voici les covariances utilisées dans tous les tests, à l'exception du dernier test en eau.

$$\sum Initial = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

Équation 2

$$\sum IMU = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Équation 3

$$\sum DVL = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}$$

Équation 4

$$\sum DEPTH = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.1 \end{bmatrix}$$

Équation 5

$$\sum SLAM_1 = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

Équation 6

Avec le dernier test utilisant la matrice suivante :

$$\sum SLAM_2 = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix}$$

Équation 7

Lors de plusieurs tests en eau (tous utilisant la matrice Σ_{SLAM_1}), il a été observé que le prototype perd rapidement le contrôle en raison d'informations contradictoires provenant à la fois des capteurs internes et de l'odométrie visuelle (VO). Même avec ces covariances, l'influence de la VO avait toujours un impact significatif sur les états résultants de l'AUV. La variabilité de l'impact de la VO découle de l'instabilité observée de l'algorithme VSLAM dans l'eau.

L'objectif final était, au minimum, d'ajuster l'orientation retournée par VSLAM pour qu'elle présente des covariances similaires à celles de l'orientation de l'IMU. Malheureusement, il reste nécessaire d'améliorer la stabilité de l'algorithme. La dernière série de tests effectués en eau a utilisé la matrice de covariance Σ_{SLAM_2}. L'objectif était de tenter d'atténuer l'effet de la VO sur l'EKF du contrôle. Les résultats n'ont été mitigés, avec aucune différence notable observée dans les résultats.

En fin de compte, la première étape consiste à créer un système capable de générer une VO fiable, plutôt que de se contenter d'utiliser des covariances pour compenser les instabilités ou les imprécisions.

CHAPITRE 8
RECHERCHES FUTURES

L'analyse a montré que les résultats n'ont pas permis d'atteindre les attentes initiales. En se basant sur l'ensemble des informations recueillies au cours de la recherche et des tests, voici les étapes qui seraient à suivre pour implémenter un algorithme VSLAM capable de fonctionner aux niveaux requis, en particulier sous l'eau.

8.1 Implémentation alternative du VSLAM

La raison d'avoir mis en œuvre l'infrastructure VSLAM existante via la bibliothèque NVIDIA Isaac ROS Visual SLAM reste pertinente compte tenu des contraintes et des informations disponibles au début de ce projet, mais il serait crucial d'explorer d'autres types d'implémentations. La principale limitation de la bibliothèque Isaac réside dans le manque de paramètres de configuration facilement accessibles et dans l'absence d'informations claires sur le pipeline de traitement. Dans ce contexte, adopter une approche plus « minimaliste » pourrait permettre de développer une solution offrant davantage de possibilités de configuration et d'ajustement des paramètres, idéale pour une utilisation sous l'eau.

Le choix de continuer à utiliser un algorithme basé sur un graphe reste considéré comme la meilleure option, car il demeure globalement le plus stable. Une solution envisageable serait ORB-SLAM3 pour deux raisons principales : il s'agit d'une solution libre bien connue et elle constitue la référence utilisée par la bibliothèque Isaac pour comparer sa propre solution. Cette approche permettrait de conserver une logique d'utilisation similaire à l'implémentation existante. Le principal défi de cette approche réside à la fois dans les difficultés d'optimisation (aucune de ces solutions n'utilisant nativement le GPU) et dans l'effort nécessaire pour l'intégrer au réseau ROS2 existant.

8.1.1 Algorithmes SLAM incluant l'acoustique

Dans un monde idéal, le meilleur choix serait de trouver un algorithme dédié à une utilisation sous-marine, incluant notamment ceux qui exploitent une entrée acoustique comme un sonar. Quelques exemples de ce type d'algorithmes sont le Visual-Inertial-Acoustic SLAM avec DVL (Huang et al., 2025), SVin2 (Rahman, Li et Rekleitis, 2019), et le SLAM utilisant un FLS (Li et al., 2018).

Ces solutions fonctionnent en tirant parti des informations acoustiques. VIA-SLAM et SVin2 sont tous deux conçus pour l'exploration sous-marine, VIA-SLAM étant particulièrement intéressant pour S.O.N.I.A., puisque le prototype principal est déjà équipé d'un DVL. L'approche FLS est également intéressante, notamment dans les situations de faible visibilité, car cet algorithme utilise un sonar comme principale source d'information. Il est conçu pour fonctionner même dans des environnements ambigus, tels que le site de la précédente compétition RoboSub, TRANSDEC à San Diego, Californie, États-Unis, où la visibilité était extrêmement réduite.

Dans tous ces cas, la présence d'un capteur acoustique tel qu'un sonar est nécessaire pour que l'algorithme fonctionne. L'équipe S.O.N.I.A. prévoit d'intégrer un sonar sur le prototype AUV8.1, ce qui lui permettrait d'utiliser ces solutions. Malheureusement, aucun projet similaire n'est prévu pour le prototype LITE1, ce qui nécessite donc d'explorer d'autres solutions.

8.1.2 Direct Sparse Visual Odometry

Lors des discussions sur l'avenir des prototypes avec l'équipe S.O.N.I.A., ils ont exprimé un intérêt pour le remplacement des caméras actuelles par des caméras monoculaires plutôt que stéréo. La raison en est que l'IA utilisée par l'équipe rencontre des difficultés pour identifier les objets en raison du mélange des couleurs. L'idée est de passer à une caméra offrant de

meilleures capacités de détection des couleurs. Ce changement pourrait ouvrir la voie à une autre approche du SLAM : l'utilisation de *Direct Sparse Visual Odometry*, ou DSO (Engel, Koltun et Cremers, 2018).

Seule, la DSO serait moins performante en environnement sous-marin, car elle ne détecte pas les caractéristiques comme les algorithmes mentionnés précédemment. Son principe repose sur l'analyse des variations de densité des pixels pour suivre l'environnement (Engel, Koltun et Cremers, 2018). Cela permet à l'algorithme de suivre les changements même lorsque le flux visuel ne présente pas de caractéristiques significatives. Malheureusement, cela signifie aussi que lorsque l'environnement produit un flux visuel où la couleur et l'intensité des pixels restent similaires en permanence (comme sous l'eau) cette approche ne fournit pas de résultats concluants. L'intérêt réside donc dans son utilisation non pas isolée, mais intégrée à une chaîne de traitement.

Il est fréquent d'utiliser plusieurs méthodes pour concevoir un algorithme global plus efficace, comme le VI-DSO (Stumberg, Usenko et Cremers, 2018), ou encore la solution complexe décrite par Fu et Lu (2025). Pour les prototypes de S.O.N.I.A., ce type de solution serait difficile, mais possible à intégrer dans un EKF existant ou un graphe de facteurs, tout en ouvrant la possibilité d'une navigation plus robuste et précise, surtout en cas d'évolution de la technologie des caméras.

8.2 Prétraitement des données

L'une des premières étapes de tout algorithme VSLAM basé sur un graphe est l'extraction de caractéristiques à partir d'une image. Dans de nombreux cas, on utilise un détecteur tel qu'ORB, SIFT ou SURF (Rublee et al., 2011). C'est à partir des résultats de ces caractéristiques détectées que l'algorithme global peut assembler la carte 3D utilisée pour la cartographie, puis pour la localisation. Cela signifie que si les données issues des images sont incohérentes ou de mauvaise qualité, tout algorithme VSLAM pur verra sa capacité de

performance réduite. Par conséquent, la recommandation proposée est d'introduire une étape de prétraitement avant que les images ne soient évaluées par les détecteurs de caractéristiques.

8.2.1 Égalisation d'histogramme

L'égalisation d'histogramme est une technique bien connue d'équilibrage des couleurs qui consiste à créer un histogramme du spectre de couleurs de l'ensemble de l'image, puis à rééquilibrer les couleurs pour qu'elles soient plus uniformes sur toute l'image (Patel, Maravi et Sharma, 2013). Cette méthode permet d'éclaircir les zones sous-exposées et d'assombrir les zones surexposées.

Avec cette approche, les images conserveraient un spectre de couleurs constant et seraient beaucoup plus résistantes aux variations d'éclairage dans l'environnement. Par exemple, lors de tests en intérieur où l'éclairage est fixe dans certaines zones, ce qui crée des régions plus sombres, ou lors de tests en extérieur avec des nuages provocants des variations de luminosité.

Alors que l'égalisation d'histogramme agit globalement sur l'image, la méthode CLAHE, *Contrast Limited Adaptive Histogram Equalization*, (Nguyen et al., 2020) est une approche beaucoup plus sûre et stable, notamment dans le cadre du VSLAM. En effet, elle minimise le risque de perte de détails liée aux corrections globales.

8.2.2 Algorithme Retinex

Il s'agit d'une approche plus complexe dans laquelle des algorithmes comme Retinex (Nguyen et al., 2020) ont pour objectif d'analyser les images du point de vue humain. Cela est réalisé en séparant l'image en ses composantes d'illumination et de réflectance. À partir de ces informations, l'algorithme tente de restaurer les couleurs naturelles de l'image, notamment dans des environnements où l'éclairage est complexe, grâce à sa capacité à équilibrer la lumière.

Contrairement à certains autres algorithmes complexes, des recherches ont démontré que Retinex est particulièrement efficace pour l'analyse sous-marine (Aguirre-Castro et al., 2022). Cela s'explique par sa capacité de correction adaptative des couleurs, permettant de résoudre des problèmes tels que le faible contraste ou la distorsion des couleurs.

Pour ce projet, l'utilisation de Retinex aurait un impact important sur les résultats, étant donné que la qualité des couleurs de l'environnement ne sera jamais stable. De plus, puisque cet algorithme est libre, il existe déjà des implémentations que l'équipe pourrait utiliser pour accélérer l'intégration.

8.2.3 Exemple de pipeline

Voici une présentation d'un exemple de chaîne de traitement (*pipeline*) qui pourrait être utilisée et qui inclut les algorithmes présentés.

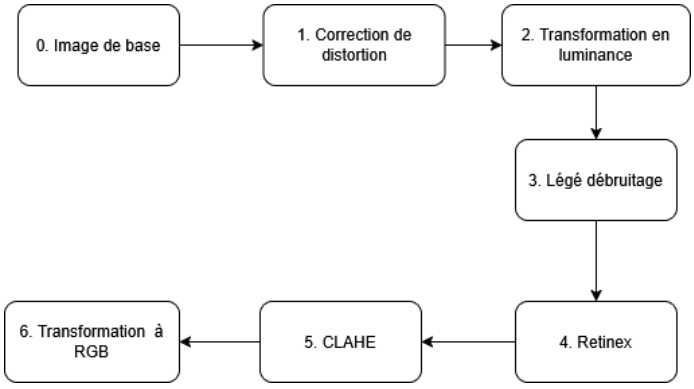

Figure 16 Preprocessing Pipeline

En parcourant la Figure, voici les explications pour chaque étape :

1. Cette étape est gérée directement par la calibration de la caméra choisie. Son rôle est de corriger toute distorsion générée par la présence dans l'eau et du hublot en acrylique.
2. Ensuite, il est recommandé de convertir l'image dans un espace de travail plus robuste, tel que le luminance (RGB → YCrCb). En effet, la plupart des techniques qui affectent

le contraste fonctionnent mieux sur la luminance que sur chaque canal RGB, ce qui permet d'éviter des décalages de couleur déstabilisants.

3. L'application d'un filtre de débruitage intermédiaire tout en préservant les contours est très utile pour minimiser l'impact des fausses caractéristiques qui peuvent être générées en raison de la nature variable de l'eau. Dans l'amélioration sous-marine, l'objectif est d'inclure du débruitage et un lissage afin de maintenir la réflectance stable (Fu et al., 2014).
4. Cette étape correspond à l'application de l'algorithme Retinex présenté précédemment. L'objectif est de corriger l'illumination et de récupérer toute couleur ou visibilité perdue. Il est extrêmement important que cette correction soit appliquée sur la luminance et non sur l'image RGB standard.
5. L'algorithme CLAHE est ensuite appliqué sur la luminance afin que les variations locales de contraste soient corrigées et permettent de révéler les textures pouvant aider à la détection des caractéristiques.
6. Enfin, comme l'implémentation actuelle du VSLAM attend une image au format RGB, la dernière étape consiste à reconvertir l'image dans ce format.

Dans cet exemple, la chaîne de traitement permet de révéler des informations potentiellement perdues, mais surtout de créer un environnement minimisant l'impact des variations de luminosité et de couleur dans l'environnement.

8.3 Variations dans la fusion des capteurs

La manière dont les données des capteurs sont fusionnées pour générer les estimations d'état constitue un facteur majeur dans la qualité des déplacements des prototypes, ce qui influence

à son tour la capacité à détecter et corriger la dérive. Il est donc important d'examiner précisément comment cette fusion est appliquée et quel en est l'impact.

8.3.1 Approche basée uniquement sur l'EKF

D'après les informations disponibles, cette approche correspond à l'implémentation actuelle de la solution. Dans cette approche, l'objectif est de laisser l'EKF réaliser la majeure partie du traitement, tandis que l'entrée du VSLAM est considérée simplement comme un capteur supplémentaire. Il est précisé que l'intégration actuelle applique la fermeture de boucle (*loop closure*) de manière asynchrone, mais les données montrent que son impact sur les résultats globaux est négligeable. C'est pourquoi il est indiqué que l'implémentation actuelle repose uniquement sur l'EKF.

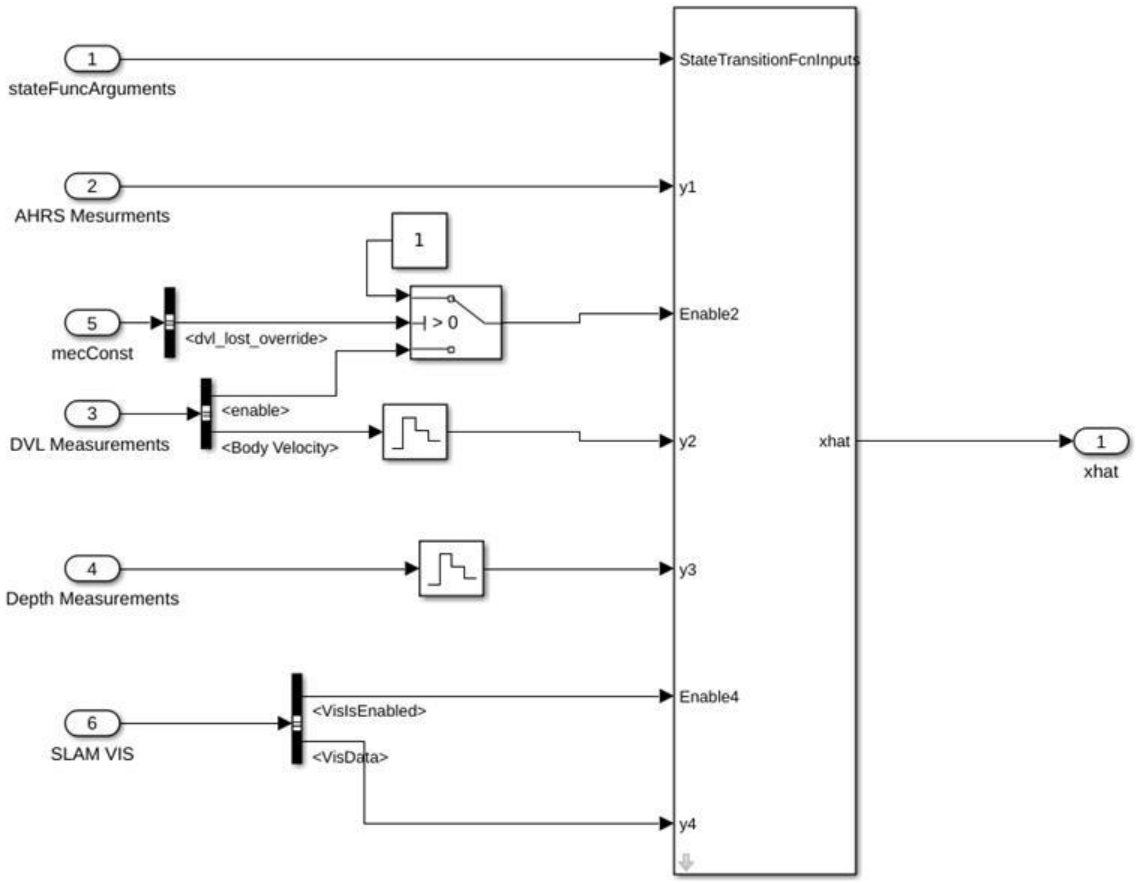

Figure 17 Implémentation EKF Seulement

Dans le modèle représenté à la Figure, l'EKF reçoit directement les données de chacun des capteurs et les associe aux 13 états correspondants. De plus, des covariances statiques sont attribuées à chaque entrée (telles que définies à la section 7.3).

L'entrée *VisData* correspond à l'odométrie visuelle (VO) contenant les 13 états et est calculée en combinant les données visuelles et inertielles de la caméra. En définitive, il s'agit d'une approche très traditionnelle de la fusion de capteurs, mais également très robuste et fiable.

Les problèmes liés à cette implémentation apparaissent lorsqu'on considère la manière exacte dont les données sont fusionnées. En pratique, l'EKF traite chaque flux de données de manière indépendante et les fusionne pour produire les états finaux. Cela signifie que les données ne

sont pas utilisées pour améliorer la qualité des résultats du VSLAM, entraînant la perte des optimisations intrinsèques au VSLAM, telles que l'effet global de la fermeture de boucle ou la cohérence de la carte à long terme. Cela découle du fait que l'EKF ne tient pas compte des données passées pour effectuer des corrections (fermeture de boucle), et que la correction de dérive ne s'applique qu'aux changements observés localement, et non aux corrections globales.

Cette situation conduit directement à la nécessité d'utiliser des covariances plus élevées pour le VSLAM, en raison des incertitudes accrues, particulièrement lorsque les données visuelles sont de qualité limitée.

8.3.2 Approche basée uniquement sur le VSLAM

À l'inverse de l'approche basée uniquement sur l'EKF, l'approche VSLAM uniquement implique que l'ensemble des données capteurs soit fusionné directement au sein de l'algorithme VSLAM, et que l'odométrie visuelle résultante constitue l'estimation d'état finale. Cette approche permet d'obtenir une sortie VSLAM potentiellement beaucoup plus stable, mais entraîne une perte d'information provenant de chaque capteur pris individuellement. De plus, tous les algorithmes VSLAM ne sont pas conçus pour prendre en compte l'ensemble des types de capteurs.

Il existe également une certaine perte d'information, puisque ce type de fusion ne traite pas explicitement les covariances, mais les considère de manière implicite à travers les étapes d'optimisation de l'algorithme. La manière dont les optimisations et la fermeture de boucle influencent le résultat final peut aussi engendrer des incohérences dans le contrôle du mouvement, selon la qualité et la disponibilité des données.

Bien qu'il serait intéressant d'observer précisément le comportement de cette approche en milieu aquatique, il est également nécessaire de considérer qu'un EKF devrait idéalement être

placé juste en amont du MPC, afin d'éviter des fluctuations de données excessives qui pourraient entraîner une défaillance du MPC.

8.3.3 Approche hybride (*late fusion*)

L'approche hybride constitue la solution théorique idéale (Dellaert et Kaess, 2017). La manière exacte de l'implémenter dépend toutefois du type d'algorithme VSLAM utilisé. L'objectif est de laisser le VSLAM effectuer l'ensemble des optimisations et des fermetures de boucle, puis d'injecter ces résultats dans l'EKF.

La première étape consiste à déterminer quelles informations sont transmises au VSLAM et lesquelles sont envoyées directement à l'EKF, indépendamment de la VO. La logique sous-jacente est d'identifier quels flux de données ont l'impact le plus bénéfique sur chaque composant.

En tenant compte des capteurs existants et en supposant que l'algorithme VSLAM utilisé est compatible avec la fusion proposée, voici une architecture suggérée : L'algorithme VSLAM recevrait les données de la caméra stéréo et de son IMU interne, ainsi que celles des capteurs contribuant à la cohérence globale, tels que le DVL et le capteur de profondeur, qui agiraient comme contraintes supplémentaires (Huang et al., 2025). De son côté, l'EKF intégrerait l'IMU centrale, le DVL, le capteur de profondeur et la pose issue du VSLAM. La justification de cette répartition repose sur le fait que le MPC nécessite des états du système à une fréquence constante.

Un facteur important à considérer avec cette approche est que la logique de fermeture de boucle n'influence pas directement la boucle de contrôle de l'EKF, tout en restant prise en compte indirectement. Les corrections sont appliquées dans le repère global afin que la cartographie et la localisation globale du VSLAM continuent de fonctionner correctement, sans introduire de sauts ou de changements significatifs au niveau local. Cela permettrait d'établir des références

globale et locale stables, assurant un fonctionnement fluide du système de contrôle et limitant l'accumulation de dérive à long terme.

Un autre point à considérer est d'éviter l'évaluation redondante des mêmes données. En pratique, si le VSLAM intègre déjà les informations du DVL et du capteur de profondeur, les liaisons directes de ces capteurs vers l'EKF devraient être désactivées ou limitées à un rôle de simple indication, plutôt qu'à une influence directe.

8.4 Validation de la stabilité visuelle

La dernière technique recommandée consiste à mettre en œuvre une méthode permettant d'évaluer la qualité du flux visuel ou de la sortie du VSLAM. La manière exacte d'y parvenir dépendra de l'implémentation spécifique des capteurs et de l'algorithme VSLAM utilisé, mais l'impact attendu demeurera le même.

L'objectif serait de relier le résultat de cet algorithme d'évaluation à des covariances variables du bloc EKF implémenté. Le caractère temporellement variable de ces covariances est essentiel, car il implique que le niveau de confiance accordé par le système à un capteur donné peut évoluer au cours du temps.

Le VSLAM fournit déjà en temps réel les covariances associées à l'odométrie visuelle qu'il génère, lesquelles pourraient être intégrées directement dans l'EKF. Toutefois, il est également pertinent de prendre en compte la qualité du flux visuel afin de déterminer si l'odométrie visuelle doit être totalement ignorée dans certaines situations. Cette approche permettrait un contrôle beaucoup plus fin de la qualité du contrôle du mouvement, en offrant la capacité de filtrer de manière appropriée des résultats imprécis ou erronés.

CONCLUSION

L'objectif de ce projet était d'évaluer la faisabilité de l'intégration d'un algorithme de VSLAM avec les prototypes existants de contrôle, dans le but de réduire l'accumulation de dérive au cours de la navigation. D'un point de vue théorique, cette approche est cohérente et pertinente, puisqu'il a été démontré qu'elle fonctionne dans divers contextes en robotique. Toutefois, un facteur qui n'a pas été pris en compte dans ces implémentations documentées est l'impact du milieu aquatique. Ainsi, des facteurs environnementaux imprévus ont eu un impact significatif tout au long du projet et ont engendré des défis importants.

En particulier, l'environnement aquatique a influencé des paramètres tels que la luminosité, la distorsion des couleurs et la faible visibilité, ce qui a fortement réduit la capacité à détecter des marqueurs géographiques dans l'espace entourant l'AUV. L'ensemble de ces éléments a conduit à une diminution de la fiabilité et de la stabilité de la chaîne de traitement VSLAM. Par conséquent, bien que le système ait démontré certaines fonctionnalités, il était loin d'atteindre le niveau de robustesse requis pour répondre à l'objectif initial en milieu aquatique. Ces limitations n'invalident pas la solution proposée, mais mettent plutôt en évidence la complexité accrue du problème lié à l'utilisation de techniques de vision dans un environnement sous-marin.

Le chapitre 8 présente plusieurs axes de recherche et pistes d'amélioration permettant de répondre à un grand nombre des lacunes identifiées lors des essais. En particulier, l'utilisation d'un algorithme de VSLAM alternatif, l'amélioration du prétraitement visuel ainsi que la mise en œuvre d'une fusion de capteurs hybride pourraient avoir un impact significatif sur la robustesse et la qualité de la solution finale, à condition de disposer de suffisamment de temps de développement. Avec davantage de raffinement, de tests et d'ajustements, les idées présentées dans ce rapport demeurent réalisables.

Par ailleurs, bien que plusieurs cas d'utilisation aient été initialement définis pour l'évaluation, seul le premier a été exploité durant la phase expérimentale. Les deux autres cas d'utilisation

n’ont pas pu être mis en œuvre en raison de problèmes de stabilité et de fiabilité des mouvements, causés par un manque de tests expérimentaux indépendants de ce projet. Cette limitation souligne davantage l’importance de mécanismes de stabilisation secondaires, en particulier dans le cas du prototype LITE1. L’intégration de telles stratégies de stabilisation serait essentielle pour permettre des capacités de déplacement plus avancées.

L’accès à une piscine afin de réaliser des tests fut également très difficile. En plus du prix, il faut également trouver un bassin disponible ainsi qu’une période avec suffisamment de membres de S.O.N.I.A. disponibles afin que les tests se déroulent comme prévu. Tous ces facteurs, sans compter les problèmes techniques imprévus, tels que le bris du *tether* utilisé lors des tests, ont fait en sorte que les premiers essais en piscine n’ont pas eu lieu avant la remise du rapport d’étape. Cela a grandement impacté la planification originelle et a causé du retard, notamment pour l’obtention des résultats et l’analyse de ceux-ci.

En conclusion, ce projet démontre que, bien que l’architecture de navigation proposée soit théoriquement solide et conceptuellement appropriée, son déploiement réussi nécessite un temps de développement nettement plus important, une expérimentation approfondie et des efforts de raffinement supplémentaires. Les enseignements tirés de ce travail constituent néanmoins une base solide pour des travaux futurs et apportent des leçons précieuses en vue de l’objectif à long terme d’une navigation autonome sous-marine fiable au sein de S.O.N.I.A.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Aguirre-Castro, O. A., E. E. García-Guerrero, O. R. López-Bonilla, E. Tlelo-Cuautle, D. López-Mancilla, J. R. Cárdenas-Valdez, J. E. Olguín-Tiznado et E. Inzunza-González. 2022. « Evaluation of underwater image enhancement algorithms based on Retinex and its implementation on embedded systems ». Neurocomputing, vol. 494, p. 148-159. <https://doi.org/10.1016/j.neucom.2022.04.074>.

Aitken, Paul. 2025. « The Future of Drone Mapping with SLAM Technology ». Drone U™. <https://www.thedroneu.com/blog/slam-technology/>. Consulté le 8 novembre 2025.

Anon. [s d]. « cuVSLAM — Isaac ROS ». <https://nvidia-isaac-ros.github.io/concepts/visual_slam/cuvslam/index.html?utm_source=chatgpt.com>. Consulté le 7 décembre 2025a.

Anon. [s d]. « Isaac ROS Visual SLAM — isaac_ros_docs documentation ». <https://soboroo.github.io/NVIDIA-ISAAC-ROS.github.io/repositories_and_packages/isaac_ros_visual_slam/index.html>. Consulté le 8 novembre 2025b.

Anon. [s d]. « Nav2 — Nav2 1.0.0 documentation ». <https://docs.nav2.org/>. Consulté le 8 novembre 2025c.

Anon. [s d]. « robot_localization wiki — robot_localization 2.7.7 documentation ». <https://docs.ros.org/en/noetic/api/robot_localization/html/index.html>. Consulté le 8 novembre 2025d.

Anon. [s d]. « ROS Package — OpenVSLAM documentation ». <https://quantumxt.github.io/openvslam/ros_package.html>. Consulté le 8 novembre 2025e.

Blue Robotics. [s d]. « Bar High-Resolution Depth/Pressure Sensors ». Blue Robotics. <https://bluerobotics.com/store/sensors-cameras/sensors/bar-depth-pressure-sensor/>. Consulté le 8 novembre 2025.

Cadena, Cesar, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid et John J. Leonard. 2016. « Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age ». IEEE Transactions on Robotics, vol. 32, no 6, p. 1309-1332. <https://doi.org/10.1109/TRO.2016.2624754>.

Campos, Carlos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel et Juan D. Tardós. 2021. « ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM ». IEEE Transactions on Robotics, vol. 37, no 6, p. 1874-1890. <https://doi.org/10.1109/TRO.2021.3075644>.

Das, Priyanka. 2020. « A COMPARATIVE STUDY OF KALMAN FILTERS AND PARTICLE FILTERS FOR LOCALIZATION IN DYNAMIC SETTINGS FOR SLAM IN UNKNOWN ENVIRONMENTS ». <https://doi.org/10.5281/ZENODO.14498207>. Consulté le 29 octobre 2025.

Dellaert, Frank et Michael Kaess. 2017. « Factor Graphs for Robot Perception ». Foundations and Trends® in Robotics, vol. 6, no 1-2, p. 1-139. <https://doi.org/10.1561/2300000043>.

DroneXperts. 2025. « Guide complet du SLAM : la technologie du secteur minier ». <https://www.dronexperts.com/article/guide-slam-technologie-secteur-minier/>. Consulté le 8 novembre 2025.

Durrant-Whyte, H. et T. Bailey. 2006. « Simultaneous localization and mapping: part I ». IEEE Robotics & Automation Magazine, vol. 13, no 2, p. 99-110. <https://doi.org/10.1109/MRA.2006.1638022>.

Engel, Jakob, Vladlen Koltun et Daniel Cremers. 2018. « Direct Sparse Odometry ». IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no 3, p. 611-625. <https://doi.org/10.1109/TPAMI.2017.2658577>.

Fu, Chenhui et Jiangang Lu. 2025. « Stereo Direct Sparse Visual–Inertial Odometry with Efficient Second-Order Minimization ». Sensors, vol. 25, no 15, p. 4852. <https://doi.org/10.3390/s25154852>.

Fu, Xueyang, Peixian Zhuang, Yue Huang, Yinghao Liao, Xiao-Ping Zhang et Xinghao Ding. 2014. « A retinex-based enhancing approach for single underwater image ». In 2014 IEEE International Conference on Image Processing (ICIP). (octobre 2014), p. 4572-4576. <https://doi.org/10.1109/ICIP.2014.7025927>. Consulté le 12 décembre 2025.

Grisetti, Giorgio, Rainer Kümmerle, Cyrill Stachniss et Wolfram Burgard. 2010. « A tutorial on graph-based SLAM ». IEEE Transactions on Intelligent Transportation Systems Magazine, vol. 2, p. 31-43. <https://doi.org/10.1109/MITS.2010.939925>.

Hanenko, Liudmyla, Kamila Storchak, Svitlana Shlianchak, Maksym Vorohob et Mylana Pitaichuk. 2025. « SLAM in Navigation Systems of Autonomous Mobile Robots⋆ ». In Cybersecurity Providing in Information and Telecommunication Systems 2025. (Kyiv, Ukraine, février 2025), p. 172-182. CEUR-WS. <https://ceur-ws.org/Vol-3991/paper13.pdf>.

Huang, Haoqian, Zhilin Liu, Li Zhang, Di Wang et Junwei Wang. 2025. « VIA-SLAM: An Underwater Visual–Inertial–Acoustic SLAM With Integrated DVL ». IEEE Transactions on Instrumentation and Measurement, vol. 74, p. 1-11. <https://doi.org/10.1109/TIM.2025.3571157>.

Impact Subsea. [s d]. « ISD4000 - AUV & ROV Depth & Temperature Sensor with AHRS ». <https://www.impactsubsea.co.uk/isd4000/>. Consulté le 8 novembre 2025.

Jung, Haebeom. 2025. zang09/ORB_SLAM3_ROS2. <https://github.com/zang09/ORB_SLAM3_ROS2>. Consulté le 8 novembre 2025.

Korovko, Alexander, Dmitry Slepichev, Alexander Efitorov, Aigul Dzhumamuratova, Viktor Kuznetsov, Hesam Rabeti, Joydeep Biswas et Soha Pouya. 2025. cuVSLAM: CUDA accelerated visual odometry and mapping. <https://doi.org/10.48550/arXiv.2506.04359>. Consulté le 7 décembre 2025.

Li, Jie, Michael Kaess, Ryan M. Eustice et Matthew Johnson-Roberson. 2018. « Pose-Graph SLAM Using Forward-Looking Sonar ». IEEE Robotics and Automation Letters, vol. 3, no 3, p. 2330-2337. <https://doi.org/10.1109/LRA.2018.2809510>.

MathWorks. [s d]. « What Is SLAM (Simultaneous Localization and Mapping)? » <https://www.mathworks.com/discovery/slam.html>. Consulté le 8 novembre 2025.

Merzlyakov, Alexey et Steve Macenski. 2021. « A Comparison of Modern General-Purpose Visual SLAM Approaches ». IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). <https://doi.org/10.48550/arXiv.2107.07589>. Consulté le 8 novembre 2025.

Mourikis, Anastasios I. et Stergios I. Roumeliotis. 2007. « A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation ». In 2007 IEEE International Conference on Robotics and Automation. (Rome, Italy, avril 2007), p. 3565-3572. IEEE. <https://doi.org/10.1109/ROBOT.2007.364024>. Consulté le 7 novembre 2025.

Mur-Artal, Raul et Juan D. Tardos. 2017. « ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras ». IEEE Transactions on Robotics, vol. 33, no 5, p. 1255-1262. <https://doi.org/10.1109/TRO.2017.2705103>.

Nguyen, Thi Phuoc Hanh, Zinan Cai, Khanh Nguyen, Sokunthearidd Keth, Ningyuan Shen et Mira Park. 2020. Pre-processing Image using Brightening, CLAHE and RETINEX. <https://doi.org/10.48550/arXiv.2003.10822>. Consulté le 12 décembre 2025.

NVIDIA. [s d]. « Déployez des machines autonomes évolutives optimisées par l'IA ». In NVIDIA. <https://www.nvidia.com/fr-fr/autonomous-machines/embedded-systems/jetson-agx-xavier/>. Consulté le 8 novembre 2025.

NVIDIA Developer. [s d]. « JetPack SDK ». In NVIDIA Developer. <https://developer.nvidia.com/embedded/jetpack-sdk-515>. Consulté le 8 novembre 2025.

NVIDIA Isaac ROS. [s d]. « Isaac ROS Visual SLAM — Isaac ROS ». <https://nvidia-isaac-ros.github.io/repositories_and_packages/isaac_ros_visual_slam/index.html#overview>. Consulté le 8 novembre 2025.

Patel, Omprakash, Yogendra P. S. Maravi et Sanjeev Sharma. 2013. « A Comparative Study of Histogram Equalization Based Image Enhancement Techniques for Brightness Preservation and Contrast Enhancement ». Signal & Image Processing : An International Journal, vol. 4, no 5, p. 11-25. <https://doi.org/10.5121/sipij.2013.4502>.

Qiao, Junfu, Jinqin Guo et Yongwei Li. 2024. « Simultaneous localization and mapping (SLAM)-based robot localization and navigation algorithm ». Applied Water Science, vol. 14, no 7, p. 151. <https://doi.org/10.1007/s13201-024-02183-6>.

Rahman, Sharmin, Alberto Quattrini Li et Ioannis Rekleitis. 2019. « SVIn2: An Underwater SLAM System using Sonar, Visual, Inertial, and Depth Sensor ». In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (novembre 2019), p. 1861-1868. <https://doi.org/10.1109/IROS40897.2019.8967703>. Consulté le 12 décembre 2025.

Rublee, Ethan, Vincent Rabaud, Kurt Konolige et Gary Bradski. 2011. « ORB: An efficient alternative to SIFT or SURF ». In 2011 International Conference on Computer Vision. (novembre 2011), p. 2564-2571. <https://doi.org/10.1109/ICCV.2011.6126544>. Consulté le 12 décembre 2025.

Stereolabs. [s d]. « ZED 2i | Stereo Camera | Stereolabs ». <https://www.stereolabs.com/en-ca/store/products/zed-2i>. Consulté le 8 novembre 2025a.

Stereolabs. [s d]. « ZED Mini Stereo Camera | Stereolabs ». <https://www.stereolabs.com/en-ca/store/products/zed-mini>. Consulté le 8 novembre 2025b.

Stumberg, Lukas von, Vladyslav Usenko et Daniel Cremers. 2018. « Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization ». In 2018 IEEE International Conference on Robotics and Automation (ICRA). (mai 2018), p. 2510-2517. <https://doi.org/10.1109/ICRA.2018.8462905>. Consulté le 12 décembre 2025.

Teledyne marine. [s d]. « Pathfinder DVL ». <https://www.teledynemarine.com/brands/rdi/pathfinder-dvl>. Consulté le 8 novembre 2025.

VectorNav. [s d]. « VectorNav's VN-100 IMU/AHRS, the world's most trusted surface mount solution ». In VectorNav. <https://www.vectornav.com/products/detail/vn-100>. Consulté le 8 novembre 2025.

Wilbers, Daniel, Christian Merfels et Cyrill Stachniss. 2019. « A Comparison of Particle Filter and Graph-Based Optimization for Localization with Landmarks in Automated Vehicles ». In 2019 Third IEEE International Conference on Robotic Computing (IRC). (Naples, Italy, février 2019), p. 220-225. IEEE. <https://doi.org/10.1109/IRC.2019.00040>. Consulté le 29 octobre 2025.

Yan, Jiang, Liu Guorong, Luo Shenghua et Zhou Lian. 2009. « A Review on Localization and Mapping Algorithm Based on Extended Kalman Filtering ». In 2009 International Forum on Information Technology and Applications. (mai 2009), p. 435-440. <https://doi.org/10.1109/IFITA.2009.284>. Consulté le 29 octobre 2025.

Zhang, Han et Yu Liu. 2024. « Direct Sparse Monocular Visual-Inertial Odometry With Covisibility Constraints ». In 2024 39th Youth Academic Annual Conference of Chinese Association of Automation (YAC). (juin 2024), p. 532-537. <https://doi.org/10.1109/YAC63405.2024.10598807>. Consulté le 12 décembre 2025