Coordination de systèmes sous-marins autonomes basée sur une méthodologie intégrée dans un environnement Open-source

Hoang Anh Pham

▶ **To cite this version:**

Hoang Anh Pham. Coordination de systèmes sous-marins autonomes basée sur une méthodologie intégrée dans un environnement Open-source. Automatique. Université de Toulon, 2021. Français. NNT : 2021TOUL0020 . tel-03788688

HAL Id: tel-03788688
https://theses.hal.science/tel-03788688

Submitted on 27 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° national de thèse : ../../2021 Année 2022

THÈSE

pour obtenir le grade de

Docteur en Automatique, Signal, Productique, Robotique

ÉCOLE DOCTORALE 548 - MER ET SCIENCE

présentée par

Hoang Anh PHAM

préparée au laboratoire COSMER - EA 7398
COnception de **S**ystèmes **M**écaniques et **R**obotiques

Coordination de systèmes sous-marins autonomes basée sur une méthodologie intégrée dans un environnement Open-source

---

Dirigée par :

M. Thierry SORIANO - Professeur, Seatech, Université de Toulon

Soutenue à Toulon, le **23 mars 2021**
devant le jury composé de :

M. Vincent CHAPURLAT	**Professeur – Institut Mines-Telecom**	**Rapporteur**
M. Guillaume ALLIBERT	**MCF-HDR – Université Côte d'Azur**	**Rapporteur**
Mme Claudia FRYDMAN	**Professeure – Université d'Aix-Marseille**	**Examinatrice**
M. Valentin GIES	**Maître de Conférences – Université de Toulon**	**Examinateur**
M. Van Hien NGO	**Professeur associé – Institut polytechnique de Hanoï**	**Examinateur**
M. Vincent RIGAUD	**Directeur du Centre Ifremer Méditerranée**	**Invité**

"Gạo đem vào giã bao đau đớn
Gạo giã xong rồi trắng tựa bông
Sống ở trên đời người cũng vậy
Gian nan rèn luyện mới thành công"

_ Hồ Chí Minh _

Ce poème emprunte l'image de la fabrication du riz (un aliment de base au Vietnam), avec ses différents stades de développement, de la germination à la maturité, en sous-entendant que le succès ne s'obtient qu'avec des efforts et du travail.

Remerciements

Poursuivre des recherches et rédiger une thèse ne s'avèrent pas une tâche facile. Encore moins pour une personne dont ce n'est pas la langue maternelle. Je n'aurais pas été là où je suis aujourd'hui sans l'aide de tant de personnes.

J'aimerais tout d'abord remercier mon directeur de thèse, **M. Thierry SORIANO** pour ses conseils, sa disponibilité, sa bienveillance et surtout sa bonne humeur en toutes circonstances. Je suis ravi d'avoir travaillé en sa compagnie car outre son appui scientifique, il a toujours été là pour me soutenir et me conseiller au cours de l'élaboration de cette thèse.

Je remercie également mes deux rapporteurs, **M. Vincent CHAPURLAT**, Professeur à l'Institut Mines-Telecom et **M. Guillaume ALLIBERT**, MCF-HDR de l'Université Côte d'Azur qui ont accepté de relire mon manuscrit et ont pris le temps d'écrire un rapport détaillé et accepté ma soutenance de thèse.

J'adresse aussi mes remerciements à tous les autres membres du jury pour leur présence ainsi que leurs commentaires. Je tiens à remercier **M. Vincent RIGAUD**, Directeur de l'IFREMER et **M. Valentin GIES**, MCF à l'IUT de Toulon, pour leurs commentaires et suggestions toujours avisés à travers les sessions du comité de suivi. Je remercie **Madame Claudia FRYDMAN**, Professeure à l'université de Aix-Marseille pour avoir accepté de participer à mon jury. Je remercie chaleureusement **M. Van Hien NGO**, Professeur à l'Institut Polytechnique de Hanoï, pour son aide et ses conseils judicieux tout au long de ma recherche.

Un grand merci à **M. Vincent HUGEL**, Directeur du laboratoire COSMER, qui m'a donné l'opportunité de travailler au laboratoire COSMER. Il a toujours fait preuve d'un enthousiasme communicatif et il a su, par ses remarques et ses conseils sur la programmation ROS pour le robot sous-marin BlueROV, fait avancer ma réflexion.

Je tiens à remercier particulièrement **Madame Samia BOUCHAFA-BRUNEAU** et **Madame Yasmina BESTAOUI-SEBBANE** de l'Université de Paris-Saclay, mes tutrices de stage de Master au laboratoire IBISC, qui m'ont toujours fait confiance et qui m'ont soutenu pour poursuivre en thèse à l'issue de mes études.

Merci à **Madame Claire DUNE** pour ses échanges sur la partie du traitement d'image. Je remercie également **Madame Sabine SEILLIER** pour son aide dans la préparation des matériels pour mes expérimentations. Merci à tous les membres du laboratoire COSMER, pour leur aide et leur encouragement tout au long de ma thèse.

Ce travail n'aurait pas pu être mené à bien sans l'aide de l'École doctorale **ED548 Mer et Sciences**, qui, au travers de son soutien matériel, a reconnu mon travail et m'a fait confiance.

Il m'est impossible d'oublier mes amis et mes collègues : **Jean-Baptiste, Matheus, Nicolas, Ornella, Manon**, avec qui j'ai partagé ces années de thèse. Leur soutien indéfectible, leur bonne humeur et leur joie de vivre m'ont permis d'effectuer mes recherches dans les meilleures conditions possibles. Merci de m'avoir fait découvrir la culture française

et surtout de m'avoir aidé à améliorer mon français. Un grand merci à Jean-Baptiste, *mon professeur de français*, qui a toujours pris le temps de bien m'expliquer les mots dans leurs détails, et qui m'a encouragé tout au long de ce travail.

Enfin, je remercie chaleureusement ma famille, mes proches, et surtout ma chère épouse **Mai Anh** pour son soutien quotidien indéfectible, sa positivité et son enthousiasme contagieux à l'égard de mes travaux de recherche comme de la vie en général. Merci de m'avoir toujours fait confiance dans toutes mes décisions.

Il me sera très difficile de remercier tout le monde car c'est grâce à l'aide de nombreuses personnes que j'ai pu mener cette thèse à son terme. Merci infiniment à ceux qui ne sont pas cités ici.

Table des matières

1 Introduction générale

1.1 Introduction

Les océans représentent les trois quarts de la surface de la Terre. Selon les experts océanographiques, environ 90 % des fonds marins sont encore un mystère pour l'Homme [1]. Comment pourrait-on les explorer ? Comment s'y prendre de manière efficace et optimale en termes de temps et de coût ? Pour y répondre, l'utilisation de la robotique est une option intéressante. En effet, l'assistance de robots fait partie des possibilités efficaces pour l'exploration de lieux difficilement atteignables pour l'Homme, comme la Fosse des Mariannes. Leurs collaborations peuvent renforcer l'efficacité des actions d'investigation.

La coordination multi-robots constitue un sujet récurrent dans les recherches scientifiques et les applications civiles. Les avantages de celle-ci par rapport à l'opération individuelle sont multiples : robustesse, adaptabilité, flexibilité et évolutivité. Nous avons remarqué de nombreuses applications, notamment dans les satellites, drones avions (ex. Intel Drone Light Shows [1]), robots mobiles (ex. Amazon robots warehouse [2]), bateaux (ex. Aquabotix's SwarmDivers [3]). Notre recherche porte principalement sur la coordination des robots sous-marins, avec de nombreuses applications potentielles telles que l'exploration des fonds marins ou la surveillance des oléoducs d'installation sous-marins.

La recherche de la coordination des robots en général et des robots sous-marins en particulier est une étude qui comprend de nombreux aspects parallèles tels que la recherche sur les systèmes mécaniques et dynamiques ; systèmes de contrôle et algorithmes de contrôle ; systèmes de détection et méthodes de localisation ; système de circuits électroniques et système d'organisation de programmes logiciels. Dans la littérature, il a beaucoup de travaux de recherche pour chaque aspect. Cependant, il existe également un problème lié au besoin d'une méthode de recherche capable de relier les aspects étudiés ci-dessus pour nous aider à hériter, réutiliser et étendre cette étude.

1. Voir les détails sur https://www.ouest-france.fr/leditiondusoir/data/941/reader/reader.html#!preferred/1/package/941/pub/942/page/4

Toutefois, la coordination multi-robots dans l'environnement sous-marin présente encore des défauts et des enjeux importants. D'une part, la transmission faible des ondes électromagnétiques dans cet environnement entraîne des difficultés de communication considérables, au niveau du débit de communication, de la distance, de la bande passante entre robots et avec le poste de contrôle. D'autre part, les technologies de positionnement de robot sous-marin sont encore limitées et le coût de l'équipement est élevé. Si nous pouvons appliquer la méthode de positionnement par satellites GPS (en anglais Global Positioning System ou GPS) aux robots qui opèrent à la surface (y compris les navires, les robots terrestres, les avions), ce n'est pas le cas pour les robots sous-marins par défaut du signal. Le GPS est basé sur des ondes électromagnétiques et la propagation est limitée sous l'eau, même si certaines approches apparaissent [2]. Par conséquent, les recherches sont principalement effectuées sur des logiciels de simulation avec des hypothèses sur l'auto-positionnement et la communication entre les robots. Surtout aujourd'hui, il y a encore un manque de recherche sur le contrôle coordonné qui réalise de multiples scénarios (évitant les collisions entre robots, évitant les collisions entre robots et obstacles) qui sont effectivement testés sur des robots open source à faible coût.

1.2 Objectifs de thèse

L'objectif de cette thèse est d'étudier la coordination de robots sous-marins dans le contexte de l'application civile et au service de la recherche. L'idée de cette étude est d'avoir un groupe de robots capables de travailler ensemble sur une tâche. Un exemple serait une formation d'équipe, où l'espacement entre les robots est maintenu. Cette coordination peut nous aider à explorer une zone plus vaste dans le même laps de temps.

Nous avons défini deux critères principaux dans le cadre de notre recherche : le premier consiste à utiliser des programmes et des algorithmes qui peuvent être réutilisables dans d'autres applications et peuvent être développées dans l'environnement open-source. Le second concerne l'utilisation des dispositifs à bas-coût (tels que plateforme de robots, des capteurs, des caméras).

Notre étude dans un premier temps sera d'analyser un système multi-robots sous-marins. Dans cette optique, la construction du cadre méthodologique permet de couvrir toutes les étapes de l'analyse à la réalisation, de sorte que les algorithmes puissent être testés rapidement et facilement. De plus, en utilisant le processus d'analyse de la conception de ce système, les utilisateurs et les chercheurs peuvent facilement apprendre, étudier et superviser l'ensemble du système, ce qui est particulièrement propice à l'étude de nouvelles caractéristiques d'héritage, d'algorithmes en cas de nécessité. Il est nécessaire de créer un framework capable d'écrire l'ensemble du système de robots sous-marins de la conception

2. Underwater GPS Developer Bundle (https://bluerobotics.com/store/sensors-sonars-cameras/underwater-gps/underwater-gps-developer-bundle/)

à la mise en œuvre.

Dans un second temps, nous approfondirons les problématiques sur les algorithmes de contrôle distribué pour les combinaisons multi-robots sous-marins pour la mise en formation et le suivi de trajectoire. Dans le détail de ce travail, nous nous concentrons en particulier sur le contrôle de la forme de la formation, qui est définie par les distances entre robots et qui ne dépendent pas du système de communication entre les robots ou du système de positionnement global. Ces algorithmes de contrôle peuvent bien fonctionner avec le système de dynamique non linéaire des robots sous-marins.

Le troisième étude consistera à étudier l'évitement des collisions pendant le processus de formation et le suivi de trajectoire avec la possibilité pour un groupe de robots d'éviter des obstacles sur leur chemin. Cette étude permet également de construire les scénarios de fonctionnement des multi-robots sous-marins au plus près d'un environnement complexe réel.

Avec le désir de construire des scénarios proches des conditions réelles, nous émettons quelques hypothèses suivantes :

— Les robots sous-marins ont une communication très limitée.
— Les robots sous-marins ont de faibles capacités d'auto-positionnement.
— Les robots sous-marins sont prévus comme des systèmes dynamiques non-linéaires avec des paramètres non spécifiés.
— Les erreurs de capteur et les erreurs de communication entraînent une perte de données du capteur peuvent se produire pendant le fonctionnement et le temps de réponse entre les composants est incertain.

1.3 Plan de la thèse

Dans **un premier chapitre**, nous évoquons brièvement l'introduction à la coordination multi-robots.

Le deuxième chapitre présente l'état de l'art incluant la méthodologie d'étude, robot sous-marin, groupe de robot sous-marin, navigation et localisation multi-robots sous-marins, architecture de coordination et la commande distribuée.

Le troisième chapitre étudie en détail la méthodologie intégrée.

Le quatrième chapitre présente le modèle dynamique non-linéaire de sous-marin.

Le cinquième chapitre est consacré au développement de la coordination pour les modèles linéaires de sous-marin.

Le chapitre sixième présente la coordination pour les modèles non-linéaires de sous-marin avec des paramètres incertains.

Le septième chapitre est dédié à la coordination pour les modèles non-linéaires et l'estimation de la position relative basée sur le filtre de Kalman étendu (en anglais Extended Kalman Filter ou EKF).

Enfin, **le dernier chapitre** constitue la conclusion et la proposition de nouvelles pistes de recherche.

1.4 Publications

Publications

1. H.A. Pham., T. Soriano, H.V. Ngo., V. Gies, *Distributed Adaptive Neural Network Control Applied to a Formation Tracking of a Group of Low-Cost Underwater Drones in Hazardous Environments*, Applied Sciences 2020, 10, 1732.

2. Soriano T., Gies V., Pham A.H., Van Hien N. (2020) *Mechatronics Iterative Design for Robots Multi-agent Integration*. In : Mechatronics 4.0, Mechatronics 2019. Lecture Notes in Mechanical Engineering. Springer, Cham.

3. H.A. Pham., T. Soriano, H.V. Ngo., *Coordination of multi-underwater drones : towards an integrated object-oriented methodology in an Opensource environment*, INSIGHT, International Council on Systems Engineering, Wiley, 2019.

4. H.A. Pham., T. Soriano, H.V. Ngo., *Applying AADL to realize embedded control systems for coordination of multiple low-cost underwater drones*, MTS/IEEE Oceans, 2019.

5. H.A. Pham., T. Soriano, H.V. Ngo., *Integrated scenarios of formation tracking and collision avoidance of multi-vehicles*, IEEE - 13th System of Systems Engineering Conference (SoSE), 2018.

6. T. Soriano., H.A. Pham., H.V. Ngo., *Analysis of coordination modes for multi-UUV based on Model Driven Architecture*, IEEE - 12th France - Japan Congress, 10th Europe - Asia Congress on Mecatronics, 2018.

Poster

- Poster au séminaire doctoral, *Coordination de systèmes sous-marins vers une méthodologie intégrée objet dans un environnement Open-source*, GDR Macs/AFIS (Association Français d'Ingénierie Système), Nancy, Decembre, 2018.

2 État de l'art

L'objectif de ce chapitre est de présenter l'état de l'art : la méthodologie d'étude, les robots sous-marins, l'architecture de la coordination multi-robots sous-marins, navigation et localisation de multi-robots sous-marins et des problèmes de coordination de multiple robots sous-marins.

2.1 Méthodologie d'étude

2.1.1 Introduction à l'ingénierie système basée sur les modèles (MBSE)

L'ingénierie système basée sur modèle (en anglais *Model based System Engineering* ou MBSE ou MBE) est une approche bien connue pour le développement de systèmes complexes. Elle possède des caractéristiques permettant de réduire la complexité du développement, d'améliorer la productivité, de gérer efficacement les changements, de réduire les délais de mise sur le marché [4], [5], de détecter et de résoudre les problèmes d'interopérabilité le plus rapidement possible [6]. MBSE s'est imposé comme une méthode efficace pour analyser la conception, la mise en œuvre et l'exécution des systèmes logiciels dans l'industrie [4], [7], [8], [9], des systèmes embarqués [10], [11], [12], des systèmes robotiques [13], [14], [15], [16], [13], des systèmes de contrôle [17], [18], [19], des systèmes multi-robots [20], [21], [22], [23], [24]. Des descriptions détaillées seront fournies dans les sections 2.1.2, 2.1.3. Selon l'article [12], les recherches en approche MBSE sont classées en six catégories correspondantes :

— Catégorie générale : Il existe dans la littérature un certain nombre de recherches pour lesquelles une solution complète couvrant toutes les activités de MBSE (modélisation, transformation, vérification et simulation de modèles) est proposée. Particulièrement, ces recherches ont un grand potentiel pour être utilisées pour les systèmes embarqués temps réel de robots. Certaines recherches peuvent couvrir simultanément plus d'une activité MBSE (par exemple, modélisation, transforma-

tion de modèle ou transformation de modèle, vérification, etc. Toutes les recherches mentionnées ci-dessus seront incluses dans la catégorie générale.

— Catégorie de modélisation (en anglais *Modeling Category*) : La spécification des besoins est l'activité principale. Par conséquent, toutes les catégories proposées peuvent contenir de l'information sur la méthodologie d'élaboration du modèle. UML [1] (en anglais *Unified Modeling Language*) et ses profils SysML/MARTE [2] (en anglais *Systems Modeling Language/ Modeling and Analysis of Real Time and Embedded systems*) sont normalement utilisés aussi bien individuellement que collectivement pour spécifier les exigences des systèmes embarqués. Cependant, il y a un certain nombre de recherches particulièrement destinées à proposer/étudier des méthodologies de modélisation en utilisant UML et ses profils SysML/MARTE. Ces recherches ne fournissent pas de proposition significative concernant d'autres activités relatives au MBE (c'est-à-dire la transformation, la vérification et la validation des modèles).

— Catégorie de transformation du modèle (en anglais *Model Transformation Category*) : La transformation des modèles est une activité importante de MBE. Il existe des travaux de recherche récents sur les techniques de transformation *de modèle à modèle* (en anglais *Model to model* - M2M) ou *de modèle à texte* (en anglais *Model to text* - M2T), ou les deux. Il pourrait également y avoir des travaux de recherche pour s'assurer de l'exactitude de la transformation du modèle. Selon [18] a proposé un changement de paradigme entre un modèle indépendant de la plate-forme (en anglais *Platform Independent Model* - PIM [3]) et un modèle spécifique à la plate-forme (en anglais *Platform Specific Model* - PSM [4]). Cependant, l'optimisation entre les modèles de transformation ainsi que le pourcentage de modèles convertis de PIM en PSM devraient être étudiés plus avant.

— Catégorie de vérification du modèle (en anglais *Model Verification Category*) : Cela contient les recherches concernant les techniques de vérification formelle et informelle du modèle pour s'assurer de l'exactitude du modèle. Normalement, les aspects comportementaux/temporels du modèle ont été vérifiés par des méthodes de vérification formelles.

— Catégorie de simulation (en anglais *Simulation Category*) : La simulation est normalement utilisée pour la validation du système en utilisant le code source généré. Il peut y avoir un certain nombre de recherches qui intègrent les outils de simulation

1. UML est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet
2. Le langage de modélisation UML a été étendu au domaine du temps réel embarqué au travers de la normalisation d'une extension nommée MARTE
3. PIM : modèle d'un système logiciel ou d'un système d'entreprise qui est indépendant de la plateforme technologique spécifique utilisée pour le mettre en œuvre.
4. PSM : un modèle de logiciel ou de système d'entreprise lié à une plate-forme technologique spécifique (par exemple, un langage de programmation, un système d'exploitation, un format de fichier de document ou une base de données spécifiques).

disponibles [25], [26]. Dans ce cas, il peut être suggéré de combiner le modèle de simulation en utilisant ROS/Gazebo [5] (en anglais *Robot Operating System*) et le modèle réel.

— Catégorie de spécification de propriété (*Property Specification Category*) : Il est important de spécifier les aspects comportementaux/temporels des systèmes embarqués au moyen de techniques/langues de spécification des propriétés.

Dans cette étude, nous souhaitons appliquer une approche basée sur MBSE pour pouvoir analyser le système de contrôle distribué pour la coordination de multi-robots sous-marins. Ceci a pour but de nous donner un aperçu de l'architecture de contrôle, ainsi que de faciliter le développement et l'expansion de ces systèmes de contrôle.

2.1.2 MBSE pour le développement de systèmes embarqués en temps réel

Un système embarqué est un hybride de matériel et de logiciel, qui combine la flexibilité du logiciel et la performance matérielle en temps réel. Par ailleurs, un système de contrôle-commande est un système informatique de contrôle de procédé. Le terme procédé est un terme générique désignant un système physique contrôlé. Afin de contrôler le procédé, le système informatique est en relation avec l'environnement physique externe par l'intermédiaire de capteurs et/ou d'actionneurs. Les grandeurs physiques acquises grâce aux capteurs permettent au système de contrôle-commande de s'informer de l'état du procédé ou de son environnement. Le système de contrôle-commande, à partir de consignes décrivant l'état voulu du procédé, calcule des commandes qui sont alors appliquées sur le procédé par l'intermédiaire d'actionneurs. En substance, un système de contrôle peut être considéré comme un système embarqué, où les principaux actionneurs et le traitement sont le matériel, et les algorithmes de contrôle sont le logiciel. Quelques exemples d'application temps réel de contrôle-commande : systèmes de transport, drone volant, sous-marin, robot de production, téléphone mobile, pilotage d'un procédé de fabrication.

2.1.2.1 Caractéristiques temporelles des systèmes embarqués en temps réel

Parce que les systèmes de contrôle pour robots sous-marins nécessitent que les signaux de contrôle soient traités et rendus rapidement (en temps réel), nous nous concentrerons sur les systèmes embarqués en temps réel. Selon [10], les contraintes temporelles peuvent être de plusieurs types :

— **Contraintes temporelles relatives** ou lâches (en anglais *soft real-time*) : les fautes temporelles sont tolérables.

— **Contraintes temporelles strictes** ou dures (en anglais *hard real-time*) : les fautes temporelles ne sont pas tolérables.

5. ROS est un ensemble d'outils informatiques open source permettant de développer des logiciels pour la robotique

— **Contraintes temporelles fermes** (en anglais *firm real-time*) : les fautes temporelles sont autorisées dans une certaine limite, par exemple une erreur toutes les trois exécutions au plus.
— **Systèmes multi-critiques** : les sous-systèmes composant le système sont caractérisés par des degrés de criticité.

Les contraintes temporelles qui sont classiquement présentées sont des contraintes de bout en bout, appelées aussi contraintes de latence. Il est nécessaire de préciser et de formaliser les caractéristiques temporelles d'un système. Nous pouvons définir de manière non exhaustive :

— **Durée d'exécution** d'une activité : l'activité d'une application, qui peut être l'enchaînement de plusieurs activités élémentaires (acquisition, traitement, commande, affichage…), possède une durée d'exécution qui peut être mesurée de diverses manières. Cette durée n'est pas constante à chaque occurrence de cette activité puisque les programmes et les enchaînements de programmes ne sont pas toujours identiques (branchement conditionnel, itération, synchronisation…).
— Cadence de répétition ou **périodicité** d'une activité : l'acquisition d'une donnée ou la commande d'un actionneur peuvent nécessiter une régularité liée par exemple à la fréquence d'échantillonnage.
— Date au plus tôt ou **date de réveil** : dans certains cas, un traitement doit être déclenché à une date précise relative par rapport au début de l'exécution de l'application ou absolue (plus rarement). Cette date de réveil n'implique pas obligatoirement l'exécution ; il peut y avoir un délai dû à l'indisponibilité du processeur.
— **Date de démarrage** : cet instant correspond à l'exécution effective de l'activité.
— **Date de fin** : instant correspondant à la fin de l'exécution de l'activité.
— Date au plus tard ou **échéance** : le traitement ou la commande d'un actionneur doit être terminé à un instant fixé par rapport au début de l'exécution de l'application. Dans le cas d'applications à contraintes temporelles strictes, cette échéance doit être respectée de façon impérative, sinon il y a faute temporelle et l'application est déclarée non valide.
— **Temps de réponse** : cette caractéristique peut s'appliquer à une activité de régulation ou à un ensemble d'activités de régulation ; elle est directement liée à la dynamique du système. Ce paramètre correspond à la différence entre la date de réveil et la date de fin de l'activité.
— **Gigue temporelle** : ce paramètre caractérise la répétabilité d'une activité au fur et mesure de ses occurrences. En effet, entre deux exécutions successives d'une même activité, ses caractéristiques temporelles peuvent changer : date d'activation, durée d'exécution, temps de réponse, etc.

2.1.2.2 MBSE pour le développement de systèmes embarqués

Selon John Hutchinson et al (2014) [27], Macco Brambilla et al (2017) [5], le **MBSE** est considéré comme l'une des approches les plus populaires en abstraction logicielle. Dans le contexte du domaine embarqué, par l'abstraction des détails, **MBSE** aide les ingénieurs logiciels à gérer les complexités du développement de logiciels embarqués [28] en automatisant les artefacts du cycle de vie de développement logiciel (en anglais *Software Development Life Cycle* - SDLC) non seulement dans la mise en œuvre [29] mais également dans les tests et la documentation. De plus, puisque des facteurs économiques tels que le délai de mise sur le marché exigent un processus de développement fiable permettant un développement rapide du SDLC, de nombreuses applications dans différents domaines (par exemple : électronique grand public, défense et aérospatiale, automobile et télécommunications) ont commencé à adopter le **MBSE**. Plus spécifiquement, plusieurs études soulignent la nécessité du **MDE** dans le monde embarqué pour minimiser les effets de l'hétérogénéité et de la complexité des plates-formes [30] ainsi que la validation et la vérification [31]. C'est la raison pour laquelle il a été régulièrement étudié et adapté pour le développement de systèmes embarqués. Les principales activités de MBE pour le développement de systèmes embarqués sont présentées dans la Figure 2.1.

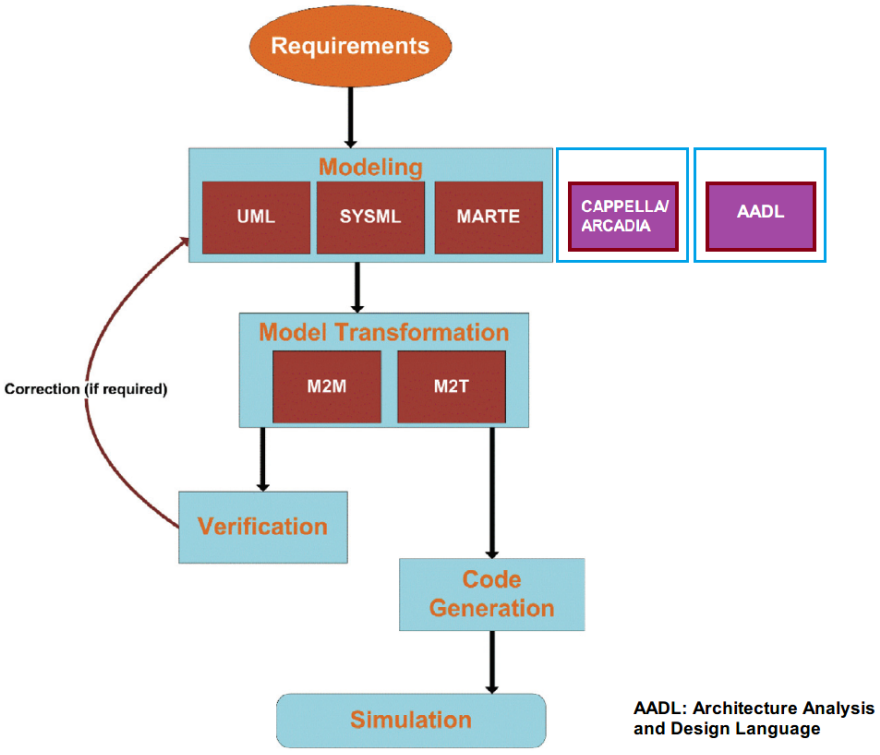


Figure 2.1 – Les principales activités de MBSE pour le développement de systèmes embarqués (selon [12])

Noter que dans la figure 2.1, il y a des ajouts Cappella/Arcadia et AADL (en anglais *Architecture Analysis and Design Language*).

— Cappella/Arcadia qui est une solution Open Source pour l'ingénierie de systèmes basés sur des modèles, est principalement utilisée pour la modélisation de systèmes complexes et critiques pour la sécurité dans le développement de systèmes embarqués pour des industries telles que l'aérospatial, l'avionique, le transport, l'espace, les communications, la sécurité et l'automobile. Il a été créé par Thales en 2007, et est en constante évolution depuis lors. L'objectif d'Arcadia/Capella est d'amener les ingénieurs système à s'adapter au changement culturel de MBE, plutôt que de faire appel à des "experts" en modélisation qui possèdent le modèle au nom des ingénieurs système.

— L'AADL est un langage textuel et graphique à exécution précise pour la modélisation de l'architecture des systèmes logiciels embarqués et de leurs plateformes cibles. L'AADL fournit des concepts de modélisation pour décrire l'architecture d'exécution des systèmes DERT (en anglais *Distributed Embedded Real-Time*) en termes de composants, connecteurs, tâches simultanées, leurs interactions et leur mappage sur une plate-forme d'exécution. Récemment, l'OMG a fourni à la spécification MARTE des lignes directrices pour mapper ses entités de modélisation aux concepts de l'AADL [32], [33], [34].

L'article [11] présente UML/MARTE pour la modélisation et la conception des systèmes embarqués dans l'exploration spatiale, qui est un système de vocodeur EFR (en anglais *Enhanced Full Rate*). Cet article également présente une approche basée sur une combinaison novatrice de *Model Driven Engineering* (MDE), *Electronic System Level* (ESL) et *Design exploration Technologies* (DET). Le cadre proposé permet de construire l'ensemble des solutions de conception possibles, c'est-à-dire l'espace de conception, de manière abstraite, standard et graphique en s'appuyant sur UML et le profil standard MARTE. À partir de ce modèle basé sur UML/MARTE, le framework de génération proposé produit un modèle de performance exécutable, configurable et rapide qui inclut le code fonctionnel des composants de l'application. Une approche similaire peut être appliquée aux systèmes embarqués de robots dans un scénario de coordination multi-robots sous-marins.

2.1.3 MBSE pour les systèmes multi-robots

Dans la thèse [35], qui présente MBSE en conception avionique et simulation aéronautique, et dans l'article [36] rend compte de l'expérience acquise lors de l'introduction de MBSE et UML/SysML sur le système d'hélicoptères autonomes Skeldar, ainsi qu'une description de la situation à l'étape de la planification du projet en ce qui a trait aux conditions commerciales.

L'article [37] étudie le maintien de la cohérence de la conception est une question

cruciale pour le développement de l'aérospatiale au niveau macro en intégrant SysML dans une logique formelle, qui peuvent être utilisées pour maintenir la cohérence à mesure que la conception évolue.

L'ingénierie de systèmes multi-robots est une discipline émergente qui vise à définir des procédures systématiques et bien fondées pour la modélisation, la conception, la réalisation, la vérification, la validation, l'exploitation et la maintenance d'un système multi-robots.

Dans l'article [23], une perspective d'ingénierie est présentée : modélisation et spécification des besoins ; conception et réalisation ; vérification et validation ; exploitation et maintenance. Cependant, cet article présente l'analyse théorique des idées, des points de vue de la perspective d'ingénierie. L'utilisation de langages de modélisation tels que UML ou/et MARTE avec les spécifications du système n'a pas encore été annoncée.

Dans la thèse [13], la méthodologie d'élaboration d'un cadre de développement de modèles pour les systèmes robotiques est présentée. L'auteur a proposé un langage de modélisation *Solution Space Modeling Language* (SSML), pour modéliser formellement la solution et spécifier les attributs de qualité pendant la phase de conception. Cependant, il semble qu'en utilisant le langage SSML, l'auteur ne mentionne pas les contraintes de temps si chaque robot est un système embarqué.

Dans l'article [21], l'auteur adopte le MDE pour la spécification et l'exécution de missions civiles de systèmes mobiles multi-robots. Spécifiquement, dans cet article, il propose une famille de langages de modélisation spécifiques au domaine (*Domain-Specific Modeling Languages - DSML*) pour la spécification des missions civiles des systèmes mobiles multi-robots.

L'article [24] présente des techniques d'ingénierie pilotées par modèle pour le développement de systèmes multi-robots basées sur la méthodologie Prometheus [38] qui est définie comme un processus détaillé approprié pour spécifier, mettre en œuvre et tester/déboguer les systèmes logiciels orientés robots. Il offre un ensemble de lignes directrices détaillées, y compris des exemples et des heuristiques, qui permettent de mieux comprendre ce qui est requis à chaque étape du processus de développement. Cependant, l'auteur ne mentionne pas les contraintes de temps si chaque robot est un système embarqué.

Comme indiqué ci-dessus, dans notre étude, nous considérons le système de contrôle distribué pour la coordination multi-robots sous-marins comme un système embarqué, qui fonctionne en temps réel. C'est pour cette raison, nous voulons évaluer les erreurs potentielles causées par des capteurs (l'un des composants clés de la construction du système de contrôle) et calculer le temps de réponse de l'ensemble du système de contrôle afin de pouvoir compléter l'analyse du système.

2.1.4 Modèles de développement pour l'implémentation MBSE

MBSE a présenté une méthodologie basée sur des modèles pour minimiser la complexité des systèmes. Cependant, pour pouvoir connecter et exploiter ces modèles, nous devons utiliser des modèles de développement logiciel tels que : Modèle V, Modèle de chute d'eau, Modèle Agile et Modèle en spirale.

a) Modèle V (voir la figure 2.2) : Le modèle V est le modèle le plus important utilisé dans le processus de test des logiciels. Il est également connu sous le nom de modèle de vérification et de validation. Il a été introduit par Paul Rook dans les années 1980. L'idée principale du modèle V est que les tâches de développement et les tâches de test sont des activités correspondantes d'importance égale, ce qui est symbolisé par les deux côtés du *V*. Cependant, ce modèle présente certains inconvénients comme suit :

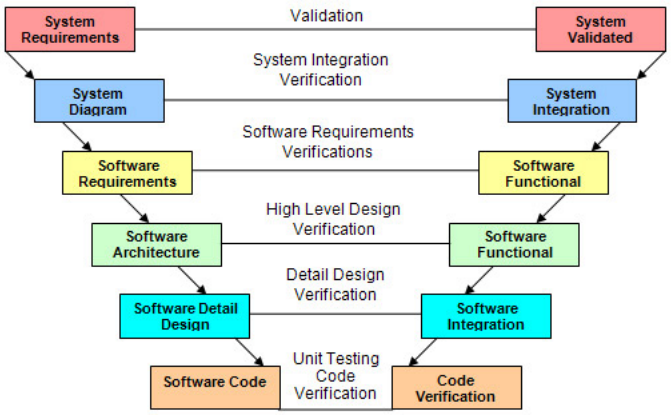


Figure 2.2 – Modèle V (Photo de ProductiveHut.com)

— Le modèle V est très rigide et le moins flexible, ce qui signifie que si l'une des exigences change, le testeur doit mettre à jour l'ensemble de la documentation du test.
— Ce modèle s'applique surtout aux grandes entreprises car il nécessite beaucoup de ressources.
— La quantité et l'intensité des niveaux de test doivent être adaptées aux besoins spécifiques du projet.

b) Modèle de chute d'eau (voir la figure 2.3) : Le concept principal de ce modèle est que ce n'est que lorsqu'un niveau de développement est achevé que le suivant est initié. À la fin de chaque phase, un examen a lieu pour déterminer si le projet est sur la bonne voie et s'il faut ou non poursuivre ou abandonner le projet.

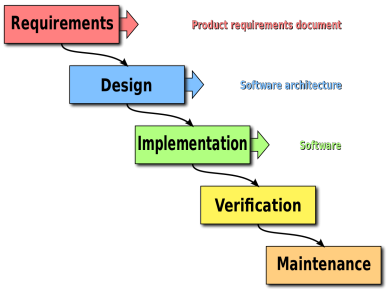


Figure 2.3 – Modèle de chute d'eau (Photo de Wikipedia.com)

L'inconvénient crucial de ce modèle est que le test est compris comme une action *ponctuelle* à la fin du projet juste avant le lancement de l'opération.

— Un niveau élevé de risque et d'incertitude
— Inflexible
— Modèle médiocre pour les projets complexes et orientés objets

c) Modèle Agile (voir la figure 2.4) : Le modèle agile de développement logiciel est un cadre conceptuel pour le génie logiciel qui favorise les itérations de développement tout au long du cycle de vie du projet. L'approche Agile, comme son nom l'indique, implique qu'il faut faire quelque chose très rapidement. Mais c'est pourquoi elle présente certains inconvénients :

— L'accent n'est pas mis sur la conception et la documentation nécessaires.
— Les processus agiles ne sont vraiment applicables qu'aux produits dont la fiabilité n'est pas très critique.

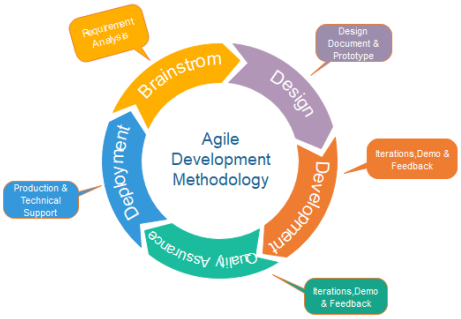


Figure 2.4 – Modèle Agile (Photo de JavaTpoint.com)

d) Modèle en spirale (Voir la figure 2.5) : Ce modèle de développement combine les caractéristiques du modèle de prototypage et du modèle de chute d'eau. Le modèle en spirale a été défini par Barry Boehm [39]. Il est une version évolutive du prototypage incrémental.

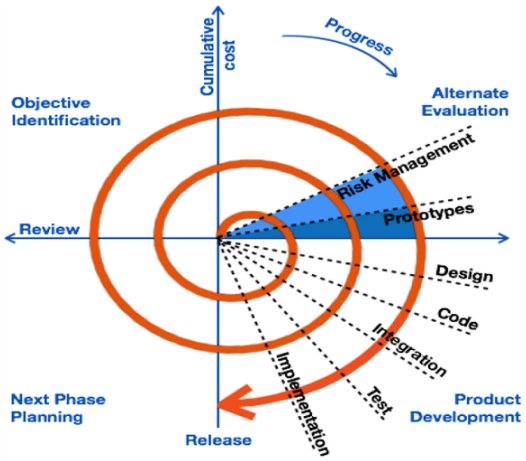


Figure 2.5 – Modèle en spirale (Photo de JavaTpoint.com)

Le modèle spirale du cycle de vie est un modèle très flexible. Les phases de développement peuvent être déterminées par le chef de projet, en fonction de la complexité du projet. Les estimations (c'est-à-dire le budget, le calendrier, etc.) deviennent plus réalistes au fur et à mesure de l'avancement des travaux, car les questions importantes sont découvertes plus tôt.

2.1.5 Conclusion sur la méthodologie d'étude

Cette section présente l'ingénierie système basée sur les modèle (MBSE). De plus, elle introduit également des modèles de développement pour l'implémentation MBSE. Concernant le contenu de la recherche sur les robots sous-marins, la complexité du logiciel de contrôle multi-robots, la diversité des algorithmes de contrôle conduit à des limitations d'héritage et de réutilisation pour s'adapter à différentes applications. Pour cette raison, il est nécessaire d'avoir une méthode d'étude qui peut couvrir :

- Analyse des besoins,
- Modélisation et Spécification des besoins,
- Conception et Réalisation,
- Vérification et Validation,
- Exploitation et Maintenance.

Les détails de cette présentation seront introduits au chapitre 3.

2.2 Robot sous-marin

Un robot sous-marin (en anglais Unmanned Underwater Vehicle ou UUV) parfois connu sous le nom de drone sous-marin, est un robot qui se déplace dans l'eau sans habitant. Il est équipé de moteurs, d'alimentations, de contrôleurs, ainsi que de capteurs IMU pour pouvoir déterminer son état. De plus, il est également équipé de capteurs pour collecter

des données sur les fonds marins, tels que : la température de l'eau, l'oxygène dissous, la turbidité, le pH, les algues bleues, etc.) ou des caméras d'observation. Ces UUV peuvent fonctionner à des profondeurs de plusieurs mètres jusqu'à des milliers de mètres. La figure 2.6 montre l'architecture générale des composants d'un UUV.

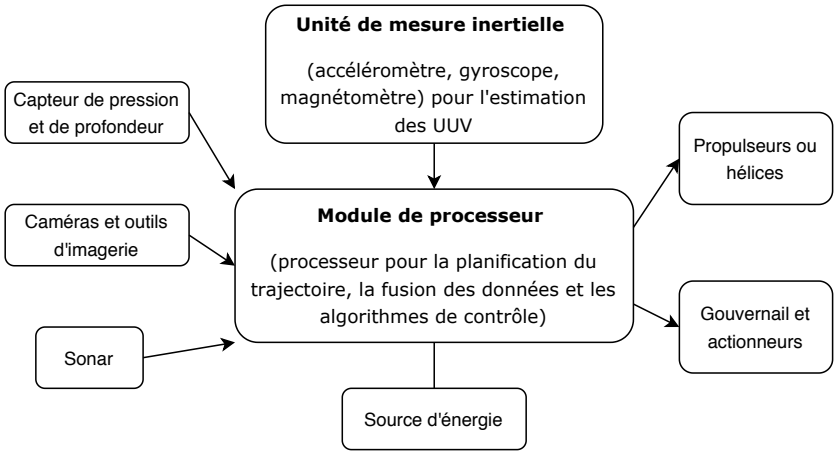


Figure 2.6 – Architecture générale des composants d'un UUV

2.2.1 Classification des robots sous-marins

Actuellement, les UUV sont classés en trois catégories : les robots sous-marins téléguidés (en anglais Remotely Operated underwater Vehicle ou ROV), les robots sous-marins autonomes (en anglais Autonomous Underwater Vehicle ou AUV) et une dernière, la catégorie hybride AUV-ROV. Les sections 2.2.1.1 à 2.2.1.3 ci-dessous présenteront les caractéristiques de chaque catégorie.

2.2.1.1 Robot sous-marin téléguidé

Un ROV est un robot capable de fonctionner sous l'eau grâce au contrôle depuis une station au sol (ou bateau). La communication entre le ROV et la station de contrôle se fait normalement par câble. Dans certains cas, la puissance de travail du ROV peut également être transmise via ce câble. De nos jours, des ROV ont été développés et utilisés dans de nombreuses applications pratiques, par exemple : recherche et sauvetage, armée, loisirs et découverte, aquaculture, biologie marine, pétrole, gaz, énergie offshore, infrastructures immergées, etc. Ils permettent aux opérateurs de capturer des séquences photo et vidéo pour inspecter et surveiller les ports, les havres et les navires, d'apporter des innovations aux inspections de conduite, de localiser des cibles sous-marines et d'explorer la profondeur de nos océans, lacs et rivières. La figure 2.7 présente certains des ROV actuels : ROV - H1000 de ECA groupe [40] (la figure 2.7a) et ROV Guardian de SubSea Tech [41] (la figure 2.7b).

(a) ROV - H1000 (Photo de ECA Groupe)

(b) Mini-ROV Guardian - SubseaTech

Figure 2.7 – Certains des ROV actuels

La figure 2.8 montre le modèle BlueROV de BlueRobotics, un modèle ROV à faible coût, OpenSource basé sur une plate-forme ROS (en anglais *Robot Operating System*). C'est un ROV au service des inspections, de la recherche et de l'aventure.

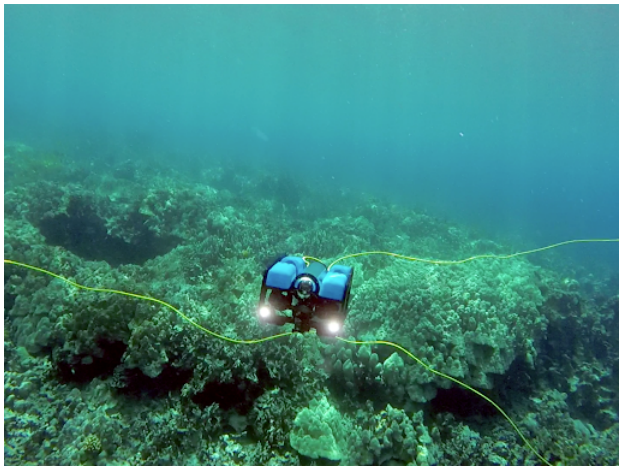


Figure 2.8 – BlueROV de BlueRobotics (Photo de DeltaROV.com)

L'avantage du ROV est dans sa capacité à maintenir une communication constante entre lui et la station de contrôle. La limite de temps de fonctionnement est effacée grâce à l'alimentation fournie par la station de commande via un câble. D'autres atouts indéniables : la rapidité du temps de déploiement ainsi que l'efficacité en termes de coût. La sécurité liée à l'enrouleur de câble pour le rappel en cas de panne.

Cependant, l'utilisation du câble entraîne également trois inconvénients pour le ROV. D'abord, le ROV ne peut fonctionner que dans la longueur du câble. Ensuite, le câble peut affecter le modèle dynamique du ROV, entraînant une instabilité du processus de contrôle. Enfin, l'utilisation du câble peut limiter l'accessibilité et la mobilité du ROV dans sa tâche de surveillance d'un environnement de terrain complexe. En cas d'utilisation en essaim du

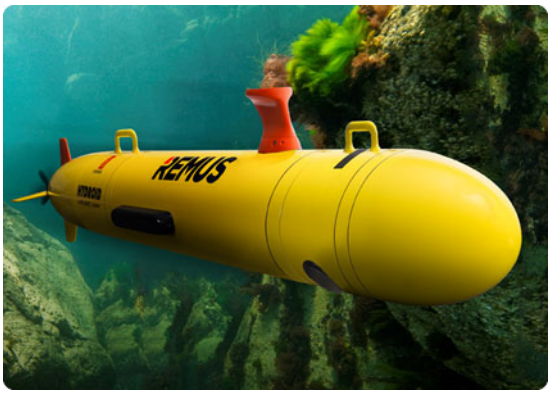
robot, il peut se former des nœuds entre les câbles.

2.2.1.2 Robot sous-marins autonomes

Un AUV est aussi un véhicule sous-marin comme un ROV. Cependant, la différence la plus importante est qu'il est équipé de capteurs ainsi que de contrôleurs qui leur permettent de fonctionner indépendamment - automatiquement sans avoir besoin de contrôle humain. Cela permet à l'AUV se déplacer de manière flexible et de fonctionner dans de vastes zones et également maintenir une trajectoire directe dans l'eau.

Les AUV peuvent voyager sans intervention de l'opérateur une fois qu'ils sont programmés à la surface. Cela permet de collecter beaucoup plus de données pendant une croisière, car le navire peut se concentrer sur d'autres tâches pendant que l'AUV est dans l'eau. La vitesse, la mobilité et la portée spatiale des AUV sont plus importantes que celles des ROV.

Les difficultés de l'AUV sont que son coût de développement est assez élevé, sa capacité de communication ainsi que son temps de fonctionnement sont limités. Les AUV ne peuvent pas fonctionner partout. Ils peuvent être influencés par de forts courants. Ils sont également moins adaptés aux zones fortement peuplées en raison des interférences acoustiques, des risques de collision. La figure 2.9 présente certains des AUV actuels : Remus 100 - AUV [42] (voir la figure 2.9a) et A9E - A18D [43] (voir la figure 2.9b).



(a) REMUS 100-AUV (Photo de Hydroid)

(b) A9-E et A18D de ECA Group

Figure 2.9 – Certains des AUV actuels

2.2.1.3 Hybride ROV-AUV

Le ROV-AUV hybride, comme son nom l'indique, est la combinaison des fonctionnalités et caractéristiques entre ROV et AUV. Plus précisément, il s'agit d'un appareil ROV relié par un câble, mais équipé à des algorithmes de contrôle supplémentaires afin qu'il puisse fonctionner automatiquement et indépendamment, dans le cas où l'opérateur sur le poste de contrôle ne peut pas le superviser. L'avantage de cette méthode est qu'il est possible

d'économiser le coût de l'équipement, de profiter des fonctionnalités automatiques de l'AUV sans utiliser de capteurs coûteux.

La figure 2.10 présente l'AUV/ROV Integra [44], qui permet aux utilisateurs de mener de multiples missions sous-marines, tout en offrant une alternative économique au déploiement d'AUV et de ROV séparés pour des tâches individualisées. Il peut être configuré avec plusieurs capteurs et manœuvré par une plate-forme intuitive et facile à utiliser, accessible depuis n'importe quel appareil connecté au site web. Le véhicule est conçu pour être utilisé dans plusieurs secteurs, notamment la surveillance, la recherche, l'évaluation environnementale et peut effectuer des recherches dans de vastes zones en mode AUV (non attaché) tout en menant des inspections détaillées en mode ROV (attaché). Les utilisateurs peuvent facilement passer du mode AUV au mode ROV en attachant la longe pour contrôler à distance les six degrés de liberté de mouvement du véhicule. Lorsque le véhicule fonctionne en mode autonome, toute la planification de la mission est effectuée dans une application intuitive basée sur Windows. Il peut atteindre une profondeur 300 m.

Figure 2.10 – Aquabotix hybride AUV-ROV

Un autre véhicule hybride sous-marin, le Nereus [45] (voir la figure 2.11), a été construit par le *Woods Hole Oceanographic Institute*. Il a été construit comme un véhicule de recherche capable d'opérer à des profondeurs allant jusqu'à 11 000 mètres, afin de pouvoir explorer le Challenger Deep - le point le plus profond de l'océan. Le fait qu'il s'agisse d'un ROV hybride signifie qu'il peut être piloté par des pilotes à bord via un câble en fibre optique mince. Cela permet des plongées profondes tout en étant très maniable ou capable de parcourir de petites zones pour mener des expériences ou collecter des échantillons de roche et de vie marine.

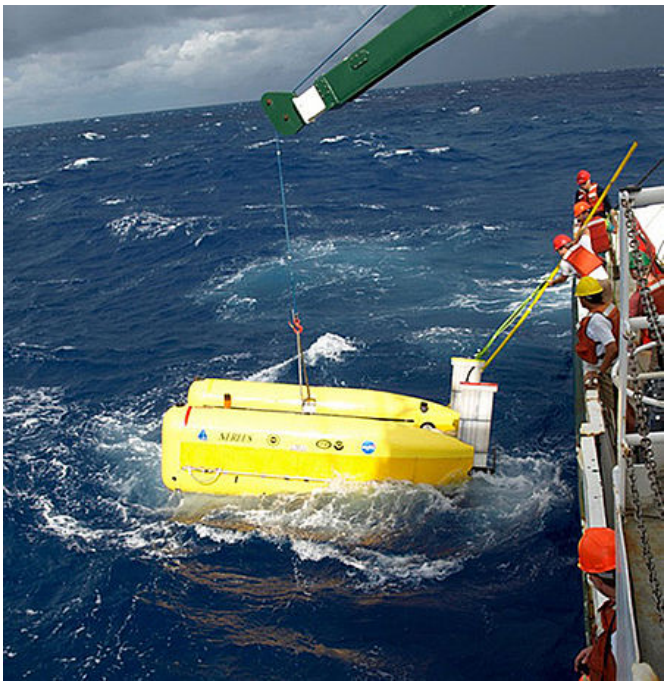


Figure 2.11 – Un véhicule hybride sous-marin Nereus

De plus, le Nereus peut facilement passer en mode automatique, où il peut être utilisé comme véhicule autonome pour surveiller ou explorer de grandes zones du fond de l'océan.

Il existe également le Double Eagle SAROV [46] (voir la figure 2.12), un véhicule équipé de capteurs sophistiqués et modernes. Il peut être contrôlé à distance ou automatiquement, ce qui en fait un hybride ROV/AUV. C'est la combinaison des caractéristiques ROV et AUV qui fait du Double Eagle SAROV un véhicule polyvalent à longue portée adapté à une variété de tâches, y compris : les missions de levés sous-marins ; activités d'identification et d'élimination des mines, détection d'objets sous-marins.

Figure 2.12 – Un véhicule hybride sous-marin Double Eagle (Photo : Saab)

Dans cette thèse, nous nous concentrons également sur la possibilité de convertir des ROV à faible coût en dispositifs hybrides ROV-AUV, en étudiant le développement des algorithmes de contrôle qui leur permettent de fonctionner partiellement automatiquement.

2.2.2 Navigation et localisation de robot sous-marin

Nous précisons d'abord la différence entre la navigation et la localisation. La précision de la navigation est celle avec laquelle l'AUV se guide d'un point à un autre. La précision de la localisation est l'erreur de localisation de l'AUV sur une carte. La navigation et la localisation des AUV posent encore beaucoup de difficultés, principalement en raison de l'atténuation rapide des signaux de haute fréquence et de la nature non structurée de l'environnement sous-marin. Au-dessus de l'eau, la plupart des systèmes autonomes reposent sur les communications radiophoniques ou à spectre étalé et le positionnement global. Cependant, sous l'eau, ces signaux ne se proprobot que sur de courtes distances et les capteurs et communications acoustiques sont plus performants. Selon l'article [47], les techniques de navigation et de localisation UUV peuvent être classées comme sur la figure 2.13.

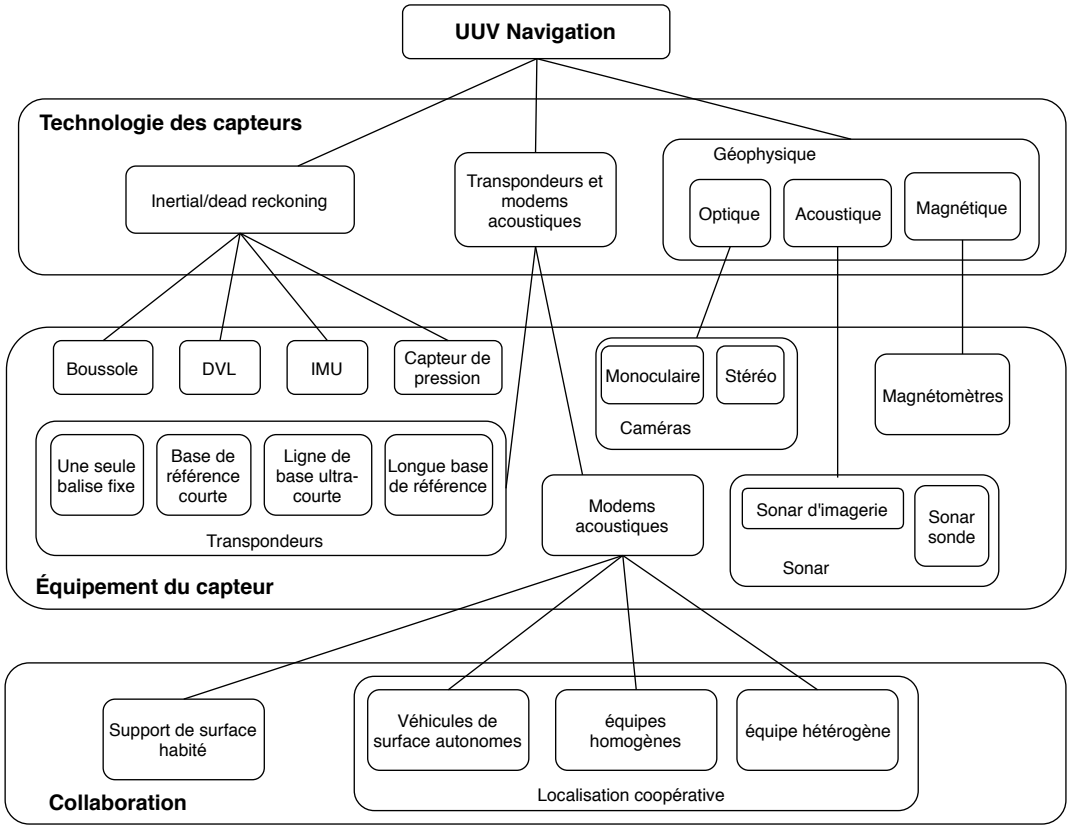


Figure 2.13 – Navigation et localisation d'UUV (selon l'article [47])

En général, ces techniques se classent dans l'une des trois grandes catégories suivantes :

Inertielle/dead reckoning : La navigation inertielle utilise des accéléromètres et des gyroscopes pour une précision accrue afin de propager l'état actuel. Néanmoins, toutes les méthodes de cette catégorie ont une croissance de l'erreur de position qui n'est pas limitée.

Transpondeur acoustique : Les techniques de cette catégorie sont basées sur la mesure du temps de vol (en anglais *Time of Flight* - TOF) des signaux de balises acoustiques ou de modems pour effectuer la navigation. Cependant, les communications acoustiques souffrent encore de nombreuses limitations comme :

- La petite largeur de bande, ce qui signifie que les nœuds de communication ont dû utiliser des techniques d'accès multiple à répartition dans le temps (en anglais Time-division multiple-access ou TMDA) pour partager l'information.
- Le faible débit de données, ce qui limite généralement la quantité de données pouvant être transmises.
- La latence élevée puisque la vitesse du son dans l'eau n'est que de 1500 m/s (lente par rapport à la vitesse de la lumière).
- La vitesse du son variable en raison des fluctuations de la température et de la salinité de l'eau.
- La transmission à trajets multiples grâce à la présence d'une limite supérieure (surface libre) et inférieure (fond marin) couplée à une vitesse du son très variable.
- Le manque de fiabilité, d'où la nécessité d'un système de communication conçu pour gérer les pertes de données fréquentes lors des transmissions.
- Le coût est très important.

La géophysique : Techniques qui utilisent l'information environnementale externe comme référence pour la navigation. Cela doit être fait avec des capteurs et des traitements capables de détecter, d'identifier et de classifier certaines caractéristiques environnementales [48]. Ceci est également étendu dans le cas de nombreux sous-marins. La détection des sous-marins voisins est également considérée comme un cas particulier de l'environnement.

Le type de système de navigation utilisé dépend fortement du type d'opération ou de mission. Dans de nombreux cas, différents systèmes peuvent être combinés pour obtenir des performances accrues. Les considérations les plus importantes sont la taille de la région d'intérêt et la précision de localisation souhaitée.

2.2.3 Conclusion sur le robot sous-marin

Cette section a présenté certaines des principales caractéristiques de l'AUV et du ROV ainsi que leurs avantages et inconvénients. À partir de là, nous proposons d'étudier un AUV-ROV hybride open-source à faible coût. Plus précisément, nous utiliserons un ROV, nommé BlueROV, puis nous développerons des algorithmes de contrôle pour le rendre partiellement automatique. Ce travail permet de convertir un ROV téléopéré en un hybride ROV-AUV autonome.

2.3 Systèmes de multi-robots

Dans cette partie, nous donnerons un aperçu des systèmes de multi-robots, qui comprend des avions, des mobiles, des bateaux, et nous couvrirons en détail les sous-marins multi-robots.

2.3.1 Systèmes de multi-robots aériens

Actuellement, il existe plusieurs applications qui utilisent des avions de drones multi-robots, tels que les Shooting Stars d'Intel, sont de très petits quadricoptères sans pilote - un hélicoptère à quatre rotors. Chaque Shooting Star pèse moins de 330 grammes. Les drones utilisés pour créer des spectacles de lumière sont équipés des éléments les plus essentiels : Des capteurs GPS (système de positionnement global) et une puissante lumière LED qui peut créer plus de quatre milliards de combinaisons de couleurs. Cela permet à ces drones de produire une plus grande variété d'effets que les feux d'artifice et d'offrir des possibilités presque illimitées de raconter des histoires artistiques dans le ciel. Il existe également des drones spécialement conçus pour voler en toute sécurité dans des espaces aériens restreints où les feux d'artifice ne sont tout simplement pas possibles, comme à l'intérieur. Les drones de spectacle lumineux sont généralement contrôlés par des centaines d'ordinateurs distants pour créer des formes et des motifs colorés dans le ciel nocturne. Et à mesure que la technologie informatique progresse, la taille des flottes augmente. Par exemple, en 2015, un titre de record du monde Guinness a été attribué à une flotte de 100 drones. En outre, avec les progrès de la technologie 5G, le suivi de la localisation deviendra plus précis, de sorte que les drones pourront voler plus près les uns des autres pour créer des motifs toujours plus détaillés et plus complexes dans le ciel.

Figure 2.14 – Des drones Intel Shooting Star forment les anneaux olympiques dans le cadre du spectacle de drones de la cérémonie d'ouverture des Jeux olympiques d'hiver PyeongChang 2018. (Photo : Intel Corporation)

2.3.2 Systèmes de multi-robots terrestres

Projet MARS (en anglais *Mobile Agricultural Robot Swarms*) utilise de petits robots opérant en essaim et une solution basée sur les nuages pour planifier, surveiller et documenter avec précision la plantation du maïs. La navigation par satellite et la gestion des données dans le nuage permettent de mener des opérations 24 heures sur 24, avec un accès permanent à toutes les données. La planification du champ nécessaire, des semences, des modèles de semences et de la densité est effectuée via l'application MARS. L'algorithme intelligent OptiVisor [6] planifie le déploiement du robot en fonction des paramètres saisis et calcule les trajectoires optimales des unités impliquées et le temps nécessaire à l'accomplissement de la tâche. Un élément clé du projet MARS est la gestion intelligente des robots déployés. Un système MARS se compose de 6 à 12 unités et peut donc couvrir environ 1 ha/h. Chaque robot est en communication constante avec le contrôleur. La mise en mémoire tampon des données et la communication redondante permettent de couvrir les ruptures de couverture du réseau. Si un robot tombe en panne, les trajectoires de toutes les unités sont automatiquement ré-optimisées et les autres robots prennent le relais.

Figure 2.15 – Un nouveau système Xaver de Fendt utilise des essaims de robots pour la plantation du maïs

2.3.3 Systèmes de multi-robots sous marins

Cette section présentera certains des systèmes actuels de multi-robots sous-marins, et se concentrera sur l'analyse des architectures de contrôle de la coordination des robots en

6. Voir https://www.fendt.com/int/fendt-mars

général et en particulier pour les robots sous-marins.

2.3.3.1 Introduction des systèmes de multi-robots sous-marins

Hydromea[7] a développé un système radio basse fréquence unique conçu pour les essaims robotiques, qui connecte les robots sous-marins comme un réseau radio multi-saut. L'algorithme de planification distribué empêche les perturbations et maximise le débit de communication. De plus, ils ont ajouté un nouveau système de localisation acoustique distribué qui permet à chaque AUV de trianguler sa position par rapport aux autres AUV de la flotte. Ce système est combiné avec le réseau de communication et un GPS capables d'obtenir une position à chaque fois qu'un des AUV est à la surface. De ce fait, chaque robot obtient des informations de position en continu - aucune infrastructure supplémentaire n'est requise. Les AUV sont également équipés de radio et de 3G pour les communications de surface. La combinaison unique de techniques de communication et de localisation évolutives nous permet de mettre en œuvre un algorithme sophistiqué de contrôle des essaims et des formations.

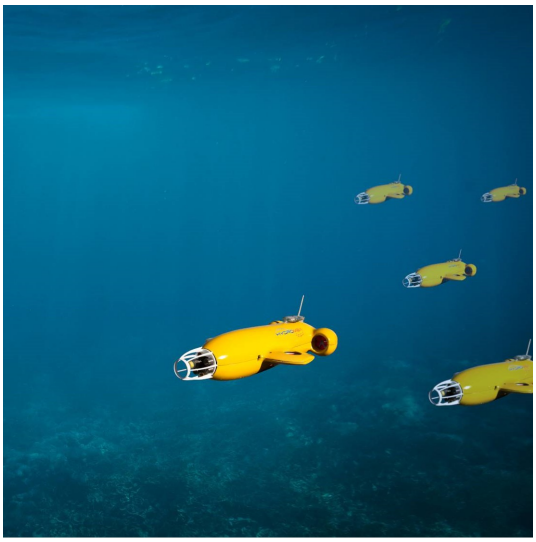


Figure 2.16 – Un essaim de robots sous-marins de Hydromea (Photo : Hydromea)

L'article [49] présente une approche robuste de convoyage de plusieurs robots qui repose sur la détection visuelle du robot leader, permettant ainsi le suivi des cibles dans des environnements 3D non structurés (voir la figure 2.17).

7. Plus d'informations peuvent être trouvées sur https://www.hydromea.com/vertex-autonomous-underwater-swarm/

Figure 2.17 – Convoyage sous-marin multi-robots utilisant le suivi visuel par détection (Photo de [49])

2.3.3.2 Architecture de coordination dans un groupe UUVs

Deux approches sont couramment adoptées pour contrôler les multi-véhicules : une approche centralisée et une approche distribuée. L'approche centralisée est basée sur l'hypothèse qu'une station centrale (installée en bateaux ou prise en charge par un robot) est disponible et suffisamment puissante pour contrôler un groupe entier de véhicules.

L'approche distribuée n'a pas besoin d'une station centrale de contrôle. Les robots doivent être autonomes, conscients de l'environnement pour prendre eux-mêmes des décisions. Cependant, cela rend les systèmes plus complexes. Plus précisément, comment les robots prendront-ils leurs propres décisions ? Les capteurs fourniront-ils les informations environnementales nécessaires à ces décisions ? Comment l'optimisation de la distance et de l'énergie du robot se fera-t-elle ?

Bien que les deux approches soient utilisées selon les situations et les conditions des applications réelles, l'approche distribuée est considérée comme celle ayant un certain nombre d'avantages tels que : l'économie d'énergie, la croissance de performance. Notamment elle s'adapte à la diminution de la bande passante, par rapport à l'approche centralisée qui demande au poste de contrôle de communiquer avec l'ensemble des robots. C'est une des raisons pour laquelle, nous préférons utiliser l'approche distribuée.

a*) *Architecture de coordination centralisée

Dans un schéma centralisé, les actions de tous les véhicules sont décidées par un signal de contrôle de la station, en utilisant toutes les informations disponibles.

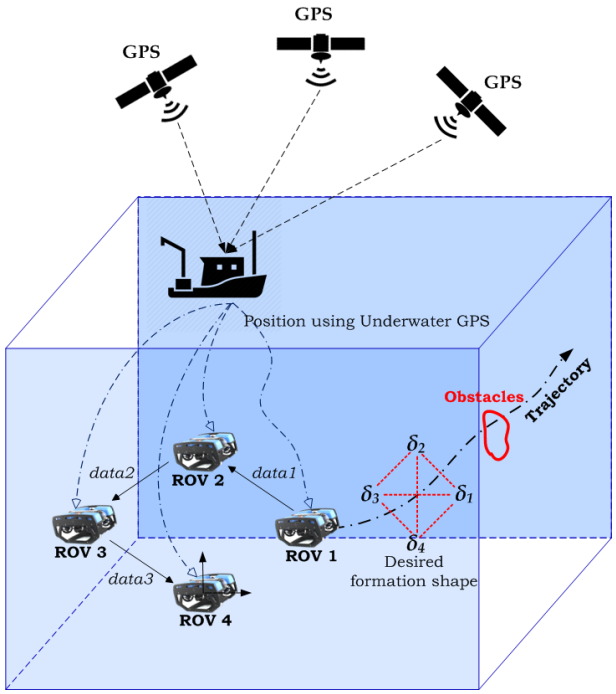


Figure 2.18 – Architecture de coordination centralisée

Cela permet d'élaborer des actions de différents véhicules simultanément, et de faciliter leur synchronisation et optimisation. Néanmoins, cette approche nécessite un calcul potentiellement lourd pour résoudre le problème centralisé unique, ce qui peut limiter le nombre de véhicules pris en charge. Cela exige également un niveau de robustesse plus élevé en matière des liaisons de transmission afin de garder une trace de tous les véhicules. Enfin, au cas où aurait lieu une panne à la station, l'ensemble du système va s'arrêter.

b) Architecture de coordination décentralisée

Dans un schéma décentralisé, le contrôle distribué réduit la charge de calcul dans la situation où chaque véhicule calcule sa propre entrée de commande, et permet donc une meilleure évolution et une certaine robustesse. Si une défaillance se produit sur l'un des véhicules, il est possible de réaffecter la mission sur la flotte restante, ce qui augmente la robustesse vis-à-vis de la mission. Les inconvénients sont à évaluer précisément ; les capacités de calcul disponibles de chaque véhicule peuvent être limitées dans la résolution des problèmes complexes ; les informations disponibles sur d'autres véhicules peuvent être inexactes, incomplètes ou connues avec un retard et donc il existe un risque de collision non négligeable en cas de défaut.

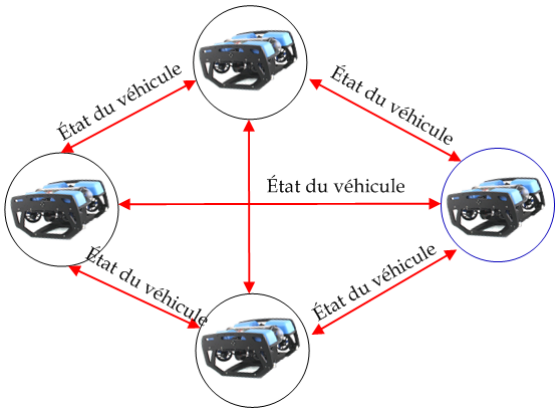


Figure 2.19 – Architecture de coordination décentralisée

Les systèmes s'inscrivant dans le cadre des systèmes distribués peuvent être trouvés dans la littérature [50], [51].

c) ***Architecture de coordination hybride***

L'architecture de coordination hybride est une combinaison de deux architectures : architecture de coordination centralisée et décentralisée. En pratique, l'architecture de coordination hybride est souvent utilisée pour tirer parti des avantages et des inconvénients des deux méthodes qui ont été mentionnés ci-dessus. Dans les architectures centralisées, l'entité superviseuse dispose d'informations globales sur l'environnement, ce qui l'aide à la prise de décision et à la coordination des actions entre les éléments du système multi-robots. D'une part, elle est responsable du contrôle du groupe et elle gère d'autre part la communication de toutes les entités robotiques. Ceci signifie que le système multi-robots dépend entièrement de cette entité. C'est dans ces deux facteurs que réside la limitation de ce type d'architecture car un dysfonctionnement du superviseur ou une incapacité à assurer une de ces deux fonctions engendreront l'interruption de la tâche et l'échec de la mission.

Par contre, les architectures décentralisées quant à elles, répondent mieux à la problématique des environnements inconnus ou dynamiques et ont de meilleurs résultats en termes de fiabilité, de flexibilité, d'adaptabilité et de robustesse. Elles se basent sur la distribution du contrôle (prise de décision et commande) et des ressources d'information et de calcul sur plusieurs entités coopérantes et communicantes. La robustesse de cette architecture réside dans son adaptabilité à l'échec ou à la perte de robots du groupe, ainsi qu'aux problèmes de communication ou de calcul. Cependant, l'absence d'un superviseur et d'une information globale sur l'environnement et l'état global du groupe rend la coordination des robots et l'optimisation de l'exécution de la tâche de plus en plus difficile.

En pratique, beaucoup de systèmes de multi-robots utilisent des architectures de coordination hybrides, par exemple [52], [53].

2.3.4 Conclusion sur les systèmes de multi-robots

Dans cette section, nous avons résumé les systèmes de systèmes multi-robots. En particulier, nous nous concentrons sur les systèmes sous-marins multi-robots. Nous choisissons également une approche d'architecture de contrôle distribuée afin d'augmenter la flexibilité et le fonctionnement indépendant de chaque robot sous-marin en communication limitée ainsi que dans les technologies de positionnement sous-marin.

2.4 Commande formation pour un groupe UUVs

À partir de l'architecture de coordination décentralisée $2.3.3.2b$, nous étudierons le suivi d'une formation, qui est largement appliqué dans la pratique. Nous aborderons ensuite les problèmes de contrôle pour la coordination multi-robots. Plus précisément, nous choisissons une approche de commande distribuée. Un problème important de la coordination distribuée consiste à développer des algorithmes ou des protocoles flux d'information, qui spécifient l'échange d'informations entre un robot et ses voisins, de sorte que le groupe dans son ensemble puisse parvenir à un accord concernant un certain objectif. Ce problème est généralement appelé problème de consensus ou de synchronisation. Cela sera présenté dans la section 2.4.2.

2.4.1 Classification des commandes en formation

Dans cette section, le suivi de la formation de commande sera présenté sous différents angles. Il convient également de noter que ces classifications peuvent ne pas être complètement indépendantes les unes des autres.

2.4.1.1 Classification basée sur la stratégie de contrôle

Le contrôle de la formation d'un groupe de robot mobiles autonomes a suscité beaucoup d'intérêt pour la recherche en raison de ses vastes applications dans de nombreux domaines : les domaines civil et militaire. Dans la littérature, la plupart des approches sur le contrôle de la formation peuvent être classées comme :

Stratégie basée sur un suivi du leader : L'idée de base des approches "leader-follower" est qu'un véhicule est désigné pour suivre la position et l'orientation d'un autre véhicule avec un certain décalage prescrit. L'avantage des approches leader-suiveur est que la spécification d'une quantité unique (le mouvement du leader) oriente le comportement du groupe. Cependant, l'inconvénient de cette méthode est que lorsque le leader du robot tombe en panne, tout le système sera affecté.

Stratégie basée sur structure virtuelle : la formation entière est traitée comme un seul corps rigide. Les mouvements souhaités pour les robots sont déterminés à partir de ceux de la

structure virtuelle. L'approche de la structure virtuelle est facile pour prescrire le comportement de l'ensemble du groupe mais nécessite une large bande passante de communication. L'approche de la structure virtuelle a été utilisée pour le contrôle de la formation des robots mobiles, des vaisseaux spatiaux et des véhicules marins.

Stratégie basée sur le comportement : l'action de contrôle pour chaque véhicule est dérivée d'une moyenne pondérée de chaque comportement souhaité, comme le maintien de la formation, la recherche d'objectifs et l'évitement d'obstacles. Cette approche permet une mise en œuvre décentralisée, mais il est difficile de la formuler mathématiquement.

2.4.1.2 Classification basée sur les variables mesurées et contrôlées

Un problème clé dans le contrôle de la formation qui suscite un intérêt particulier est de savoir comment stabiliser et maintenir une forme géométrique de formation d'une manière distribuée, et ce sera l'un des points centraux de cette thèse. Dans [54], différents types de stratégies de contrôle des formes de formation sont étudiés et comparés en termes de capacité de détection et de topologie d'interaction.

L'approche basée sur la position : Les robots perçoivent leurs propres positions par rapport à un système de coordonnées global. Ils contrôlent activement leurs propres positions pour obtenir la formation souhaitée, qui est prescrite par les positions souhaitées par rapport au système de coordonnées global [55], [56], [57].

L'approche basée sur le déplacement : dans laquelle la formation souhaitée est spécifiée par un certain nombre de positions relatives inter-robots et de contrôleurs linéaires [58], [59].

L'approche basée sur la distance : dans laquelle la formation souhaitée est spécifiée par un certain nombre de distances entre robots et des lois de contrôle non linéaires sont nécessaires [60], [58], [61], [62].

Dans le cas des robots sous-marins, où les méthodes de positionnement et de communication sous l'eau sont limitées, l'approche basée sur la distance ou basée sur la position relative entre robots (système de référence local) semble préférable. Dans nos recherches, afin de pouvoir expérimenter sur de vrais modèles de robots, nous avons choisi une approche basée sur le déplacement.

2.4.2 Problème de consensus

Le consensus sert de principe fondamental pour la conception d'algorithmes de coordination multi-robots distribués. Par conséquent, la recherche d'un consensus a été l'un des principaux axes de recherche dans l'étude de la coordination distribuée multi-robots. Pour combler l'écart entre l'étude des algorithmes de consensus et les nombreuses propriétés physiques héritées des systèmes réels, il est nécessaire et significatif d'étudier le consensus en tenant compte de nombreux facteurs, tels que l'actionnement, le contrôle, la communication, le calcul et la dynamique du véhicule, qui définissent certaines caractéristiques

Tableau 2.1 – Distinctions entre le contrôle de la formation en fonction de la position, du déplacement et de la distance [54].

	Basé sur la position	**Basé sur le déplacement**	**Basé sur la distance**
Variables mesurées	Positions des robots	Positions relatives des voisins	Positions relatives des voisins
Variables contrôlées	Positions des robots	Positions relatives des voisins	Distances entre robots
Systèmes de coordonnées	Un système de coordonnées global	Systèmes de coordonnées locales alignés sur l'orientation	Systèmes de coordonnées locales
Topologie de l'interaction	Pas nécessaire	Connectivité ou existence d'arbre couvrant (*spanning tree*)	Rigidité

importantes des systèmes réels. Dans la littérature, il y a eu de nombreux algorithmes de recherche pour résoudre le problème de consensus. Ils sont appliqués aux systèmes : simple intégrateur [63], [64], [65] double intégrateur [66], [67] ou dynamique d'attitude du corps rigide [68], [69].

Un algorithme de contrôle de consensus typique est conçu comme suit :

$$u_i(t) = \sum_{j=1}^{n} a_{ij}(t)\left[x_j(t) - x_i(t)\right] \tag{2.1}$$

Avec $\dot{x}_i(t) = u_i(t), \quad i = 1, ..., n$ est la cinématique de l'intégrateur-simple de chaque véhicule, $a_{ij}(t)$ est la valeur de la matrice d'adjacence correspondante au temps t. Les détails de l'algorithme consensus peuvent être trouvés dans [70], [71].

La présentation des questions de consensus théoriques et pratiques pour les robots en général nous donnera un aperçu général de l'étude de coordination multi-robots, en particulier pour le cas du robot sous-marin.

2.4.2.1 Théorie des problèmes de consensus

a*) *Les topologies et dynamiques stochastiques des réseaux : Dans les systèmes multi-véhicules, la topologie du réseau entre tous les véhicules joue un rôle crucial dans la détermination du consensus. L'objectif est d'identifier explicitement les conditions nécessaires et/ou suffisantes sur la topologie du réseau pour qu'un consensus puisse être atteint sous des algorithmes correctement conçus. Il est souvent raisonnable de considérer le cas où la topologie du réseau est déterministe sous des canaux de communication idéaux. En conséquence, la recherche principale sur le problème du consensus a été menée sous une topologie déterministe de réseau fixe et de commutation, c'est-à-dire que la matrice d'adjacence $\mathcal{A}(t)$ qui est présentée dans la section 5.1 est déterministe. D'autre part, lorsque

l'on considère les défaillances aléatoires de communication, les chutes aléatoires de paquets et les instabilités des canaux de communication héritées des canaux de communication physique, il est nécessaire et important d'étudier le problème du consensus dans le cadre stochastique où une topologie de réseau évolue en fonction de distributions au hasard.

Le consensus sur une topologie de réseau stochastique a d'abord été étudié en [72], où certaines conditions suffisantes sur la topologie du réseau ont été données pour garantir un consensus avec les systèmes à cinématique d'intégrateur. D'autres résultats de consensus sous une topologie de réseau stochastique ont été présentés dans [73], [74].

b*) *Les systèmes dynamiques complexes : Comme le consensus concerne le comportement d'un groupe de véhicules, il est naturel de considérer la dynamique du système pour les véhicules pratiques dans l'étude du problème du consensus. Bien que l'étude du consensus dans le cadre de diverses dynamiques de systèmes soit due à l'existence de dynamiques complexes dans les systèmes pratiques, il est également intéressant d'observer que la dynamique des systèmes joue un rôle important dans la détermination de l'état final du consensus. Par exemple, un consensus bien étudié des systèmes multi-robots à cinématique simple-intégrateur converge souvent vers une valeur finale constante [75], [76]. Cependant, un consensus pour une dynamique double d'intégrateur pourrait admettre une valeur finale dynamique (c'est-à-dire une fonction temporelle). Ces enjeux importants motivent l'étude du consensus dans le cadre de diverses dynamiques du système [77].

c*) *Vitesse de convergence en temps défini : En plus de l'étude sur le problème du consensus avec les contraintes physiques mentionnées dans les sous-sections précédentes, il est également important d'étudier la performance de contrôle du problème du consensus. Du point de vue du contrôle, il est naturel de proposer des algorithmes de contrôle appropriés et d'analyser la stabilité, et d'optimiser les algorithmes de contrôle proposés sous certains indices de performance de contrôle. Dans cette sous-section, le problème de la vitesse de convergence est passé en revue, ce qui est important pour la mesure de performance pour les algorithmes de consensus.

Dans le prolongement de l'étude de la vitesse de convergence pour le problème du consensus, le consensus à temps fini, c'est-à-dire l'obtention d'un consensus dans un temps fini, a également été étudié récemment. Comparé à la plupart des recherches existantes sur le problème du consensus, le consensus à temps fini démontre une propriété de rejet des perturbations et une robustesse face aux incertitudes.

d*) *Les robots hétérogènes : Outre les problèmes ci-dessus, l'hétérogénéité du modèle dynamique (pour les robots) pose également un défi pour l'étude des algorithmes de contrôle coordonné. Autrement dit, l'algorithme peut être implémenté et utilisé pour de nombreux types différents de modèles de robots.

Bien que l'étude des questions théoriques consensuelles ait été réalisée, très peu de recherches ont été appliquées aux systèmes de robots réels (en particulier pour les robots

sous-marins). En supposant qu'il existe des systèmes de positionnement global et des systèmes de communication robotisés, ce n'est pas vraiment adapté aux robots sous-marins. C'est pour ces raisons que l'extension de la recherche et la mise en œuvre de ces algorithmes pour les robots sous-marins est pour nous un objectif de recherche.

2.4.2.2 Pratique des problèmes de consensus

a*) *Les effets de délais : Le délai apparaît dans presque tous les systèmes pratiques pour plusieurs raisons : vitesse de communication limitée ; heures supplémentaires requis par le capteur pour obtenir l'information de mesure ; temps de calcul requis pour générer les entrées de contrôle ; et temps d'exécution requis pour les entrées en cours d'exécution. En général, le délai reflète une propriété importante héritée dans les systèmes pratiques en raison de l'actionnement, du contrôle, de la communication et du calcul. Bien que le consensus sur le délai ait fait l'objet d'études approfondies, on suppose souvent que le délai est constant ou aléatoire. Cependant, le temps de retard lui-même peut obéir à sa propre dynamique, qui dépend éventuellement de la distance de communication, de la charge de calcul totale et de la capacité de calcul, etc. Par conséquent, il est plus approprié de représenter le retard comme une autre variable du système à prendre en compte dans l'étude du problème de consensus. En outre, il est également important de prendre en compte simultanément le retard et d'autres contraintes physiques dans le processus de l'étude du problème du consensus.

Un algorithme de consensus étudié pour la cinématique à intégrateur unique est donné en 2.1, où l'on suppose maintenant qu'il existe un retard de temps. Deux types de délais, le délai de communication (T_{ij}) et le délai d'entrée (T_i^p) ont été pris en compte. Alors l'équation du système en boucle fermée de système intégrateur simple utilisant 2.1 devient :

$$\dot{x}_i(t) = \sum_{j=1}^{n} a_{ij}(t)\left[x_j\left(t - T_i^p - T_{ij}\right) - x_i\left(t - T_i^p\right)\right] \tag{2.2}$$

b*) *Quantification et données-échantillon : Le consensus quantifié a été étudié récemment avec la motivation du traitement numérique du signal. Ici, le consensus quantifié fait référence au consensus lorsque les mesures sont numériques plutôt qu'analogiques. Par conséquent, l'information reçue par chaque robot n'est pas continue et pourrait avoir été tronquée en raison de contraintes de précision finie numérique comme [78], [79].

En raison des limites des unités de mesure et de contrôle, il est souvent impossible d'acquérir des mesures d'information à une vitesse arbitrairement rapide et d'exécuter les entrées de contrôle instantanément. Par conséquent, les systèmes en boucle fermée sont modélisés de façon hybride. C'est-à-dire que le système physique est décrit en temps continu, tandis que les mesures et les entrées de commande sont décrites par des signaux discrets.

Dans ce cas, le consensus est étudié dans un cadre de données échantillonnées, appelé

consensus de données échantillonnées, qui reflète les limites héritées des unités physiques de mesure et de contrôle. Entre-temps, il est également important de souligner que les algorithmes de consensus des données échantillonnées nécessitent beaucoup moins d'échange d'information et de puissance de calcul que les algorithmes de consensus en temps continu.

c) ***Consensus basé sur des événements :*** Dans la commande programmée, la transmission des données et la mise à jour du contrôleur sont effectuées périodiquement, indépendamment de l'état du système. La période constante d'échantillonnage et de transmission doit garantir la stabilité dans le pire des cas, ce qui limite son applicabilité à une catégorie plus large d'applications d'intérêt pratique. Par exemple, les robots autonomes sont souvent équipés de microprocesseurs à ressources limitées pour communiquer avec d'autres robots par l'intermédiaire du réseau avec une bande passante limitée. Il peut se produire que la période que le microprocesseur et le réseau peut tolérer ne pas être en mesure de garantir la stabilité. De plus, même si la période est acceptable, il s'agit clairement d'un gaspillage du réseau et des ressources de calcul, surtout lorsqu'il y a très peu de fluctuations entre deux données échantillonnées successives. Pour surmonter les inconvénients causés par le caractère conservateur du contrôle programmé dans le temps, un contrôle basé sur les événements est proposé pour les plates-formes numériques. Dans le contrôle basé sur les événements, les instants de transmission des données et de mise à jour du contrôle sont déterminés par certains événements qui sont déclenchés en fonction de règles prédéfinies.

d) ***Les effets asynchrones :*** Dans la plupart des recherches existantes sur le problème du consensus, on suppose que tous les robots mettent à jour leurs états de façon synchrone, ce qui nécessite une horloge synchronisée pour l'ensemble du groupe de robots. Cependant, une telle horloge synchronisée pourrait ne pas exister dans les applications réelles. Cela motive la conception d'algorithmes de consensus de manière asynchrone, c'est-à-dire que chaque robot met à jour ses propres états indépendamment des temps de mise à jour des autres robots.

e) ***Consensus avec la saturation de l'actionneur :*** Dans les systèmes de commande réels, tous les dispositifs d'actionnement de commande sont sujets à la saturation d'amplitude. La force, la tension, le débit et toutes les entrées physiques imaginables dans toutes les applications imaginables de la technique de commande sont finalement limitées. Des détails sur l'effet de la saturation de l'actionneur peuvent être trouvés dans [80], [81].

f) ***Consensus avec fautes :*** Les dysfonctionnements des systèmes, provoqués par l'usure ou les dommages, dans les actionneurs, les capteurs ou d'autres composants du système est inévitablement pour un seul système. Pour les multi-robots, la probabilité que des robots soient confrontés à des défaillances augmente en raison de l'échelle et de la complexité croissante des systèmes. De plus, les systèmes multi-robots peuvent aussi souffrir de défauts topologiques. Comme la topologie de communication dans les systèmes multi-robots joue un rôle important dans le problème de consensus, les défauts topologiques peuvent

influencer la performance globale ou même la stabilité dans les systèmes multi-robots. De même, un défaut survenant dans un seul robot peut également affecter la performance de l'ensemble du système. Par conséquent, lorsque des pannes surviennent dans les systèmes multi-robots, il est nécessaire de trouver la panne et de prendre des mesures pour assurer la performance du système qui correspondent respectivement à la détection et à l'isolation des défauts et le problème de contrôle tolérant aux pannes.

g) Les réseaux avec interaction antagonistes avec problèmes de consensus : Dans un protocole de consensus, un accord entre robots est obtenu grâce aux efforts de collaboration de tous les robots, exprimé par un graphe de communication avec des poids non négatifs. Cependant, comme dans l'article [82] il est possible de parvenir à une forme d'accord également en présence d'interactions antagonistes, modélisées comme des poids négatifs sur le graphe de communication.

Tout comme la présentation des problèmes théoriques du consensus, sa présentation des problèmes pratiques a pour but de nous aider à comprendre les problèmes qui existent lors de la coordination de plusieurs robots sous-marins. Ceci est particulièrement pertinent pour notre objectif de pouvoir déployer des systèmes de contrôle distribués sur des modèles de robots réalistes à faible coût.

2.4.3 Problème des systèmes non linéaires avec des incertitudes

Dans le contexte d'UUV, il existe également des paramètres non linéaires imprévisibles du modèle dynamique. Cela affecte la conception des contrôleurs distribués. Nous supposons que la dynamique d'UUV i est modélisée comme un système linéaire incertain :

$$\dot{\mathbf{x}}_i = A_i \mathbf{x}_i + B_i \left[\tau_{hi} + \mathbf{f}_i\left(\mathbf{x}_i\right) + \mathbf{w}_i\left(t\right)\right] \tag{2.3}$$

où $\mathbf{x}_i \in \mathbb{R}^6$ est le vecteur d'état du robot i ;
$\mathbf{f}_i\left(\mathbf{x}_i\right) \in \mathbb{R}^3$ est l'incertitude inconnue du robot ;
$\mathbf{w}_i\left(t\right) \in \mathbb{R}^3$ représente la perturbation inconnue correspondante ;
$A_i \in \mathbb{R}^{6\times 6}$ et $B_i \in \mathbb{R}^{6\times 3}$ sont des matrices connues, et (A_i, B_i) est supposé stabilisable.

a) Approche utilisant le contrôle adaptatif et le réseau neuronal qui compensent tous les types d'incertitudes du modèle : l'article [83] présente un contrôle de la formation des leaders et des suiveurs des véhicules marins de surface autonomes non actionnés avec un couple limité. Dans ce cas, un réseau neuronal et des techniques de contrôle adaptatives robustes sont également adoptés pour préserver la robustesse du contrôleur contre les dynamiques non modélisées et les perturbations environnementales induites par les vagues et les courants océaniques. De plus, l'article [84] présente un contrôleur de formation distribué pour un groupe de véhicules sous-marins, dont la dynamique est sujet aux non-linéarités, aux incertitudes paramétriques $\mathbf{f}_i\left(\mathbf{x}_i\right)$ et aux perturbations externes $\mathbf{w}_i\left(t\right)$. Ce

schéma de contrôle de la formation distribuée est proposé en combinant la méthode de linéarisation de la rétroaction et la théorie de la compensation robuste. L'article [85] présente un contrôle adaptatif de la formation pour les AUV sous-actionnés dans des conditions incertaines. Le contrôleur n'utilise que la distance relative, l'angle et la direction de cap du leader. Récemment, des articles ont été publiés sur l'application du réseau de neurones à l'étude du contrôle de suivi de trajectoire des paramètres des véhicules sous-marins autonomes et des perturbations externes [86], [87], [88], [89].

De plus, la commande adaptative distribuée de suivi de consensus asymptotique basée sur un réseau neuronal pour les systèmes multi-robots non linéaires avec quantification d'entrée et défauts d'actionneur est étudiée en [90], [91].

Comme exemple d'application connu, le contrôle distribué de la formation à l'aide de potentiels artificiels et d'un réseau neuronal est appliqué pour les robots spatiaux. Ils sont attachés à un filet de pêche magnétique pour attraper des débris spatiaux en orbite [92] (voir la figure 2.20).

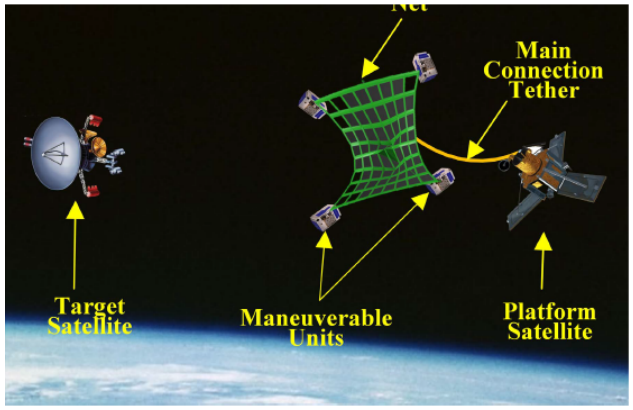


Figure 2.20 – Robot à filet spatial attaché (Photo : [92])

Dans les articles [93],[94],[95], le contrôle adaptatif de consensus pour une classe de systèmes multi-robots non linéaires à retardement temporel utilisant des réseaux neuronaux est étudié. Et également dans les articles [96],[97], les algorithmes de consensus de la formation de multiples-robots dans des conditions de bruit et de retardement dans le temps sont présentés. De plus, on trouve également l'introduction d'un contrôle de consensus adaptatif distribué de systèmes hétérogènes multi-robots chaotiques avec des retards de temps inconnus dans [98].

L'article [99] présente un nouveau schéma de contrôle de l'apprentissage en formation avec de nouveaux algorithmes distribués de contrôle qui est proposé pour le contrôle de formation multi-AUV. Ce nouveau schéma se distingue des nombreuses techniques de contrôle de formation existantes par les aspects suivants ; il permet de surmonter les deux problèmes difficiles liés à l'hétérogénéité et à l'incertitude non linéaire pour le contrôle de la formation d'UUV multiples, il réalise la conception et l'implémentions des contrôles

distribués d'une manière entièrement distribuée sans impliquer aucune information globale. Il permet d'obtenir le système de contrôle de formation distribué avec une capacité distinctive d'adaptation à un AUV incertain dynamique, cela grâce à un apprentissage local précis, à des connaissances représentation/stockage et la réutilisation de l'expérience.

b*) *Approche utilisant le contrôle adaptatif basé sur la théorie de Lyapunov et la technique du "backstepping" : l'article [100] a présenté une méthode de conception pratique, qui est développée pour le contrôle coopératif du suivi de plusieurs véhicules sous-marins autonomes (AUV). Chaque AUV est modélisé par un système dont les paramètres varient dans le temps, dont la dynamique non linéaire est inconnue et dont les perturbations sont inconnues. Un contrôleur distribué adaptatif robuste est conçu pour chaque AUV de sorte que tous les AUV se synchronisent finalement sur les trajectoires souhaitées avec la formation requise. De plus, ces contrôleurs sont distribués en ce sens que la conception du contrôleur pour chaque AUV ne nécessite qu'un état relatif en formation entre lui-même et ses voisins. Dans [101], l'article a présenté un protocole de contrôle garantissant l'établissement de la formation avec des performances prescrites tout en évitant les collisions malgré la présence de perturbations externes et l'incertitude du modèle dynamique. L'article [59] traite de la conception des gains de contrôle distribués pour un consensus dans les systèmes multi-robots avec une dynamique non linéaire en combinant l'approche de la structure variable et la méthode adaptative. Cette conception est basée uniquement sur les informations locales de la structure du réseau. Puis, une loi adaptative distribuée est proposée pour chaque robot suiveur en fonction des informations locales des robots voisins et du robot leader si ce robot suiveur est informé.

Comme indiqué ci-dessus, les robots sous-marins sont considérés comme des systèmes non linéaires. Dans la littérature, il existe également en études de contrôle pour ce système non linéaire. Cependant, dans le but d'avoir une approche facilement personnalisable pour de nombreux types de robots différents, nous avons choisi une approche de réseau de neurones. Bien que des études théoriques des réseaux de neurones aient été effectuées, leur mise en œuvre pour des systèmes réels, en particulier sur des systèmes embarqués de robots sous-marins à bas coût reste difficile.

2.4.4 Problème de navigation et localisation

Actuellement, le problème du positionnement et de la communication sous l'eau pour les robots sous-marins présente encore de nombreuses limitations technologiques ainsi que de coût. L'article [60] étudie le problème du suivi des formations pour les systèmes multi-robots, pour lesquels un schéma estimateur-contrôleur distribué est conçu en s'appuyant uniquement sur les systèmes de coordonnées locales des robots de telle sorte que le centroïde de la formation contrôlée suive une trajectoire donnée. L'article [102], [103] aborde le sujet du contrôle de la formation des véhicules sous-marins, en utilisant soit des mesures de

position relative inter-véhicules, soit des mesures de distance uniquement.

L'article [101] traite du problème du contrôle de la formation basé sur la distance pour les multiples véhicules sous-marins autonomes (AUV) dans une architecture leader-suiveur. Le véhicule sous-marin autonome leader est chargé de suivre une trajectoire souhaitée et les véhicules sous-marins autonomes suivants tentent de mettre en place une structure de formation prédéfinie en atteignant des distances spécifiques entre leurs voisins AUV, tout en évitant les collisions et en permettant en même temps une localisation relative. Plus précisément, un protocole de contrôle décentralisé d'une complexité minimale est proposé, qui permet d'établir des formations prescrites, arbitrairement rapides et précises. Le signal de commande de chaque véhicule est calculé en fonction de la position relative de ses voisins et de sa propre vitesse uniquement, qui peuvent facilement acquises par les capteurs embarqués sans nécessiter de communication réseau explicite.

L'article [104] présente un nouveau système de suivi pour les véhicules sous-marins autonomes (AUV) naviguant en formation serrée, basé sur la vision et l'utilisation de marqueurs lumineux actifs. La localisation acoustique peut être très efficace sur des distances moyennes à longues, mais elle n'est pas aussi avantageuse sur de courtes distances lorsque la sécurité des véhicules exige une plus grande précision et des taux de mise à jour plus élevés. Le système proposé permet l'estimation de la pose d'un véhicule cible à courte distance, avec une grande précision et une grande vitesse d'exécution. Pour étendre le champ de vision, une caméra omnidirectionnelle est utilisée. Cette caméra offre une couverture complète de l'hémisphère inférieure et permet de suivre simultanément plusieurs véhicules dans des positions différentes.

2.4.5 Problème d'évitement de collision et d'obstacle

En pratique, des collisions entre robots peuvent se produire lorsqu'ils travaillent ensemble, ce qui est très indésirable. C'est pourquoi l'évitement des collisions a été une autre question importante pour un contrôle coopératif réussi. En plus des problèmes mentionnés ci-dessus, l'environnement sous-marin est également un environnement avec une structure topographique complexe, de sorte que le groupe d'UUV doit également être en mesure d'éviter les collisions avec des obstacles pendant le mouvement, ainsi que de pouvoir éviter les collisions entre eux lors de la formation de la forme souhaitée.

L'article [105] a présenté un nouveau schéma de contrôle robuste basé sur un réseau neuronal combinant le compensateur adaptatif et le gain de contrôle adaptatif afin de l'évitement d'obstacles et collision. En outre, l'article [106] a introduit un contrôle de la formation et une prévention des collisions pour les systèmes multi-robots basés sur l'estimation de la position. L'article [107] a également utilisé la fonction potentielle avec des paramètres de réglage en ligne pour garantir l'évitement des collisions entre robots lors de la formation. L'article [108] présente une méthode distribuée pour la navigation d'une

équipe de robots tout en reconfigurant leur formation pour éviter les collisions. L'article [109] présente une méthode adaptative de réseau neuronal à carte auto-organisée pour le contrôle distribué de la formation d'un groupe de véhicules sous-marins autonomes. Grâce à cela, un groupe d'AUV peut éviter les obstacles et changer de forme en cas de nécessité.

2.4.6 Conclusion sur la commande formation pour un groupe UUVs

Cette section a présenté la classification des algorithmes de contrôle distribué pour la coordination multi-robots, ainsi que les problèmes théoriques et pratiques rencontrés dans l'étude des algorithmes de la coordination entre robots. En particulier, nous présentons des problèmes de navigation et de localisation de robots sous-marins. En raison de cette limitation, combinée à l'analyse de la section 2.3, nous avons choisi une approche pour développer des algorithmes de contrôle basés sur des signaux mesurables (ou estimés). Cette approche sera démontrée au chapitre 7, le programme sur l'expérimentation des robots sous-marins réels. Par ailleurs, par rapport aux recherches ci-dessus, nous souhaitons présenter un scénario complet pour le robot sous-marin. C'est-à-dire qu'en plus du fait qu'ils peuvent former et suivre la formation, ils sont également intégrés en incorporant des algorithmes de contrôle pour éviter les collisions entre robots et entre robots et obstacles pour aider les robots à fonctionner automatiquement. Cela vise également à convertir un ROV Open Source à faible coût en ROV-AUV hybride.

3 Méthodologie intégrée

Ce chapitre présente la méthodologie de recherche basée sur le modèle en spirale. On introduit ensuite un framework pour déployer et tester les concepts proposés. En outre, ce chapitre présente également l'utilisation de AADL (en anglais *Architecture, Analyse et Langage de Conception*) pour analyser les performances de fiabilité et temps du robot sous-marin réel - BlueROV, ainsi que pour la validation des concepts proposés.

3.1 Méthodologie d'ingénierie de systèmes

3.1.1 Méthodologie itérative d'ingénierie de systèmes

Nous supposons d'abord que cette thèse c'est comme un projet où nous ne savons pas par où commencer et comment commencer ? Qu'est-ce qu'est le modèle sous-marin ? Qu'est-ce qu'est la coordination des contrôles ? Quelle approche de conception peut nous aider à commencer à voir les résultats rapidement ? Et elle a également la capacité de se ré-étendre ainsi que la capacité de réutiliser des composants précédemment étudiés ?

Sur la base de la méthodologie MBSE, nous présentons la méthodologie d'ingénierie de système de robot sous-marin comme le montre la figure 3.1.

Comme mentionné dans la section 1.2, nous définissons 4 cibles à atteindre (équivalents aux exigences de l'étape d'analyse de la méthodologie MBSE) :

— Tout d'abord, étudier le contrôle distribué d'un ensemble de robot sous-marins pour la mise en formation et le suivi de trajectoire.
— Deuxièmement, étudier l'évitement des collisions pendant le processus de formation.
— La troisième est l'étude de l'évitement des obstacles sur la trajectoire.
— Enfin, la construction des expérimentations à bas coût pour prouver les concepts proposés dans l'environnement de recherche.

Avec les cibles définies et sur la base des caractéristiques de la méthode d'analyse orientée objet, nous définissons 3 cycles de recherche correspondant à 3 modèles dynamiques de

robots sous-marins du simple au complexe. Ces 3 cycles correspondent également à 3 cycles dans une spirale étendue.

— Cycle 1 : Étude de **Modèle de simulation simple**. Plus précisément, au cours du premier cycle, nous considérons le modèle de robot UUV comme une équation linéaire. Après cela, nous avons cherché à créer des algorithmes de contrôle des coordonnées et à éviter les collisions entre robots et entre robots et obstacles. Ces algorithmes seront rapidement testés sur les logiciels Matlab / Simulink.
— Cycle 2 : Étude de **Modèle de simulation complet**. Correspondant au deuxième cycle, nous avons étendu notre étude au modèle de robot en tant que non-linéaire avec des paramètres dynamiques inconnus et un bruit modélisant l'influence de l'environnement. L'étude de ce modèle permet au modèle d'être plus proche du modèle réel des robots UUV. Dans cette étape, les algorithmes seront testés et validés pour le modèle de simulation BlueROV sur le logiciel de simulation ROS / Gazebo.
— Cycle 3 : Étude de **Modèle robot sous-marin réel - BlueROV**, qui est Open-Source et low-cost. Correspondant au troisième cycle, nous étudierons le déploiement des concepts proposés sur robot sous-marin réel - BlueROV.

Avec l'utilisation de la méthodologie itérative d'ingénierie de systèmes, nous pouvons clairement montrer les étapes du processus de recherche. En outre, il est facile d'identifier les tâches, ainsi que les modules réutilisables pour les prochaines étapes. Avec cette approche, il est possible de rendre la recherche programmée, étape par étape, des modèles simples aux modèles complexes.

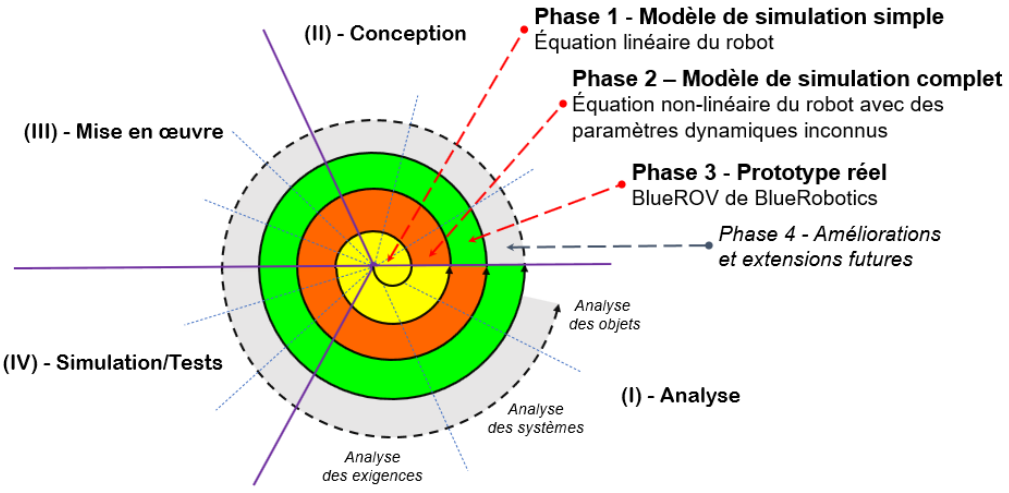


Figure 3.1 – Méthodologie itérative d'ingénierie de système robot sous-marin

3.1.2 Proposition d'une carde de l'étude

En plus de l'introduction de la méthodologie itérative d'ingénierie de système du robot sous-marin, nous avons également construit un framework pour pouvoir faire la transformation entre les modèles de recherche : Modèle comportement, Modèle d'architecture et Modèle Robots. En utilisant ce framework, nous pouvons rapidement implémenter la Vérification et Validation des algorithmes de contrôle ainsi que des algorithmes évitent les collisions entre robots et entre robots et obstacles (voir la Figure 3.2).

Dans le modèle de comportement, on trouve les algorithmes qui ne dépendent pas d'un modèle particulier, tels que les algorithmes d'évitement des collisions, les algorithmes d'évitement des obstacles et les algorithmes de contrôle qui sont développés pour vérifier rapidement les performances, basés sur Matlab/Simulink. Nous allons ensuite transférer les signaux de sortie de ces algorithmes dans le modèle robot, qui exécute des prototypes virtuels ou réels et s'appuiera sur le système d'exploitation sous *Robot Operating System* (ROS) et Gazebo. Le modèle sur Matlab/Simulink peut se connecter directement avec le simulateur de simulation Gazebo, ou peut-être à partir de modèles sur Matlab/Simulink nous pouvons générer des bibliothèques C++, qui peuvent être utilisées comme une bibliothèque dans un environnement ROS. Le simulateur de Gazebo reçoit et envoie des messages sur les sujets suivants. Il reçoit des commandes de vitesse de Matlab/Simulink, sous forme de messages de type ***geometry_msgs/Twist*** dans la topic ***/commands/velocity***. Il envoie des informations d'odométrie à Matlab/Simulink, sous forme de messages de type ***nav_msgs/Odometry*** au sujet ***/odom***. L'utilisation de modèles algorithmiques sur Matlab/Simulink nous aidera à tester et vérifier rapidement les exigences.

De plus, le modèle d'architecture est utilisé à partir de AADL pour analyser les performances de fiabilité et temps du prototype réel - BlueROV, ainsi que pour la validation des concepts proposés. En utilisant les modèles AADL pour écrire la spécification des exigences en temps, performance et fiabilité, nous pouvons déterminer avec précision les exigences ainsi que les contraintes de liaison des composants du système, qui sont décrites au niveau de l'abstraction et de la généralisation. Il convient de noter qu'avec l'utilisation d'AADL, le but est uniquement à ce stade d'analyser l'architecture du système de manière générale ; ce qui peut déjà être déterminé à partir de l'analyse des algorithmes de contrôle ainsi que des algorithmes de positionnement et de navigation pour robot sous-marin. À cette étape, nous n'entrerons pas dans les détails de la mise en œuvre des algorithmes, mais nous nous concentrerons uniquement sur l'analyse de la fonctionnalité des algorithmes ainsi que de la relation entre eux dans l'ensemble du système.

À partir des modèles AADL définis, il existe deux approches pour tester la simulation et la réalisation de modèles UUV.

— Premièrement, les modèles AADL seront convertis en fichiers XML. Ensuite, les fichiers XML générés par l'ingénierie itérative seront générés dans des bibliothèques

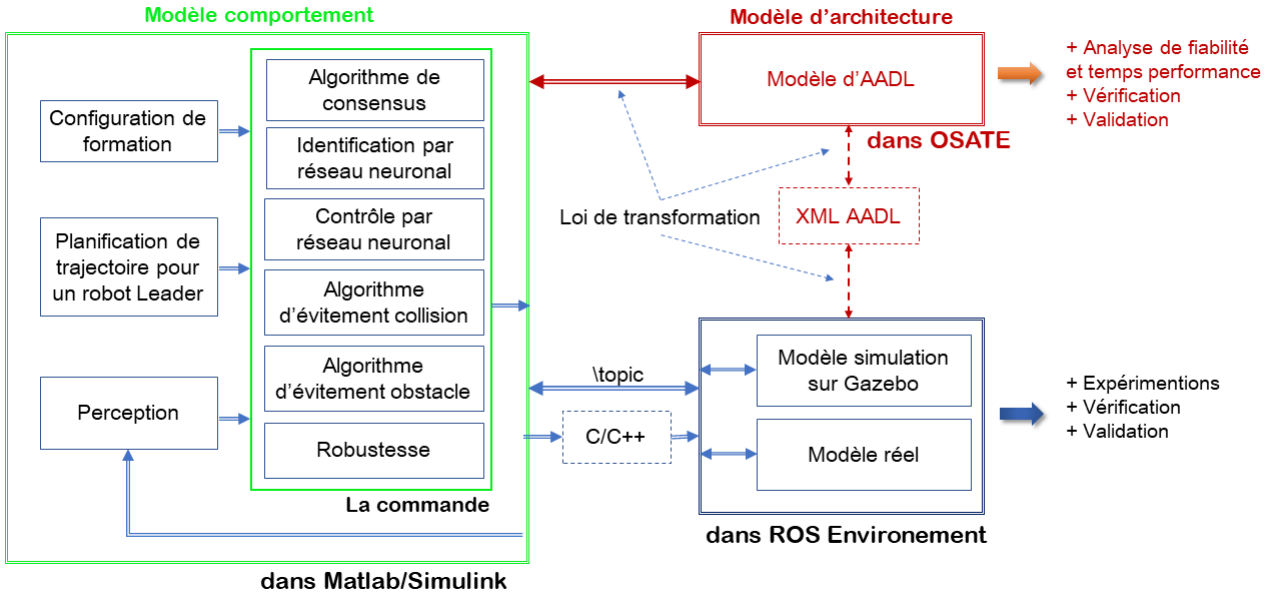


Figure 3.2 – Proposition d'une carde de l'étude et la transformation modèles

C/C++, qui seront implémentées en utilisant l'environnement ROS. Cependant, le générateur de code de cette méthode est créé manuellement en Python. Des détails sur cette approche peuvent être consultés dans [110] et [111].

— Deuxièmement, l'utilisation des plate-formes disponibles. En fait, nous avons choisi une seconde approche. Dans cette approche, nous utilisons les règles de transformation de modèles appliquées au modèle AADL afin d'avoir les modèles sur Matlab/Simulink. De plus, le modèle sur Matlab/Simulink peut être directement connecté au simulateur Gazebo [25] pour s'adapter aux robots virtuels et aux robots ROS actuels, ou à partir des modèles sur Matlab/Simulink, nous pouvons générer des bibliothèques C/C++, qui peuvent être utilisées comme bibliothèque dans un environnement ROS.

3.2 Modèle d'architecture BlueROV en utilisant l'AADL

Cette section décrit l'utilisation de l'AADL pour pouvoir analyser l'ensemble du système de contrôle du robot sous-marin, avec les 4 cibles définies ci-dessus (voir la section 3.1.1). Ceci est également considéré comme un tronc commun pour l'analyse des trois cycles de l'étude de la figure 3.1, et sera détaillé dans les chapitres 5-6-7.

3.2.1 Introduction de l'AADL

Le langage AADL (en anglais Architecture Analysis and Design Language ou AADL) est un langage de modélisation d'architecture qui capture les aspects matériels (*hardware*)

et logiciels (*software*) d'un système et leur interaction. Le langage s'adresse principalement aux systèmes en temps réel critiques pour la sécurité où les capteurs et les actionneurs sont étroitement couplés à des composants logiciels et facilitent l'analyse de l'interaction entre les composants matériels et logiciels.

L'AADL supporte deux notations : graphique et textuelle. Cependant, la version textuelle a plusieurs des avantages. Elle est totalement standardisée sans ambiguïté. Dans la version graphique, il n'y a pas de méthode standard pour représenter une propriété (et honnêtement, il peut être assez difficile de représenter graphiquement toutes les propriétés AADL). La version textuelle est finalement ce que les outils liront et traiteront. C'est aussi la représentation commune qui est interopérable entre les outils. Si nous utilisons plusieurs outils AADL, nous devrons probablement au moins comprendre la version textuelle. Les couches graphiques peuvent être inefficaces lors de la navigation dans un modèle et la mise en page est souvent mauvaise. Un modèle textuel permet d'éviter ces problèmes. Avec la version textuelle, on n'a pas besoin d'éditeurs souvent insuffisamment développés. Nous pouvons même écrire des modèles en utilisant de bons éditeurs de texte tels que Vim ou Emacs.

3.2.2 Modèle d'architecture BlueROV en utilisant l'AADL

En particulier, nous appliquerons ce framework pour la commande de la formation multi-robots sous-marins BlueROV dans le cas d'étude. La plate-forme sous-marine BlueROV qui est le Remotely Operated Vehicle (ROV) de BlueRobotics est basée sur le logiciel Open-source ArduSub et le pilote automatique Pixhawk. La figure 3.3 présente une représentation simplifiée AADL du système BlueROV à l'aide de composants AADL et de connexions de ports de données et d'événements. Nous avons pris certaines décisions en matière de conception architecturale en créant cette représentation. Tout d'abord, puisque nous n'avons pas l'intention de modéliser en détail des capteurs tels que la pression, baromètre, accéléromètre, boussole, gyroscope, capteur de température et de tension actuelle, modèle acoustique, modèle de communication, unité GPS, actionneur de direction, nous les représentons comme des dispositifs. En revanche, en représentant l'interface pilote et les unités d'affichage comme des systèmes, il est possible de développer des modèles détaillés de ces composants. Nous représentons le contrôleur BlueROV comme un processus AADL, reflétant la décision que la fonctionnalité de contrôle sera implémentée dans le logiciel. Nous incluons les ressources matérielles de calcul (c.-à-d. les processeurs), mémoire et bus) dans la représentation. Au départ, nous utilisons un seul processeur, une unité de mémoire et un bus. Cependant, pour atteindre les capacités des autres versions du BlueROV, des ressources matérielles supplémentaires sont nécessaires. Pour simplifier la présentation, nous avons inclus des ports de données et d'événements explicites et des connexions où le signal de désengagement est représenté par des ports d'événements et des connexions.

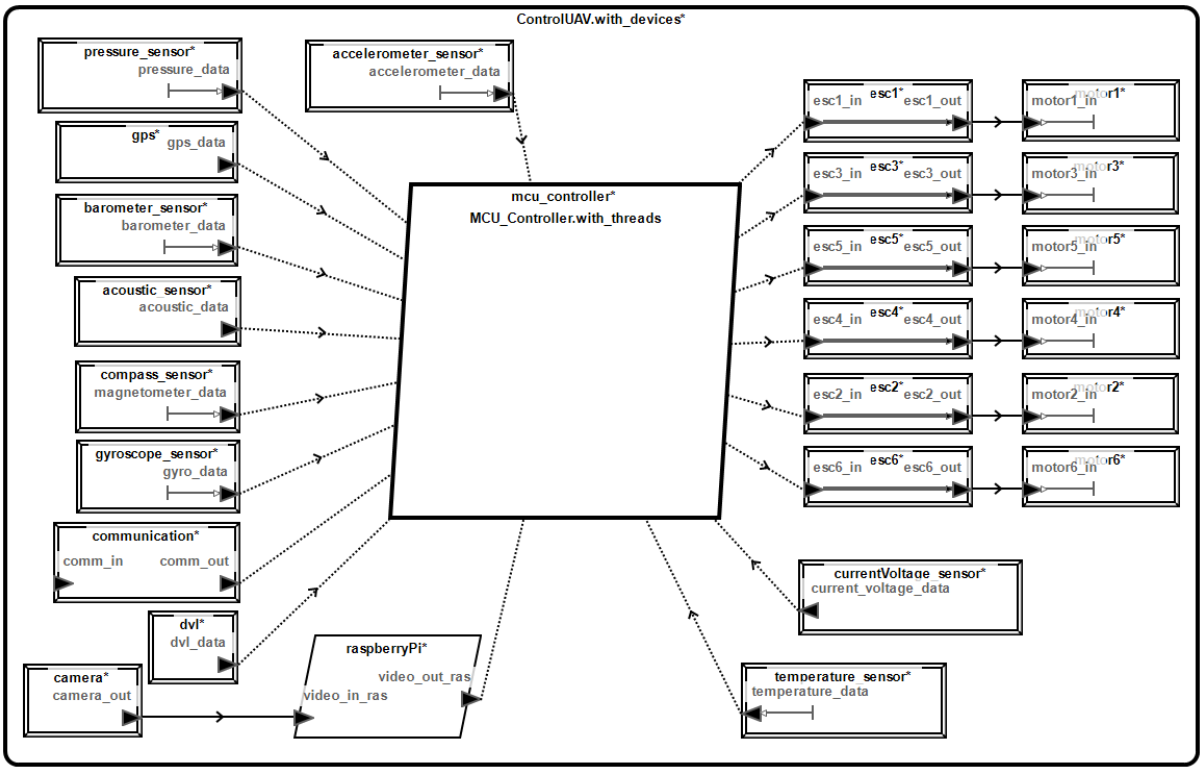


Figure 3.3 – Composants AADL pour le BlueROV dans l'outil OSATE

Pour tous les autres transferts, nous utilisons des ports de données et des connexions d'échantillonnage. Nous n'avons montré aucune connexion d'accès physique pour aucun élément du système.

La figure 3.4 montre graphiquement un exemple de déclaration d'implémentation pour le contrôleur MCU de type procès dans le modèle AADL pour une coordination de plusieurs UUVs. Il y a huit sous-composants listés pour le processus :

— traitement d'images : Ce thread reçoit les vidéos de la caméra embarquée montée devant BlueROV. En supposant qu'à l'aide de certains algorithmes de traitement d'image, nous pouvons calculer la position relative des obstacles pour BlueROV, ainsi que calculer la distance entre eux.

— fusion de capteurs : Ce thread reçoit les signaux des capteurs, puis traite et calcule les données d'orientation, les données de position, les données de vitesse, les données d'accéléromètre des robots. Les algorithmes du filtre de Kalman, AHRS (en anglais *Attitude and Heading Reference System*), etc. peuvent être adressés pour cela.

— planification du cheminement : Ce thread va recevoir les way-points et ensuite utiliser des algorithmes pour calculer la trajectoire désirée que BlueROV doit déplacer. Les algorithmes peuvent être Bézier ou autres.

— algorithme d'obstacle d'évitement : Après avoir reçu la position relative de l'obstacle

au BlueROV, un certain nombre de fonctions telles que les fonctions potentielles sont utilisées pour aider les robots à éviter les obstacles dans le rayon de l'installation précédente.

— algorithme de collision d'évitement : Une coordination du contrôle de plusieurs BlueROVs en fonction de la formation peut conduire à des collisions entre les robots à l'étape initiale. Par conséquent, en passant par ce bloc, on évite les problèmes des collisions.

— algorithme de consensus : C'est le fil qui exécute en utilisant l'algorithme de consensus. Les BlueROVs peuvent se déplacer dans une configuration donnée.

— contrôle total : Ce thread est une combinaison de signaux de contrôle à partir d'un algorithme de collision d'évitement de trois threads, algorithme d'obstacle d'évitement et consensus algorithme. Il est ensuite envoyé à la matrice de répartition.

— matrice de répartition : Lors de la réception d'un signal de commande, à partir du total de contrôle, ce bloc calculera les signaux de commande pour chaque moteur à travers la matrice d'affectation.

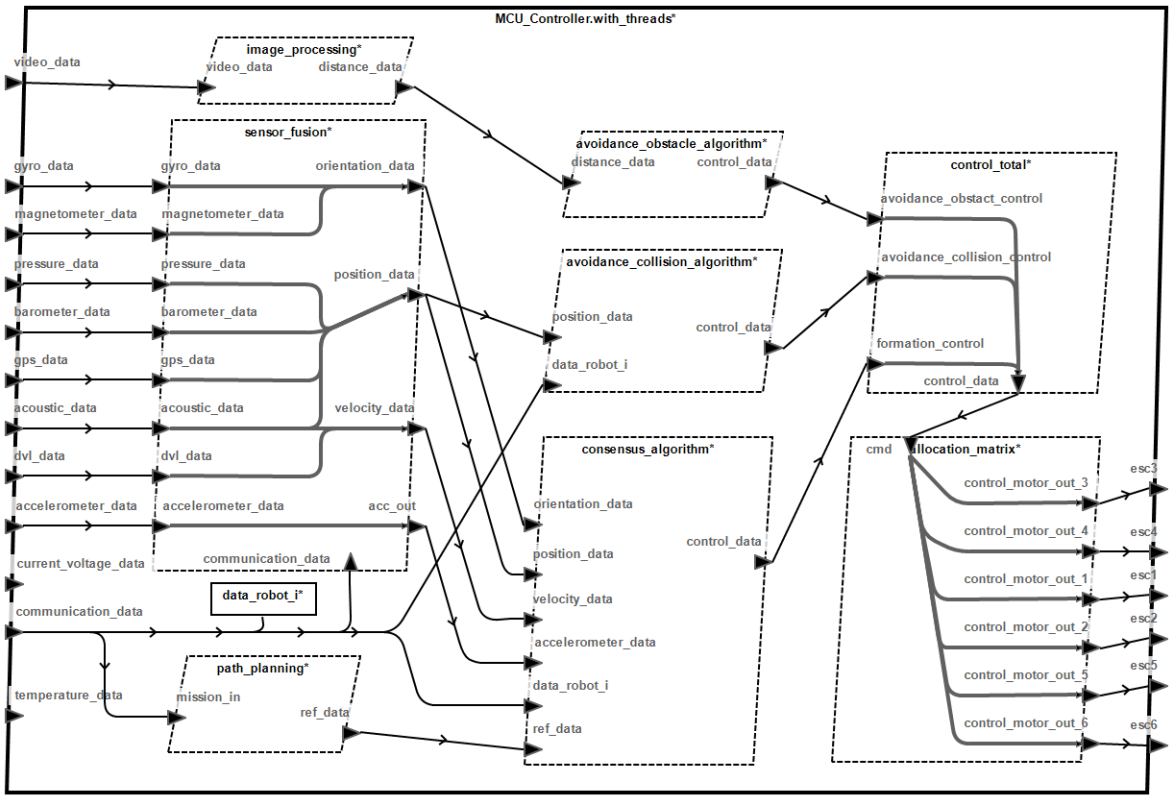


Figure 3.4 – Suivi de la formation de multi-BlueROV dans OSATE

Nous avons utilisé un plug-in pour OSATE (en anglais *Open Source AADL Tool Environment*) qui effectue une analyse de latence de flux dans le plus mauvais des cas. Ce plug-in détermine la latence des implémentions de flux déclarées pour les composants et la

compare à la latence spécifiée par la spécification de flux correspondante du composant. La figure 3.5 illustre un exemple de rapport de flux de bout en bout généré automatiquement par OSATE.

Latency analysis with preference settings: asynchronous system/major partition frame/worst case as deadline/best case as empty queue

Latency results for end-to-end flow 'gyro_to_motor' of system 'ControlUAV.with_devices'

Result	Min Specified	Min Actual	Min Method	Max Specified	Max Actual	Max Method	Comments
device gyroscope_sensor		0.0ms	first		0.0ms	first	
device gyroscope_sensor		6.0ms	processing time		8.0ms	processing time	Using execution time as deadline
connection gyroscope_sensor.gyro_out -> mcu_controller.gyro_in		0.0ms	no latency		0.0ms	no latency	
process mcu_controller		0.0ms	no latency		3.0ms	deadline	
connection mcu_controller.esc1_out -> esc1.esc1_in		0.0ms	no latency		0.0ms	no latency	
device esc1		6.0ms	processing time		8.0ms	processing time	Using execution time as deadline
connection esc1.esc1_out -> motor1.motor1_in		0.0ms	no latency		0.0ms	no latency	
device motor1		1.0ms	processing		3.0ms	deadline	
Latency Total	0.0ms	13.0ms		0.0ms	22.0ms		
Specified End To End Latency		0.0ms			0.0ms		
End to end Latency Summary							
WARNING	Expected end to end						

Figure 3.5 – Exemple de rapport de flux de bout en bout pour un capteur gyroscopique

	A	B	C	D	E	F	G	H
1	Componer	Error Model Element	Hazard T	Description	Crossreference	Failure	Failure Effe	Operation
2	camera	"ItemOmission on camera_out"		"No picture from the camera"	"N/A"	"ItemOmission"		"all"

```
------ Camera ------------------------------------------------
device Camera
    features
        camera_out: out data port;
    annex EMV2 {**
        use types ErrorLibrary;
        error propagations
        camera_out : out propagation{ItemOmission};
        flows
            ef0: error source camera_out{ItemOmission};
        end propagations;

        properties
            emv2::occurrencedistribution =>[probabilityValue => 0.01e-4; Distribution => Poisson;]
                    applies to camera_out.itemomission;
            emv2::severity => ARP4761::Major applies to camera_out.itemomission;
            emv2::likelihood => ARP4761::Probable applies to camera_out.itemomission;
            emv2::hazards => ([ crossreference => "N/A";
                    failure => "ItemOmission"; phases  => ("all");
                    description => "No picture from the camera";
                    comment => "Would impact the detection of obstacle"; ])
                applies to camera_out.itemomission;
        **};
end Camera;
```

Figure 3.6 – Exemple de rapport d'évaluation du risque fonctionnel pour une caméra

La figure 3.6 montre un exemple de rapport d'évaluation du risque fonctionnel pour une caméra. Nous avons créé des types d'erreurs qui spécifient liées à une caméra. Par souci de simplicité, seuls trois types d'erreurs sont spécifiés : *CameraError* - un type d'erreur

générique pour indiquer une erreur de l'appareil photo ; *LensDirty* qui est un type de CameraError ; *FocusError* qui est un autre type de *CameraError* spécifique à la fonction *Autofocus*. Les types d'erreurs sont définis dans un package AADL.

3.3 Conclusion du chapitre

Ce chapitre a présenté la méthodologie de l'étude de la coordination de groupe de robots sous-marin basé sur la méthode d'intégration orientée objet. Nous avons introduit une boucle de recherche en 3 cycles.

— cycle 1 : Recherche avec des équations dynamiques linéaires.
— cycle 2 : Étude des équations dynamiques non linéaires.
— cycle 3 : Déploiement d'algorithmes sur le modèle de robot sous-marin Open-Source et low-cost.

Pour chaque cycle de recherche, il y a 4 étapes : Analyse, Conception, Mise en œuvre et Simulation / Tests. Avec cette approche, on peut facilement hériter, réutiliser et modifier les modules étudiés dans les cycles précédents. De plus, nous présentons également un cadre de recherche composé de trois modèles : Modèle de comportement, Modèle d'architecture et Modèle Robots. La transformation entre les modèles peut être effectué partiellement automatiquement. Cela rend les 4 étapes de recherche ci-dessus peuvent être effectuées rapidement et intuitivement.

L'approche AADL de l'architecture logicielle permet une analyse préliminaire de la structure macroscopique des futurs algorithmes à écrire, et également de l'influence possible de défauts et de délais sur le comportement.

4

Modèle du robot sous-marin

Comme pour les autres véhicules marins, la dynamique des Véhicules Sous-Marins (dans ce cas est UUV - **U**nmanned **U**nderwater **V**ehicles) est non linéaire, très couplée en mouvement et sensible aux incertitudes hydrodynamiques. Les hypothèses suivantes peuvent être formulées lors du calcul de l'équation dynamique générale du UUV afin de simplifier l'effort de modélisation. Il s'agit notamment de : un UUV est un corps rigide qui est entièrement immergé une fois dans l'eau ; l'eau est supposée être le fluide idéal, incompressible, non visqueux (sans friction) ; un UUV se déplace lentement pour des opérations telles que l'inspection d'un pipeline ; le repère fixé à la terre est inertiel ; les perturbations dues aux vagues sont négligées car elles sont complètement submergées ; La dynamique de l'ancrage n'est pas modélisée (dans le cas de ROV - **R**emotely **O**perated **V**ehicle). Les notations standard de la ***S**ociety of **N**aval **A**rchitects & **M**arine **E**ngineers* (**SNAME**) utilisées pour le véhicule maritime sont indiquées au tableau 4.1.

Tableau 4.1 – Les notations standards sont utilisées pour le véhicule maritime.

DOF	Motions	Force et Couple	Vitesse linéaire et angulaire	Position et angles de Euler
1	Surge	X	u	x
2	Sway	Y	v	y
3	Heave	Z	w	z
4	Roll	K	p	ϕ
5	Pitch	M	q	θ
6	Yaw	N	r	ψ

4.1 La cinématique du robot sous-marin

La transformation utilisant les angles d'Euler fournit une transformation importante entre la dynamique exprimée dans le repère du robot (désigné par $\{b\}$ ou on a $O_bX_bY_bZ_b$) et le repère inertiel absolu (désigné par $\{n\}$) telle que vue dans la Figure 4.1. Comme les accélérations d'un point à la surface de la terre peuvent être négligées pour les robots sous-marins lents, le repère fixé à la terre peut être considéré comme un repère inertiel. Les équations cinématiques qui représentent la transformation d'Euler peuvent être écrites comme :

$$\dot{\eta} = J_\Theta(\eta)\,\mathbf{v} \tag{4.1}$$

où $\eta = \left[\mathbf{p}^n_{b/n}, \Theta\right]^\top$ avec $\mathbf{p}^n_{b/n} = (x, y, z)$ donnant les coordonnées du centre du robot sous-marin exprimées dans le repère inertiel absolu (désigné par n) et $\Theta = (\phi, \theta, \psi)$ donnant les trois angles d'Euler.
$\mathbf{v} = \left[\nu^b_{b/n}, \omega^b_{b/n}\right]^\top$ avec $\nu^b_{b/n} = (u, v, w)$ est la vitesse linéaire du robot sous-marin dans son propre repère ; $\omega^b_{b/n} = (p, q, r)$ est la vitesse angulaire du robot sous-marin dans son propre repère.

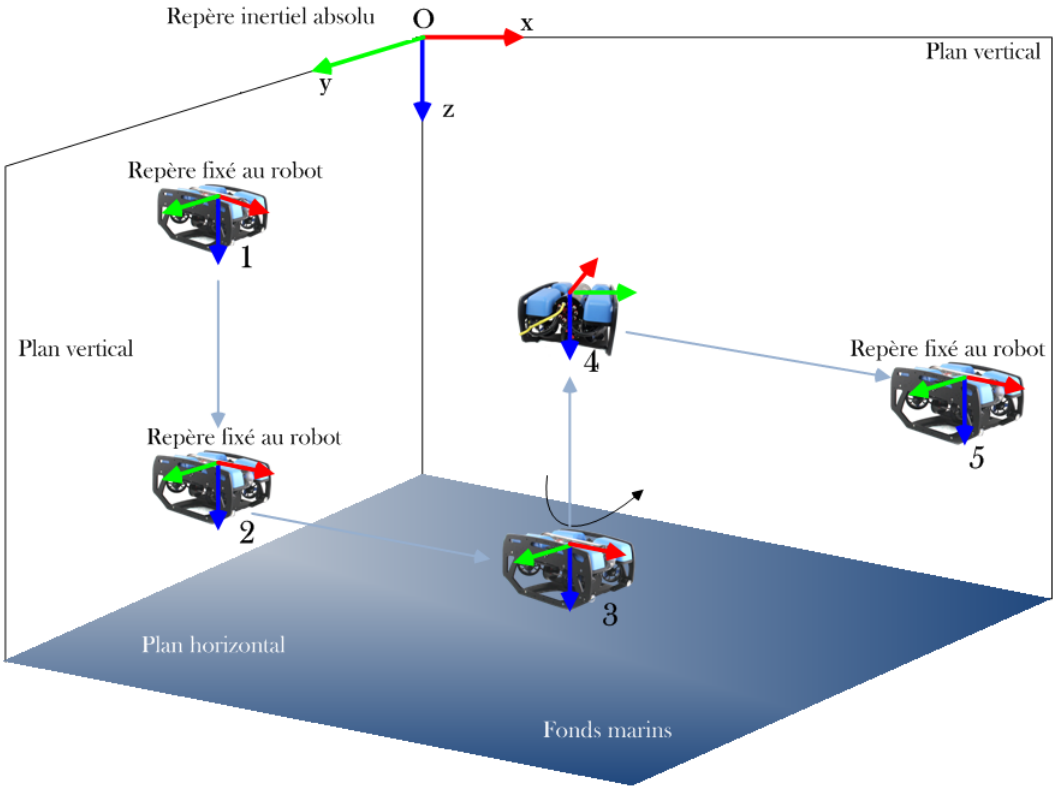


Figure 4.1 – Proposition d'un suivi sous-marin de pipeline pour les UUVs

$$\Rightarrow \begin{bmatrix} \dot{\mathbf{p}}_{b/n}^{n} \\ \dot{\Theta} \end{bmatrix} = \begin{bmatrix} R_b^n(\Theta) & 0_{3\times3} \\ 0_{3\times3} & T_\Theta(\Theta) \end{bmatrix} \begin{bmatrix} \nu_{b/n}^{b} \\ \omega_{b/n}^{b} \end{bmatrix} \tag{4.2}$$

où

$$T_\Theta(\Theta) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix} \tag{4.3}$$

$$R_b^n(\Theta) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{4.4}$$

avec $c = cos$ et $s = sin$

4.2 La cinétique du robot sous-marin

En se basant sur la deuxième loi du mouvement de Newton, qui relie masse m, accélération $\dot{\mathbf{v}}$ et force. La cinétique du corps rigide peut être exprimée sous forme matricielle (selon [112]) :

$$M_{RB}\dot{\mathbf{v}} + C_{RB}(\mathbf{v})\,\mathbf{v} = \tau_{RB} \tag{4.5}$$

où $M_{RB} \in \mathbb{R}^{6\times6}$ est la matrice masse-inertie ;
$C_{RB} \in \mathbb{R}^{6\times6}$ est la matrice des forces de Coriolis ;
$\mathbf{v} = [u, v, w, p, q, r]^\top$ est le vecteur qui représente vitesses linéaires et angulaires dans le repère fixé au robot sous marin ;
$\tau_{RB} = [X, Y, Z, K, M, N]^\top$ est le vecteur des forces et moments qui s'appliquent sur le robot sous-marin.

Les matrices M_{RB}, C_{RB} et τ_{RB} sont présentés ci-dessous.

---

i) Selon [112], on a **la matrice masse-inertie M_{RB} comme suivante** :

$$M_{RB} = \begin{bmatrix} mI_{3\times3} & -mS\left(r_g^b\right) \\ mS\left(r_g^b\right) & I_g \end{bmatrix}$$

avec m est la masse du robot sous-marin ; $r_g^b = [x_g, y_g, z_g]^\top$ est le vecteur de O_b à **C**entre de **G**ravité (**CG**) exprimé en $\{b\}$; I_g est la matrice d'inertie sur le (**CG**) du robot sous-marin. Noter que : La matrice I_g est calculée sur le modèle CAD (en anglais *Computer Aided Design*). $S(\bullet)$ est défini comme suit :

$$S(\lambda) = -S^\top(\lambda) = \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix}, \quad \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \tag{4.6}$$

On peut obtenir

$$M_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x & -I_{xy} & -I_{xz} \\ mz_g & 0 & -mx_g & -I_{yx} & I_y & -I_{yz} \\ -my_g & mx_g & 0 & -I_{zx} & -I_{zy} & I_z \end{bmatrix} \tag{4.7}$$

I_x, I_y, I_z sont les moments d'inertie autour de X_b, Y_b, Z_b ; $I_{xy} = I_{yx}, I_{xz} = I_{zx}, I_{yz} = I_{zy}$.

Hypothèse 1 (*La matrice masse-inertie*). *Dans le cas du point O_b (l'origine du cadre de référence fixé sur le robot) qui coïncide avec le **C**entre de **G**ravité (**CG**) du robot sous-marin. Par conséquent, on a $r_g^b = [0, 0, 0]^\top$; $I_{xy} = I_{yx} = I_{xz} = I_{zx} = I_{yz} = I_{zy} = 0$.*

La matrice M_{RB} devient :

$$M_{RB} = \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_g \end{bmatrix} \tag{4.8}$$

---

ii) Selon [112], avec l'hypothèse des paramètres indépendants de la vitesse, on peut obtenir **la matrice des forces de Coriolis** $C_{RB}(\mathbf{v})$ suivante :

$$C_{RB}(\mathbf{v}) = \begin{bmatrix} mS(\mathbf{v_2}) & -mS(\mathbf{v_2})\, S\left(r_g^b\right) \\ mS\left(r_g^b\right) S(\mathbf{v_2}) & -S(I_b\mathbf{v_2}) \end{bmatrix} \tag{4.9}$$

où $\mathbf{v_2} = [p, q, r]^\top$; $S(\bullet)$ est défini comme l'équation 4.6 ; I_b est la matrice d'inertie sur le point O_b

$$I_b = I_g - mS^2\left(r_g^b\right) \tag{4.10}$$

$$\Rightarrow C_{RB}(\mathbf{v})\,\mathbf{v} = \begin{bmatrix} mS(\mathbf{v_2}) & -mS(\mathbf{v_2})\, S\left(r_g^b\right) \\ mS\left(r_g^b\right) S(\mathbf{v_2}) & -S(I_b\mathbf{v_2}) \end{bmatrix} \mathbf{v}$$

Avec l'hypothèse 1, on peut obtenir

$$C_{RB}(\mathbf{v})\,\mathbf{v} = \begin{bmatrix} 0 & -mr & mq & 0 & 0 & 0 \\ mr & 0 & -mp & 0 & 0 & 0 \\ -mq & mp & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & rI_z & -qI_y \\ 0 & 0 & 0 & -rI_z & 0 & pI_x \\ 0 & 0 & 0 & qI_y & -pI_x & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \tag{4.11}$$

$$= \begin{bmatrix} m(qw - rv) \\ m(ru - pw) \\ m(pv - qu) \\ qr(I_z - I_y) \\ rp(I_x - I_z) \\ qp(I_y - I_x) \end{bmatrix} \tag{4.12}$$

iii) De plus, **le vecteur de force et de moment externe** τ_{RB} comprend les forces et moments hydrodynamiques τ_{Dyn}, hydrostatiques τ_{Stat} dus à l'amortissement et à l'inertie du fluide environnant, ainsi que la force et le moment de contrôle τ_{Act} (la force et le moment de l'environnement comme le vent, les vagues sont négligés $\tau_{Dist} \approx 0$)

$$\tau_{RB} = \tau_{Dyn} + \tau_{Stat} + \tau_{Act} + \tau_{Dist} \tag{4.13}$$

avec

$$\tau_{Dyn} = -M_A \dot{\mathbf{v}} - C_A(\mathbf{v})\mathbf{v} - D(\mathbf{v})\mathbf{v} \tag{4.14}$$

Dans cet équation M_A qui est la matrice de masse ajoutée hydrodynamique s'écrit comme suit :

$$M_A = -\begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix} \tag{4.15}$$

Pour les applications AUV et ROV, les robots pourront se déplacer à basse vitesse. Cela suggère que la contribution des éléments hors diagonale de la matrice M_A peut être négligée, donc :

$$M_A = -\begin{bmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{\dot{w}} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{\dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{\dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{\dot{r}} \end{bmatrix} \tag{4.16}$$

où $X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}$ est la masse ajoutée en x, y, z direction due à la translation dans l'eau. $K_{\dot{p}}, M_{\dot{q}}, N_{\dot{r}}$ est l'augmentation de l'inertie autour de x, y, z-axe due à la rotation dans l'eau.

$C_A(\mathbf{v})$ est la matrice hydrodynamique de Coriolis et centripète ajoutée :

$$C_A(\mathbf{v}) = \begin{bmatrix} 0 & 0 & 0 & 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v \\ 0 & 0 & 0 & Z_{\dot{w}}w & 0 & -X_{\dot{u}}u \\ 0 & 0 & 0 & -Y_{\dot{v}}v & X_{\dot{u}}u & 0 \\ 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v & 0 & -N_{\dot{r}}r & M_{\dot{q}}q \\ Z_{\dot{w}}w & 0 & -X_{\dot{u}}u & N_{\dot{r}}r & 0 & -K_{\dot{p}}p \\ -Y_{\dot{v}}v & X_{\dot{u}}u & 0 & -M_{\dot{q}}q & K_{\dot{p}}p & 0 \end{bmatrix} \tag{4.17}$$

$D(\mathbf{v})$ est la matrice d'amortissement qui contient les termes d'amortissement linéaire D et les termes d'amortissement quadratique ou non linéaire $D_n(\mathbf{v})$ comme suit :

$$D(\mathbf{v}) = D + D_n(\mathbf{v}) \tag{4.18}$$

Hypothèse 2 (*Matrice d'amortissement hydrodynamique*). *L'UUV fonctionne à une vitesse linéaire de 1 m/s (ou une vitesse angulaire de 0,5 rad/s), il se trouve bien à l'intérieur de la zone d'amortissement linéaire de 2 m/s maximum (voir la zone circulaire à la figure 4.2). Par conséquent, l'amortissement hydrodynamique linéaire a été pris en compte.*

Hypothèse 3 (*Matrice d'amortissement hydrodynamique*). *Les éléments hors diagonale en $D(\mathbf{v})$ sur un véhicule sous-marin sont petits par rapport aux éléments diagonaux.*

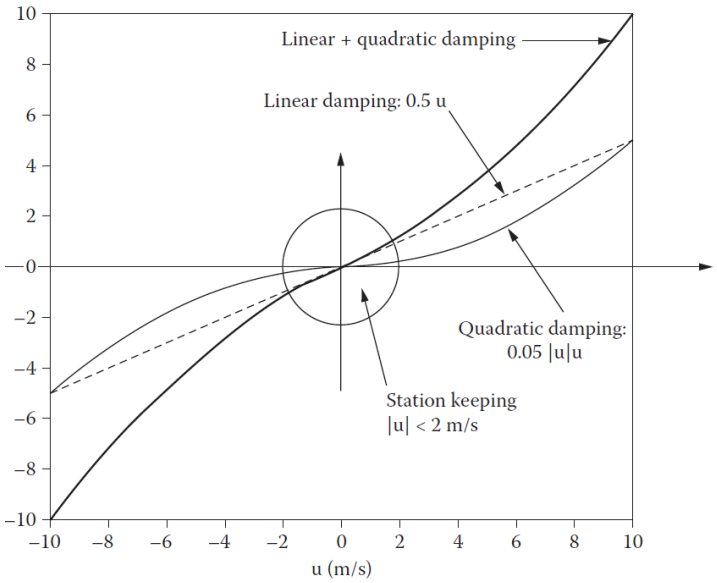


Figure 4.2 – Amortissement hydrodynamique à basse et haute vitesse du véhicule. (Photo de T. I. Fossen [113]).

L'équation 4.18 devient :

$$D(\mathbf{v}) = -\begin{bmatrix} X_u & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_w & 0 & 0 & 0 \\ 0 & 0 & 0 & K_p & 0 & 0 \\ 0 & 0 & 0 & 0 & M_q & 0 \\ 0 & 0 & 0 & 0 & 0 & N_r \end{bmatrix} \tag{4.19}$$

$\tau_{Stat} = -g(\eta)$ représente les effets hydrostatiques. En cela :

$$g(\eta) = \begin{bmatrix} (W-B)\sin\theta \\ -(W-B)\cos\theta\sin\phi \\ -(W-B)\cos\theta\cos\phi \\ -z_B B\cos\theta\sin\phi \\ -z_B B\sin\theta \\ 0 \end{bmatrix} \tag{4.20}$$

où $\eta = [x, y, z, \phi, \theta, \psi]^\top$ est la position et angles Euler ; $B = \rho g V$ est la force de flottabilité ; $W = mg$ est le poids immergé du robot sous-marin ; ρ est la densité de l'eau ; g est l'accélération de la gravité ; V est le volume de fluide déplacé par le robot. z_B est la distance de **CG** (centre de gravité) à **CB** (centre de flottabilité).

Hypothèse 4 (*Matrice de flottabilité et de force gravitationnelle*). *Lors de la conception d'un UUV, il est souhaitable de rendre le ROV flottant de façon neutre ou légèrement positif en ajoutant un flotteur ou une masse d'équilibrage supplémentaire. Avec cela, le UUV devient flottant de façon neutre, $W = B$.*

Hypothèse 5 (*Matrice de flottabilité et de force gravitationnelle*). *En plaçant une masse supplémentaire sur le ROV pour faire coïncider les coordonnées $X - Y$ du **CB** avec la coordonnée $X - Y$ du **CG**, c'est-à-dire $x_G = x_B = 0, y_G = y_B = 0$.*

L'équation 4.20 devient :

$$g(\eta) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -z_B B\cos\theta\sin\phi \\ -z_B B\sin\theta \\ 0 \end{bmatrix} \tag{4.21}$$

$\tau_{Act} = Tf(u)$ est la matrice de la force et du moment de contrôle. Où T est la matrice décrivant la géométrie des servomoteurs.

Donc, de l'équation 4.5 et l'équation 4.13, on a l'équation dynamique du robot sous-marin :

$$\begin{aligned} M_{RB}\dot{\mathbf{v}} + C_{RB}(\mathbf{v})\mathbf{v} &= \tau_{RB} \\ \Leftrightarrow M_{RB}\dot{\mathbf{v}} + C_{RB}(\mathbf{v})\mathbf{v} &= \tau_{Dyn} + \tau_{Stat} + \tau_{Act} \\ \Leftrightarrow M_{RB}\dot{\mathbf{v}} + C_{RB}(\mathbf{v})\mathbf{v} + M_A\dot{\mathbf{v}} + C_A(\mathbf{v})\mathbf{v} + D(\mathbf{v})\mathbf{v} + g(\eta) &= \tau_{Act} \\ \Leftrightarrow M\dot{\mathbf{v}} + C(\mathbf{v})\mathbf{v} + D(\mathbf{v})\mathbf{v} + g(\eta) &= \tau_{Act} \end{aligned} \tag{4.22}$$

où $M = M_{RB} + M_A$ et $C(\mathbf{v}) = C_{RB}(\mathbf{v}) + C_A(\mathbf{v})$

4.3 Modèle du robot sous-marin dans un plan horizontal

4.3.1 3 degrés de liberté (3 DOF)

Afin de réduire le couplage de mouvement, il est souvent utilisé un schéma de commande de manœuvre découplé. Un robot sous-marin peut se déplacer soit dans un plan horizontal, soit dans un plan vertical (voir la Figure 4.1). Cela réduit le nombre de degrés de liberté contrôlables impliqués dans la conception des systèmes de contrôle. Donc, les mouvements du robot sous-marin peuvent être découplés en deux sous-systèmes simples tels que le :

— Sous-système de plan horizontal : la vitesse $\mathbf{v}_h = [u, v, r]^\top$ et la position $\eta_h = [x, y, \psi]^\top$.

— Sous-système de plan vertical : la vitesse $\mathbf{v}_v = [u, w, q]^\top$ et la position $\eta_v = [x, z, \theta]^\top$.

Dans le cadre de recherche de la thèse, nous nous concentrerons sur le mouvement horizontal. Pour cette raison, les équations cinématiques du mouvement se réduisent de l'expression générale 6 DOF (4.2) à une rotation principale autour de l'axe z (équivalent à 3DOF).

$$J_\Theta(\eta) \Rightarrow R(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.23}$$

et l'équation (4.2), nous avons les équations cinématiques dans 3 DOF devient :

$$\dot{\eta}_h = R(\psi)\mathbf{v}_h \tag{4.24}$$

$$\Leftrightarrow \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{4.25}$$

$$\Leftrightarrow \begin{cases} \dot{x} = u\cos(\psi) - v\sin(\psi) \\ \dot{y} = u\sin(\psi) + v\cos(\psi) \\ \dot{\psi} = r \end{cases} \tag{4.26}$$

4.3.2 Espace d'état du robot sous marin

Le modèle du robot sous-marin (4.24)-(4.22) est non linéaire en raison des transformations cinématiques. Ce modèle peut être linéarisé de manière dynamique en introduisant les coordonnées parallèles du robot [112], qui sont définies dans un repère fixé au robot avec des axes parallèles au repère inertiel absolu (désigné par p).

$$\eta_p = R^{\top}(\psi)\,\eta_h \tag{4.27}$$

où η_p se compose de la position (x_p, y_p) et cap ψ du robot sous-marin exprimée en coordonnées parallèles du robot, $\eta_p = [x_p, y_p, \psi]^{\top}$.

En utilisant une hypothèse de basse vitesse $\dot{\psi} \simeq 0$, selon [114], il s'ensuit que le modèle du robot sous-marin (4.24) et (4.22) peut être exprimé par

$$\dot{\eta}_p = \mathbf{v}_h \tag{4.28}$$

$$M_h\dot{\mathbf{v}}_{\mathbf{h}} + D_h(\mathbf{v}_h)\,\mathbf{v}_h + g_h(\mathbf{v}_h) - R(\psi)\,b = \tau_h \tag{4.29}$$

où $b \in \mathbb{R}^3$ désigne une perturbation extérieure induite par le vent, les vagues et les courants océaniques ; $g_h(\mathbf{v}_h) \in \mathbb{R}^3$ représente l'hydrodynamique non modélisée ; $M_h \in \mathbb{R}^3$; $D_h \in \mathbb{R}^3$

On peut modeler les équation 4.28 et 4.29 dans un modèle d'espace d'états de la forme :

$$\dot{\mathbf{x}} = A\mathbf{x} + B\left[\tau_h + \mathbf{f}(\mathbf{x}) + \mathbf{w}(t)\right] \tag{4.30}$$

où $\mathbf{x} \in \mathbb{R}^6$ est le vecteur d'état du robot ;
$\mathbf{f}(\mathbf{x}) \in \mathbb{R}^3$ est l'incertitude inconnue du robot ;
$\mathbf{w}(t) \in \mathbb{R}^3$ représente la perturbation inconnue correspondante ;
$A \in \mathbb{R}^{6\times6}$ et $B \in \mathbb{R}^{6\times3}$ sont des matrices connues, et (A, B) est supposé stabilisable.

$$A = \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix}, \qquad B = \begin{bmatrix} 0_{3\times3} \\ M_h^{-1} \end{bmatrix} \tag{4.31}$$

$$\mathbf{f}(\mathbf{x}) = -D_h(\mathbf{v}_h)\mathbf{v}_h - g_h(\mathbf{v}_h), \qquad \mathbf{w}(t) = R(\psi)\,b \tag{4.32}$$

4.4 Modélisation du sous-marin BlueROV-1

4.4.1 BlueROV - Open plate-forme physique

Le BlueROV-1 est la plate-forme open source et économique utilisée. La figure 4.3 montre les composants de BlueROV-1 avec le contrôle d'architecture. Le cadre et les propulseurs BlueROV sont inclus dans un ensemble de Blue Robotics. En plus du cadre BlueROV, un Raspberry Pi a été utilisé comme ordinateur de bord (contrôle de haut niveau) et un HKPilot Mega 2.7 a été utilisé comme unité entrée/sortie (contrôle de bas niveau). Le BlueROV I/O se compose d'un HKPilot Mega 2.7 qui est basé sur Ardupilot Mega. Le HKPilot Mega 2.7 possède les capteurs suivants sur puce : Magnétomètre - HMC5883L, Baromètre - MS5611-01BA, Unité de mesure inertielle (IMU) - MPU6000. Un capteur de pression externe MS5837-30BA qui est enfermé dans un boîtier étanche par Blue Robotics a été connecté au HKPilot Mega 2.7 par I2C. Le HKPilot Mega 2.7 contrôle également les six ESC (en anglais *Electronic Speed Controller*).

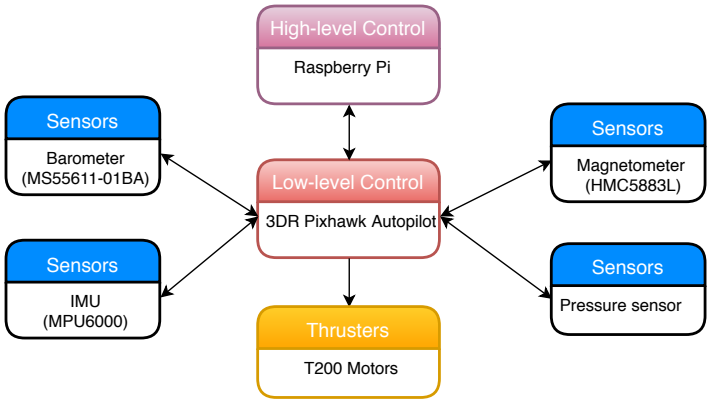


Figure 4.3 – Structure physique du BlueROV-1

4.4.2 Modélisation du sous-marin BlueROV-1 sous ROS/Gazebo

Pour simuler un BlueROV dans un simulateur de robot, tel que Gazebo [1], V-REP, etc., nous devons définir les propriétés physiques de la liaison robot telles que la géométrie, la couleur, la masse et l'inertie, et les propriétés de collision de la liaison. Nous définissons ensuite toutes ces propriétés à l'intérieur du modèle du robot. Le format **U**niversel **R**obotic **D**escription **F**format (URDF) est un format de fichier XML utilisé dans ROS et Gazebo pour décrire tous les éléments du robot. De plus, selon Manhães et al [115], le simulateur de véhicule sous-marin sans pilote qui est une extension du simulateur de robotique

1. Gazebo est un simulateur 3D, cinématique, dynamique et multi-robot permettant de simuler des robots articulés dans des environnements complexes, intérieurs ou extérieurs, réalistes et en trois dimensions

open-source Gazebo aux scénarios sous-marins peut être simulé par plusieurs robots sous-marins. La figure 4.4 montre les composants utilisés pour modéliser le BlueROV-1 qui ont été développés sur ROS/Gazebo avec URDF. Il est composé de 7 blocs :

— **(I) :** Modèle de cadre BlueROV
— **(II) :** Modèle de propulseur
— **(III) :** Modèle de capteurs
— **(IV) :** Effets hydrostatiques et hydrodynamiques
— **(V) :** Étendre : Perturbations et modèle de communication
— **(VI) :** Équation de la dynamique de l'AUV
— **(VII) :** Modèle de simulation BlueROV-1

Le modèle de BlueROV-1 sous ROS/Gazebo sera utilisé ensuite pour la simulation de modèle PSM (en anglais *Platform Specific Model*) dans le chapitre 6.

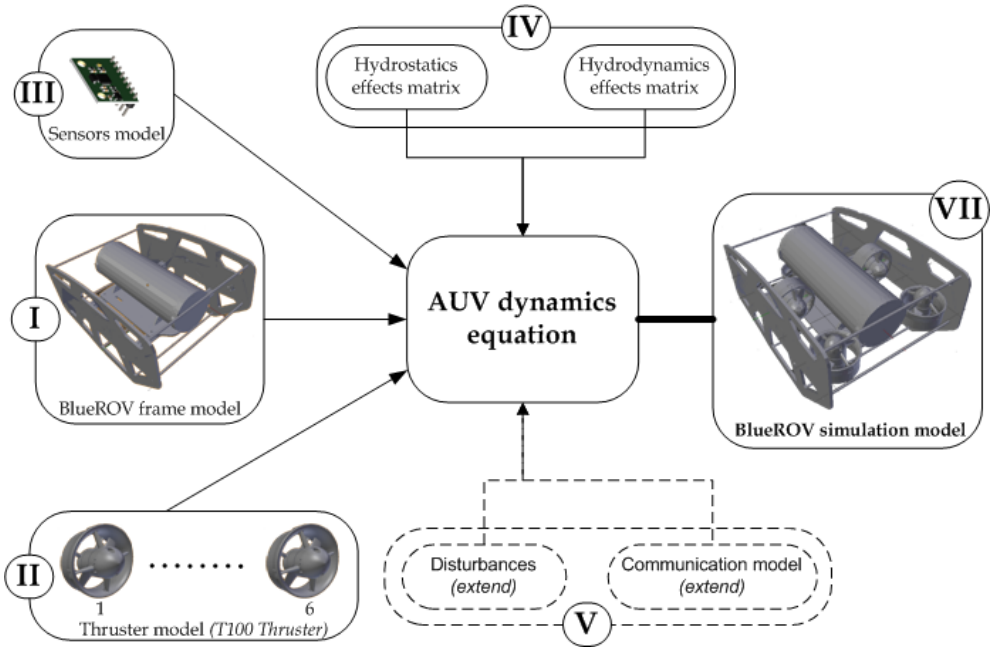


Figure 4.4 – BlueROV-1 avec ses modèles constitutifs

Tableau 4.2 – La table de référence des valeurs du robot sous-marin BlueROV [116], [117]

Notation	Valeur	Description	Type
m	6.621 kg	La masse du ROV.	Mesuré
g	9.82 m/s^2	Constante gravitationnelle.	–
ρ	1000 kg/m^3	Densité de l'eau.	–
l_{x1}	0.19 m	Distance entre le propulseur 1 et le CG dans la direction x	Mesuré
l_{y1}	0.11 m	Distance entre le propulseur 1 et le CG dans la direction y	Mesuré
l_{y2}	0.11 m	Distance entre le propulseur 2 et le CG dans la direction y	Mesuré
l_{x2}	0.19 m	Distance entre le propulseur 2 et le CG dans la direction x	Mesuré
l_{y3}	0.11 m	Distance entre le propulseur 3 et le CG dans la direction y	Mesuré
l_{x5}	0.17 m	Distance entre le propulseur 5 et le CG dans la direction x	Mesuré
l_{y4}	0.11 m	Distance entre le propulseur 4 et le CG dans la direction y	Mesuré
l_{z6}	0 m	Distance entre le propulseur 6 et le CG dans la direction z	Mesuré
z_B	-0.0420 m	Distance entre CG et CB.	Mesuré
K_p	-0.8842 $kg.m^2$	Coefficient d'amortissement linéaire dû à la rotation dans eau autour de l'axe des x.	Estimé
$K_{p\|p\|}$	-0.6682 $kg.m^2$	Coefficient d'amortissement quadratique dû à la translation dans la direction x	Estimé
M_q	-0.8547 $kg.m^2$	Coefficient d'amortissement linéaire dû à la rotation dans eau autour de l'axe des y.	Estimé
$M_{q\|q\|}$	-0.3354 $kg.m^2$	Coefficient d'amortissement quadratique dû à la translation dans la direction y	Estimé
N_r	-1.0280 $kg.m^2$	Coefficient d'amortissement linéaire dû à la rotation dans eau autour de l'axe des z.	Estimé
$N_{r\|r\|}$	-1.0249 $kg.m^2$	Coefficient d'amortissement quadratique dû à la translation dans la direction z	Estimé
A_p	0.8337 $kg.m^2$	Inertie et inertie augmentée autour de l'axe x	Estimé
B_q	0.7987 $kg.m^2$	Inertie et inertie augmentée autour de l'axe y	Estimé
C_r	1.1250 $kg.m^2$	Inertie et inertie augmentée autour de l'axe z	Estimé

4.5 Conclusion du chapitre

Ce chapitre a présenté et analysé le modèle cinétique du robot sous-marin. En commençant par l'équation de Fossen, nous analysons ensuite indépendamment le mouvement UUV dans les directions verticale et horizontale. Nous nous concentrons sur l'étude du modèle de mouvement UUV dans la direction horizontale (c'est-à-dire en omettant le mouvement de l'angle de roulis et de tangage), et avec l'hypothèse que l'origine de l'UUV coïncide avec le centre de gravité, on obtient l'équation cinématique de l'UUV. Ensuite, nous utilisons cette équation pour étudier les algorithmes de contrôle coordonné du groupe robot sous-marin (qui seront présentés au chapitre 5, et ce chapitre correspond également à la phase 1 de la méthodologie itérative d'ingénierie de systèmes de la figure 3.1).

Ensuite, nous représenterons l'équation cinétique non linéaire de l'UUV sous la forme d'un modèle d'espace d'états de la forme en supposant que la vitesse angulaire de l'UUV est très petite (approximativement nulle). Avec la création de l'équation d'espace d'états, nous continuons à étudier les algorithmes de contrôle coordonné du groupe de robots sous-marins (cette étude sera présentée au chapitre 6 - correspondant à la phase 2 par Méthodologie itérative d'ingénierie de systèmes de la figure 3.1).

Enfin, nous utiliserons une plateforme de robot sous-marins open-source et low-cost pour tester et valider les algorithmes que nous avons étudiés (cette partie de l'étude sera introduite au chapitre 7 - correspondant à la phase 3 par Méthodologie itérative d'ingénierie de systèmes de la figure 3.1).

5

Coordination pour modèles linéaires de sous-marin

5.1 Introduction de la théorie de graphe pour le contrôle de la formation

Il est naturel de modéliser l'échange d'informations (cela peut être compris comme les relations de voisinage) entre les véhicules à l'aide de graphiques orientés/non-orientés. Considérons un système de n véhicules connectés, sa topologie réseau peut être modélisée sous la forme d'un graphique non-orienté dénoté $\mathcal{G} = (\mathcal{V}, \mathcal{W})$, où $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ et $\mathcal{W} \subseteq \mathcal{V} \times \mathcal{V}$ sont respectivement le groupe de véhicules et le groupe d'arêtes qui relient les véhicules l'un à l'autre. Deux matrices sont fréquemment utilisées pour représenter la topologie du réseau :

La matrice d'adjacence $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$

$$\mathcal{A} = \begin{cases} a_{ij} > 0 & \text{if } (v_j, v_i) \in \mathcal{W} \\ a_{ij} = 0 & \text{dans le cas contraire} \end{cases} \tag{5.1}$$

La matrice de Laplacien $\mathcal{L} = [\ell_{ij}] \in \mathbb{R}^{n \times n}$ avec

$$\ell = \begin{cases} \ell_{ii} = \sum_{j=1}^{n} a_{ij} & \\ \ell_{ij} = -a_{ij} & \forall i \neq j \end{cases} \tag{5.2}$$

Le Laplacien $\mathcal{L}$ a au moins une seule valeur propre nulle avec un vecteur propre $\mathbf{1}$ correspondant composé de tous les nombres 1.

5.2 Coordination dans un groupe de sous-marins basée sur le modèle simple intégrateur

Cette section s'adresse à la coordination de groupes de sous-marins qui sont modélisés comme les modèles simple-intégrateurs. Plus précisément, les sous-marins vont se mettre en formation et suivre la trajectoire donnée. Pendant la mise en formation, les sous-marins pourront éviter les collisions. En outre, les sous-marins également évitent les collisions avec des obstacles (voir la Figure 5.1).

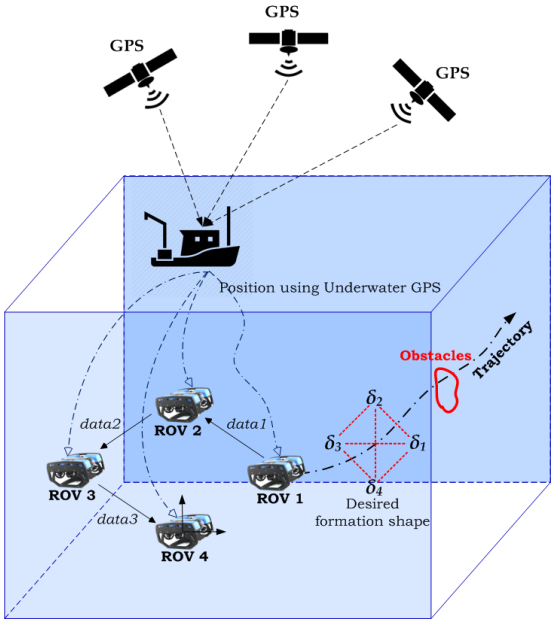


Figure 5.1 – Les scénarios d'étude

en supposant que :

— Les sous-marins peuvent se localiser dans le système de coordonnées global.
— Les sous-marins peuvent communiquer entre eux pour envoyer leur propre statut aux sous-marins restants par le graphe G.
— Les sous-marins se déplacent horizontalement.
— La forme de formation désirée est donnée comme $\delta^* = \left[(\delta_1)^\top ....(\delta_n)^\top\right]^\top \in \left(\mathbb{R}^d\right)^n$ dans le système de coordonnées global (voir la Figure 5.1). La trajectoire de référence est lisse.
— Pour éviter les collisions, nous avons supposé un capteur de proximité embarqué, qui a une portée de détection fixe avec le rayon, D_{out}.

Donc de l'équation cinématique 4.26 dans le chapitre 4 on obtient

$$\dot{x} = u\cos(\psi) - v\sin(\psi) \tag{5.3}$$
$$\dot{y} = u\sin(\psi) + v\cos\psi \tag{5.4}$$
$$\dot{\psi} = r \tag{5.5}$$

Hypothèse 6 *Supposons un robot sous-marin se déplaçant à une vitesse d'avance $U > 0$ telle que $u \approx U$ et $v \approx 0$.*

En utilisant l'hypothèse 6, les équations cinématiques du robot sous-marin peuvent être modélisées comme :

$$\dot{x} \approx u\cos(\psi) \tag{5.6}$$
$$\dot{y} \approx u\sin(\psi) \tag{5.7}$$
$$\dot{\psi} = r \tag{5.8}$$

Par conséquent, les équations cinématiques pour le robot sous-marin i dans un repère inertiel sont les suivantes :

$$\dot{x}_i = c_{u_i}\cos(\psi_i) \tag{5.9}$$
$$\dot{y}_i = c_{u_i}\sin(\psi_i) \tag{5.10}$$
$$\dot{\psi}_i = c_{r_i} \tag{5.11}$$

où $(x_i, y_i) \in \mathbb{R}^2$ sont les coordonnées cartésiennes du UUV i^{th}, $\psi_i \in [0, 2\pi)$ est l'orientation du UUV i^{th} par rapport à la référence inertielle et c_{u_i}, c_{r_i} sont les entrées de vitesse linéaire et angulaire, respectivement.

Donc, il est possible de linéariser par transfert d'un point fixe situé en dehors du centre du véhicule désigné par $\xi_i = (\xi_{xi}, \xi_{yi})^\top$ (comme indiqué dans la figure 5.2) :

$$\xi_{xi} = x_i + d_i\cos(\psi_i) \tag{5.12}$$

$$\xi_{yi} = y_i + d_i\sin(\psi_i) \tag{5.13}$$

Avec :

$$\begin{bmatrix} c_{u_i} \\ c_{r_i} \end{bmatrix} = \begin{bmatrix} \cos(\psi_i) & \sin(\psi_i) \\ -1/d_i\sin(\psi_i) & -1/d_i\cos(\psi_i) \end{bmatrix} \begin{bmatrix} \kappa_{xi} \\ \kappa_{yi} \end{bmatrix} \tag{5.14}$$

On a un modèle simple intégrateur comment suivant :

$$\begin{bmatrix} \dot{\xi}_{xi} \\ \dot{\xi}_{yi} \end{bmatrix} = \begin{bmatrix} \kappa_{xi} \\ \kappa_{yi} \end{bmatrix}, \quad o\grave{u} \quad i = 1, 2..., n \tag{5.15}$$

où $\kappa_i = [\kappa_{xi}, \kappa_{yi}]^\top$ est l'entrée de commande pour i^{th} UUV. La proposition de loi de contrôle

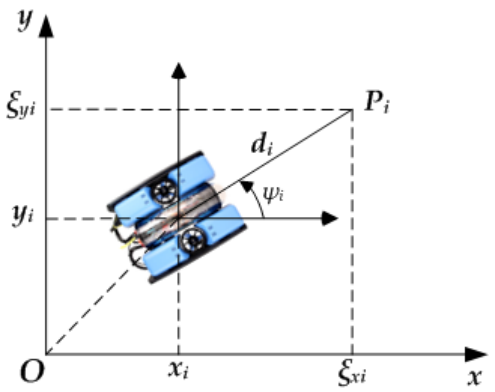


Figure 5.2 – Un point fixe situé à l'extérieur du centre de l'UUV

de formation, κ_i est divisée en deux parties : un terme consensuel qui est fait le suivi de formation, et un terme d'évitement de collision qui est fait l'évitement de collision. Cette loi de contrôle est représentée par :

$$\kappa_i = \kappa_i^{fo} + \kappa_i^{ca} \tag{5.16}$$

où $\kappa_i^{fo} = \left[\kappa_{xi}^{fo}, \kappa_{yi}^{fo} \right]^\top$ est la partie de mise en formation, et $\kappa_i^{ca} = \left[\kappa_{xi}^{ca}, \kappa_{yi}^{ca} \right]^\top$ est la partie d'évitement de collision.

5.2.1 Suivi de formation

En supposant que la topologie d'interaction des véhicules est fixe et connectée par un graphique dirigé ; il existe au moins un UUV qui peut accéder à l'information de trajectoire et quand un sous-ensemble de véhicules a accès à ξ^r, avec les algorithmes de consensus, on a : $\xi_i \to \xi^r + \delta_i$ et $\xi_i - \xi_j \to \delta_i - \delta_j$. Selon [71], un algorithme de consensus avec un état de référence variant dans le temps est donné comme suit :

$$\begin{aligned}
\kappa_i^{fo} &= \dot{\delta}_i + f\left(t, \xi^r\right) - \sum_{j=1}^{n} g_{ij} k_{ij} \left[(\xi_i - \xi_j) - (\delta_i - \delta_j)\right] - \alpha_i \left(\xi_i - \delta_i - \xi^r\right), \qquad i = \ell \\
\kappa_i^{fo} &= \dot{\delta}_i + \frac{1}{\sum_{j=1}^{n} g_{ij} k_{ij}} \sum_{j=1}^{n} g_{ij} k_{ij} \{\dot{\xi}_j - \dot{\delta}_j - \gamma_i \left[(\xi_i - \xi_j) - (\delta_i - \delta_j)\right]\}, \qquad i \neq \ell
\end{aligned} \tag{5.17}$$

où $\xi_i \in \mathbb{R}^2$ sont l'état du véhicule i^{th} ; $\delta_i - \delta_j, \forall i \neq j$, indique la formation souhaitée entre les états de UUV ; ξ^r est un état de référence variant dans le temps ; k_{ij} est un facteur de pondération positif, et g_{ij} vaut 1 si l'information du véhicule j au véhicule i et 0 sinon, $\forall i, j \in \{1, ..., n\}$; ℓ indique l'index du véhicule qui a accès à ξ^r ; $\alpha_{ij} > 0$.

En appliquant l'équation (5.17) pour contrôler le véhicule correspondant, la distance entre véhicules converge progressivement vers une valeur souhaitée $\|\delta_i - \delta_j\|$.

5.2.2 Formation avec évitement des collisions entre les UUVs

Considérons deux régions autour d'un véhicule, comme le montre la figure 5.3. Lorsqu'un véhicule se trouve à l'intérieur de la zone de prévention des collisions d'un autre véhicule, il est possible d'introduire une fonction $V(\xi)$ à potentiel lisse strictement décroissant pour éviter la collision.

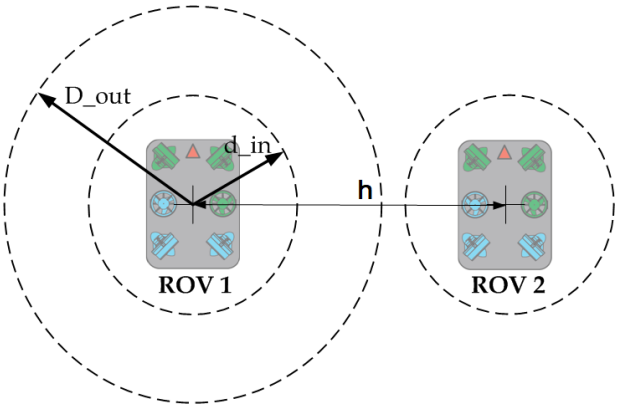


Figure 5.3 – Partition de deux régions d'un UUV pour éviter une collision

La fonction $g(h)$, qui aide à faire varier la fonction potentielle en douceur, est définie comme suit :

$$g(h) = \begin{cases} k_1 \text{ si } h \in (0, d_{in}), \\ k_2(h) \text{ si } h \in [d_{in}, D_{out}], \\ 0 \text{ si } h \notin (0, D_{out}). \end{cases} \tag{5.18}$$

où $k_1 = 1$; $h = \|\xi_i - \xi_j\|$ est la distance entre véhicules et

$$k_2(h) = \frac{1}{2}\left[1 + cos\left(\pi \frac{h - d_{in}}{D_{out} - d_{in}}\right)\right]$$

Pour générer un gradient basé sur la région d'évitement des collisions, une fonction de potentiel collectif lisse est définie comme suit :

$$V(\xi) = \frac{1}{2}\sum_i \sum_{j \in N_i} \psi_c(\|\xi_i - \xi_j\|) \tag{5.19}$$

où : N_i est un ensemble de robots contigus de robot j ; et

$$\psi_c(h) = \int_{D_{out}}^{h} \phi(s)\, ds$$

avec :

$$\phi(h) = \begin{cases} \dfrac{-h}{\sigma_{ij} + h^2} g(h) \text{ si } h \in (0, d_{in}) \\ \dfrac{-h}{(\sigma_{ij} + h^2)} g(h) \text{ si } h \in [d_{in}, D_{out}] \\ g(h) \text{ si } h \notin (0, D_{out}] \end{cases}$$

où σ_{ij} est un paramètre qui est ajusté en ligne par chaque robot sous-marin à ce moment pour éviter les collisions. Les détails de σ_{ij} peuvent être trouvés dans [107]. Enfin, le terme d'évitement des collisions κ_i^{ca} peut être écrit ainsi :

$$\kappa_i^{ca} = -\nabla_{p_i} V(\xi) = \sum_{j \in N_i} \phi(\|\xi_j - \xi_i\|) \frac{\xi_j - \xi_i}{\|\xi_j - \xi_i\|} \tag{5.20}$$

où $\nabla_{p_i} V(\xi)$ signifie le gradient d'une fonction $V(\xi)$, c'est-à-dire les dérivées partielles de $V(\xi)$ par rapport p_i.

5.2.3 Formation avec évitement d'obstacles

Afin d'éviter les collisions avec des obstacles lors du déplacement de plusieurs véhicules, nous avons utilisé l'approche [118] pour modifier la trajectoire en temps réel lorsqu'un des robots détecte un emplacement de l'obstacle. Il est basé sur la technique appelée la courbe Bézier qui est présentée comme suit :

$$P(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i \tag{5.21}$$

où P_i est le point de contrôle ; n est le degré de l'équation ; $\binom{n}{i}$ est le coefficient binomial ; $t \in [0, 1]$. Cependant, l'équation (5.21) donne l'équation d'une courbe Bézier qui commence à $t = 0$ et se termine à $t = 1$. Selon [118], il est possible d'autoriser un intervalle de paramètres arbitraire $t \in [t_0, t_1]$ de telle sorte que $P(t_0) = P_0$ et $P(t_1) = P_n$ puis la courbe Bézier définie sur un intervalle de paramètres arbitraire par $P_{[t_0,t_1]}(t)$ comme :

$$\begin{aligned} P_{[t_0,t_1]}(t) &= \frac{\sum_{i=0}^{n} \binom{n}{i} (t_1 - t)^{n-1} (t - t_0)^i P_i}{(t_1 - t_0)^n} \\ &= \sum_{i=0}^{n} \binom{n}{i} \left(\frac{t_1 - t}{t_1 - t_0}\right)^{n-i} \left(\frac{t - t_0}{t_1 - t_0}\right)^i P_i \end{aligned}$$

où n est le nombre de points de contrôle.

Une courbe lisse peut être définie en utilisant le point de contrôle $A0, A1, A2, A3$ comme

illustré dans la Figure 5.4. Pendant ce temps, le trajectoire de référence d'origine $f\left(t,\xi^{r}\right)$ est remplacée par la courbe de Bézier $P_{[t_0,t_1]}\left(t\right)$. Il faut noter que la courbe Bézier ne passe pas par les points intermédiaires $A2, A3$ en même temps qu'elle passe par le terminal $A0$ et $A3$. Les étapes détaillées de la mise en œuvre sont décrites sous la forme **Algorithme 1**.

Algorithm 1: Proposer de faire une trajectoire modifiée pour éviter les obstacles

Data: Distance entre l'UUV et l'obstacle d
Result: Trajectoire pour l'UUV
while $d <= R_{out}$ **do**
  Mettre à jour les distances entre les robots et les obstacles;
  Déterminer les points $A0, A1, A2, A3$;
  if $d <= r_{in}$ **then**
    Faire une trajectoire modifiée en utilisant la courbe Bézier ;
    UUVs suivent une trajectoire modifiée;
  else
    Revenir à une trajectoire initiale;
  end
end

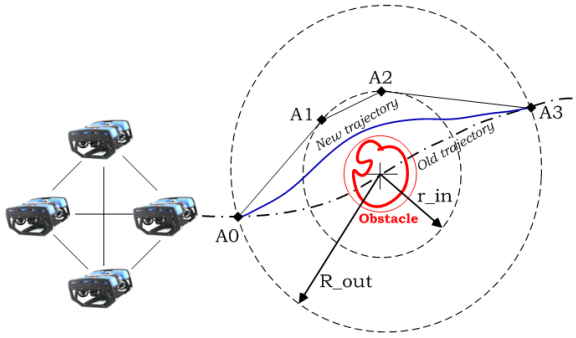


Figure 5.4 – Partition de deux régions d'une UUV pour éviter les obstacles

En utilisant la nouvelle trajectoire qui est créée par l'algorithme 1, un ensemble robots peut se déplacer et éviter automatiquement des obstacles en temps réel.

5.3 Résultats de la simulation

5.3.1 Dans Matlab

Nous avons fait la simulation dans Matlab pour vérifier les résultats théoriques et démontrer l'efficacité du suivi de formation et de l'obstacle d'évitement d'un modèle multi plate-forme indépendant. La configuration de formation souhaitée δ_i est prédéfinie par des coordonnées orthogonales telles que

$(\delta_{x1}, \delta_{y1}) = (0, 2), (\delta_{x2}, \delta_{y2}) = (-2, 0),$

$(\delta_{x3}, \delta_{y3}) = (0, -2)$, $(\delta_{x4}, \delta_{y4}) = (2, 0)$

et les états initiaux des véhicules sont les suivants

$(\xi_{x1}, \xi_{y1}) = (-2, 2)$, $(\xi_{x2}, \xi_{y2}) = (0, 3.5)$,

$(\xi_{x3}, \xi_{y3}) = (1, -4)$, $(\xi_{x4}, \xi_{y4}) = (3, 0)$.

Les paramètres de la simulation sont choisis comme suit : $\alpha_{ij} = 1$, $k_{ij} = 1$. La région extérieure $D_{out} = 2m$, la région intérieure $r_{in} = 2m$. Supposons que l'état de référence variant dans le temps soit donné comme suit

$$\xi^r = \left[30 \sin\left(\pi t/100\right), 20 \sin\left(\pi t/50\right)\right]^T \tag{5.22}$$

La position de l'obstacle est $(x_{ob}, y_{ob}) = (15, 15)$.

Les trajectoires de quatre véhicules et la formation finale suivant la trajectoire ont été montrées dans la Figure 5.5 et Figure 5.6.

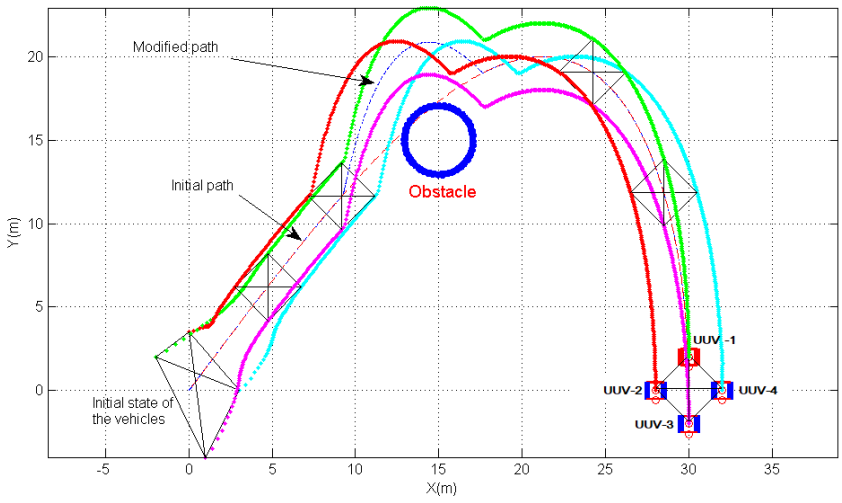


Figure 5.5 – Trajectoire de quatre robots sans évitement des collisions entre eux

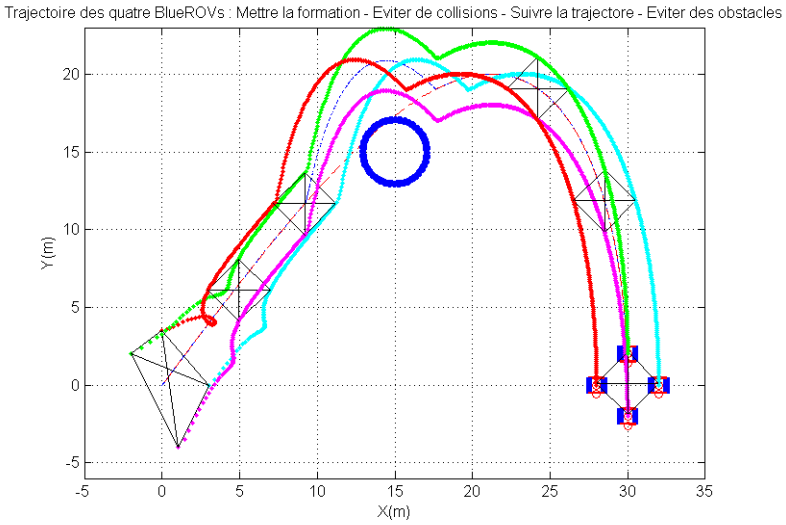


Figure 5.6 – Trajectoire de quatre robots avec l'évitement des collisions entre eux et l'évitement d'obstacle

La figure 5.7 et la figure 5.8 illustrent la distance minimale entre robots pour chaque robot. On peut observer que cette mesure de distance ne passe jamais en dessous de $d_{in} = 1m$ dans la Figure 5.8. Ceci garantit l'absence de collision entre les robots. On observe que tous les robots convergent vers un point d'équilibre constant, ce qui garantit l'obtention de la formation souhaitée avec quatre robots sous-marins.

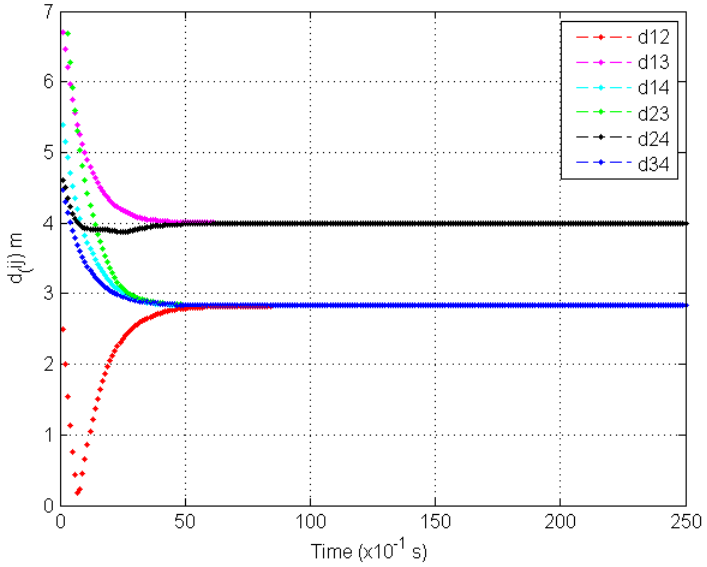


Figure 5.7 – Distances entre les robots sans l'évitement des collisions. (***Nous pouvons voir que la distance entre les robots 1 et 2*** (d_{12}) ***est inférieure à*** $1m$***. Cela conduit à une collision entre ces deux robots. La valeur*** $1m$ ***a été définie en tenant compte de la taille des robots.***)

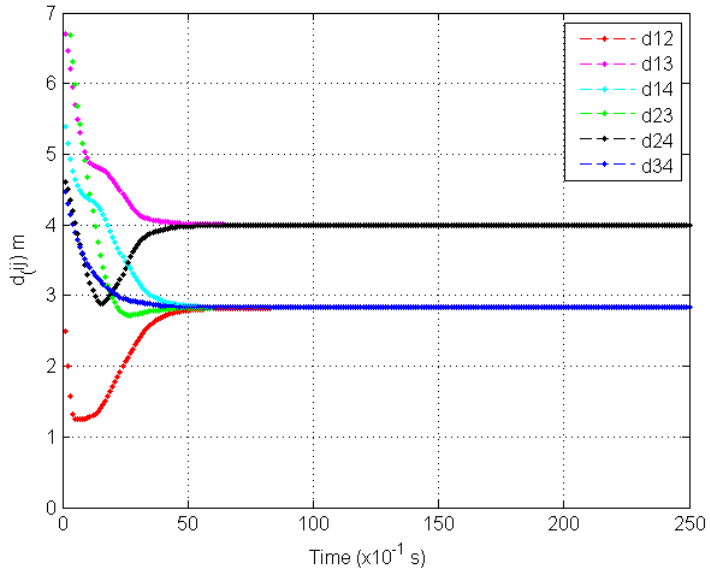


Figure 5.8 – Distances entre les robots avec évitement des collisions. (***Dans ce cas, nous pouvons voir que la distance entre les robots*** d_{ij} ***est toujours supérieure à 1. Cela garantit l'absence de collision entre les robots.***)

Enfin, la figure 5.9 montre les distances entre les robots sous marins et les obstacles qui ne descendent jamais sous $r_{in} = 2m$. Cette distance garantit qu'il n'y a pas de collision entre les robots sous-marins et les obstacles.

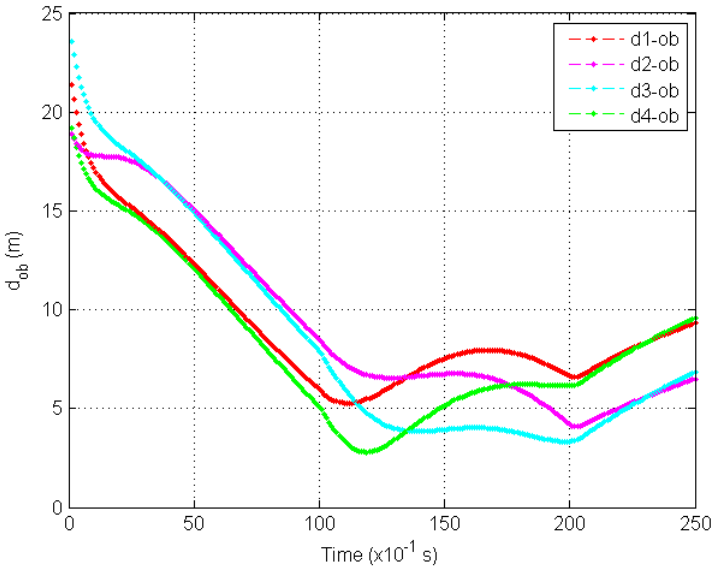


Figure 5.9 – Distances entre les robots et l'obstacle. (***En supposant que la position de l'obstacle est donnée, on constate que la distance entre les robots et l'obstacle*** (d_{i-ob}) ***est toujours supérieure à 2m (la valeur est prédéfinie et peut être ajustée). Cela garantit qu'il n'y a pas de collision entre les robots et les obstacles.***)

5.3.2 La combinaison entre Matlab/Simulink et ROS/Gazebo

La figure 5.10 présente le résultat de la relation entre MatlabSimulink et ROSGazebo.

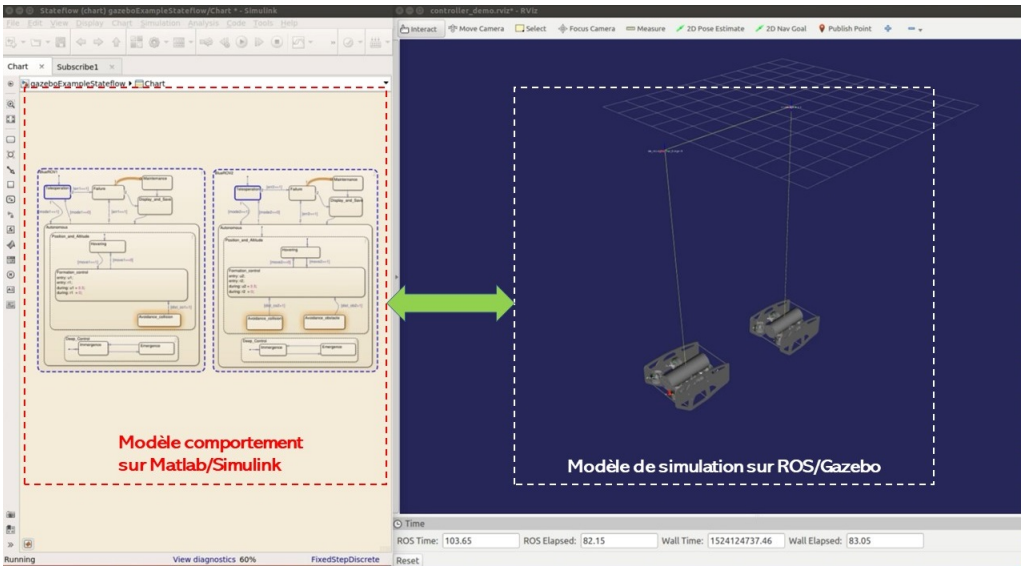


Figure 5.10 – La combinaison entre Matlab/Simulink et ROS/Gazebo

Des algorithmes tels que l'algorithme de contrôle de formation, l'algorithme d'évitement de collision entre robots, entre robots et obstacles sont développés dans le framework proposé (voir la figure 3.2) sur la partie modèle comportement (sur Matlab/Simulink), puis des signaux de contrôle sont envoyé directement au modèle de simulation (sur ROS/Gazebo). La structure des noeuds ROS sera détaillée au chapitre d'expérimentation 7.

5.4 Conclusion du chapitre

Ce chapitre a présenté la simulation du scénario de le suivi des formations, l'évitement des collisions et des obstacles d'un groupe UUV, dont le modèle dynamique est considéré comme linéaire. La forme de formation souhaitée qui est créée est maintenue et suit la trajectoire par l'algorithme de consensus tandis que la fonction potentielle est traitée pour un évitement de collision. En outre, ce chapitre a abordé les algorithmes pour les véhicules sous-marins à plate-forme qui sont validés avec un modèle indépendant de la plate-forme.

Les résultats de simulation du premier prototype du cycle 1 sont satisfaisants.

6

Coordination pour modèles non linéaires de sous-marin

L'utilisation du réseau neuronal a été largement appliquée ces dernières années, comme : traitement d'image, reconnaissance vocale, reconnaissance de l'écriture manuscrite et contrôle. En fonction de l'application spécifique, chaque exigence spécifique disposera de l'architecture de réseau de neurones appropriée. Cependant, la caractéristique commune de ces architectures de réseau est qu'elles utilisent les concepts suivants : fonctions d'activation, fonctions de perte, couches d'entrée et de sortie, poids, couches cachées. Pour étudier le modèle de robot non linéaire sous-marin avec des paramètres non définis, nous avons mené l'étude et utilisé deux architectures de Réseau de neurones : l'architecture **Recurrent Neural Network** et l'architecture **Radial Basic Network**.

6.1 Modèle de sous-marin non-linéaire

Comme expliqué au chapitre 5, nous avons l'équation de la dynamique du robot i, qui peut être remodelé dans un modèle d'espace d'états de la forme :

$$\dot{\mathbf{x}}_i = A_i \mathbf{x}_i + B_i \left[\tau_{hi} + \mathbf{f}_i\left(\mathbf{x}_i\right) + \mathbf{w}_i\left(t\right)\right] \tag{6.1}$$

où $\mathbf{x}_i \in \mathbb{R}^6$ est le vecteur d'état du robot i ;
$\mathbf{f}_i\left(\mathbf{x}_i\right) \in \mathbb{R}^3$ est l'incertitude inconnue du robot ;
$\mathbf{w}_i\left(t\right) \in \mathbb{R}^3$ représente la perturbation inconnue correspondante ;
$A_i \in \mathbb{R}^{6\times 6}$ et $B_i \in \mathbb{R}^{6\times 3}$ sont des matrices connues et (A_i, B_i) est supposé stabilisable.

$$A_i = \begin{bmatrix} 0_{3\times 3} & I_{3\times 3} \\ 0_{3\times 3} & 0_{3\times 3} \end{bmatrix}, \qquad B_i = \begin{bmatrix} 0_{3\times 3} \\ M_{hi}^{-1} \end{bmatrix} \tag{6.2}$$

$$\mathbf{f}_i\left(\mathbf{x}\right) = -D_{hi}(\mathbf{v}_{hi})\mathbf{v}_{hi} - g_{hi}\left(\mathbf{v}_{hi}\right), \qquad \mathbf{w}_i(t) = R\left(\psi_i\right) b \tag{6.3}$$

Selon [88], on choisit un contrôleur de retour local :

$$\tau_{hi} = K_r \mathbf{x}_i + \mathbf{u}_i \tag{6.4}$$

de manière à ce que $A + BK_r$ soit de *Hurwitz*. Où $\mathbf{u}_i \in \mathbb{R}^3$ est l'entrée de contrôle.

On considère un modèle de référence :

$$\dot{\mathbf{x}}_L = A_L \mathbf{x}_L + B_L r\left(\mathbf{x}_L, t\right) \tag{6.5}$$

où $\mathbf{x}_L \in \mathbb{R}^6$ représentent l'état du robot référence ; $r\left(\mathbf{x}_L, t\right) \in \mathbb{R}^3$ représente une entrée de référence. Le modèle de référence peut être considéré comme un leader virtuel dans le groupe et il génère une trajectoire souhaitée pour l'ensemble du groupe 6.1 à suivre.

Hypothèse 7 *Considérer un réseau G de systèmes multi-agents composé de N agents et d'un leader. Supposons que le réseau G soit non dirigé et connecté, et qu'au moins un agent ait accès à l'état du modèle de référence.*

Ici, le but de ce travail est de concevoir une loi de contrôle distribuée u_i pour chaque agent 6.1 afin de suivre le modèle de référence 6.5 de sorte que l'état de chaque agent se synchronise avec celui du modèle de référence, c'est-à-dire, $\mathbf{x}_i \to \mathbf{x}_L$.

Hypothèse 8 *L'incertitude du robot $\mathbf{f}_i\left(\mathbf{x}_i\right)$ peut être paramétrée linéairement par un réseau de neurones (RN) comme suit*

$$\mathbf{f}_i\left(\mathbf{x}_i\right) = W_i^{\top} \varphi_i\left(\mathbf{x}_i\right) + \varepsilon_i, \qquad \forall \mathbf{x}_i \in D \tag{6.6}$$

où $W_i \in \mathbb{R}^{s\times m}$ est une matrice de poids idéal constant et inconnu, et satisfait $\|W_i\| \leq W_{iM}$ avec $W_{iM} \in \mathbb{R}$ une constante positive ; $\varphi_i(\bullet) : \mathbb{R}^n \to \mathbb{R}^s$ est une fonction vectorielle connue de la forme $\varphi_i(x_i) = [\varphi_{i1}(x_i), \varphi_{i2}(x_i), ..., \varphi_{is}(x_i)]^\top$ satisfait $\|\varphi_i(x_i)\| \leq \varphi_{iM}$ avec $\varphi_{iM} \in \mathbb{R}$ étant une constante positive ; ε_i est l'erreur d'approximation satisfaisant $\|\varepsilon_i\| \leq \varepsilon_{iM}$ avec $\varepsilon_{iM} \in \mathbb{R}$ une constante positive ; $D \subset \mathbb{R}^n$ est un domaine suffisamment grand.

6.2 Coordination pour modèle non linéaire du sous-marin en utilisant RNN

6.2.1 Architecture de Recurrent Neural Network (RNN)

Un réseau récurrent est un réseau avec rétroaction ; certaines de ses sorties sont reliées à ses entrées. Cela lui permet d'afficher un comportement dynamique temporel. Contrairement aux réseaux de neurones à réaction, les RNN peuvent utiliser leur état interne (mémoire) pour traiter des séquences d'entrées. Cela les rend applicables à des tâches telles que la reconnaissance de l'écriture manuscrite ou la reconnaissance vocale non segmentée et connectée. Cette architecture utilise deux réseaux de neurones : un réseau de contrôleurs et un réseau de modèles du robot, comme le montre la figure 6.1. Le modèle d'installation est d'abord identifié, puis le contrôleur est formé pour que la sortie de l'installation suive la sortie du modèle de référence.

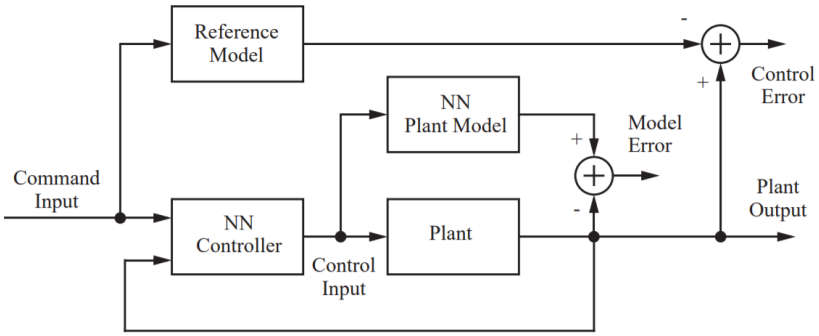


Figure 6.1 – Architecture de Contrôle Avec Modèle de Référence (CAMR) (selon l'article [119])

L'architecture de modèle de référence exige qu'un contrôleur par réseau neuronal (RN) distinct soit entraîné. La formation des contrôleurs est coûteuse sur le plan informatique, car elle nécessite l'utilisation d'une *backpropagation* dynamique. La figure 6.2 montre les détails du modèle *processus physique* du réseau de neurones et du contrôleur du réseau de neurones. Il existe trois jeux d'entrées de régulateur : les entrées de référence retardées, les sorties de régulateur retardées (entrées de *processus physique*) et les sorties de *processus physique* retardées. Pour chacune de ces entrées, nous sélectionnons le nombre de valeurs retardées à utiliser. Généralement, le nombre de retards augmente avec la commande de

processus physique. Il y a deux ensembles d'entrées pour le modèle de *processus physique* du réseau de neurones : les sorties de contrôleur retardées et les sorties de *processus physique* retardées.

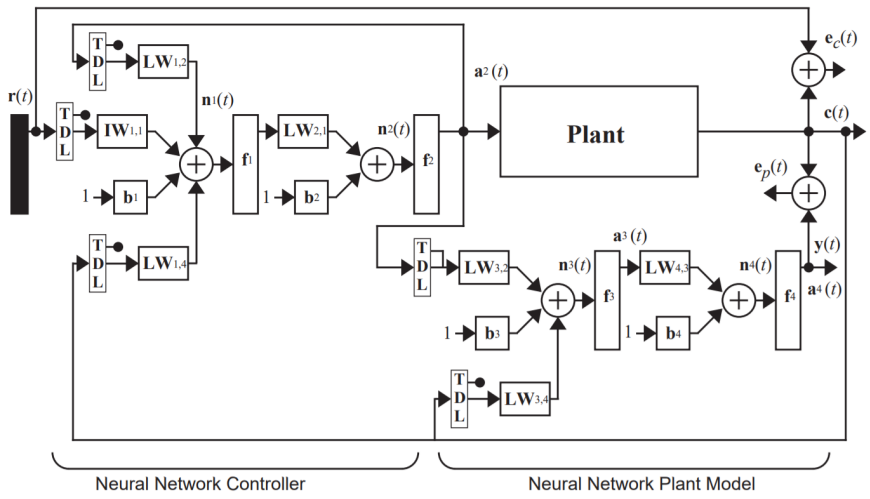


Figure 6.2 – Structure de contrôle avec modèle de référence détaillée (selon l'article [119])

Pour l'identification non linéaire des boîtes noires, nous devons également nous assurer que les entrées et sorties du système couvrent la plage de fonctionnement pour laquelle le contrôleur sera utilisé. Pour nos applications, nous recueillons généralement des données d'entraînement tout en appliquant des entrées aléatoires qui consistent en une série d'impulsions d'amplitude et de durée aléatoires. La durée et l'amplitude des impulsions doivent être choisies avec soin pour permettre une identification précise. Si l'identification est mauvaise, le système de contrôle qui en résulte peut échouer. Lorsque les performances en régime permanent sont faibles, il est utile d'augmenter la durée des impulsions d'entrée. Malheureusement, dans un ensemble de données de formation, si nous avons trop de données en conditions stables, les données d'entraînement peuvent ne pas être représentatives du comportement typique de la centrale. Ceci est dû au fait que les signaux d'entrée et de sortie ne couvrent pas adéquatement la région qui va être contrôlée. Nous devons choisir les données d'entraînement de manière à produire des performances transitoires et stables adéquates.

Les données utilisées pour former le contrôleur de référence du modèle sont générées en appliquant un signal de référence aléatoire qui consiste en une série d'impulsions d'amplitude et de durée aléatoires. Ces données peuvent être générées sans exécuter le *processus physique*, mais en utilisant la sortie du modèle de réseau neuronal à la place de la sortie de le *processus physique*.

6.2.2 Identification modèle du robot en utilisant un RN

Le but de cette section est de déterminer les paramètres d'état du robot sous-marin une fois que les paramètres d'entrée et de sortie sont disponibles. Pour créer l'ensemble de données d'entrée et de sortie du robot sous-marin, on utilise l'équation du mouvement qui est la suivante :

$$\begin{cases} \dot{x} = u\cos\theta \\ \dot{y} = u\sin\theta \\ \dot{\theta} = r \end{cases} \tag{6.7}$$

où (x, y, θ) sont la position et l'angle du robot ; (u, r) sont les signaux de contrôle. Plus précisément, le but ici est lorsque nous avons des données d'entrée et de sortie, grâce à l'utilisation du réseau pour pouvoir déterminer la forme de l'équation 6.7.

Ensuit, l'objectif est de former le contrôleur pour que le robot suive le modèle de référence :

$$\begin{cases} \dot{x}_r = 0.05u_r\cos\theta_r \\ \dot{y}_r = 0.08u_r\sin\theta_r \\ \dot{\theta}_r = 0.048r_r \end{cases} \tag{6.8}$$

L'architecture de contrôle de modèle de référence comporte deux sous-réseaux (voir la figure 6.3). Un sous-réseau est le modèle du robot que nous voulons contrôler. L'autre sous-réseau est le contrôleur.

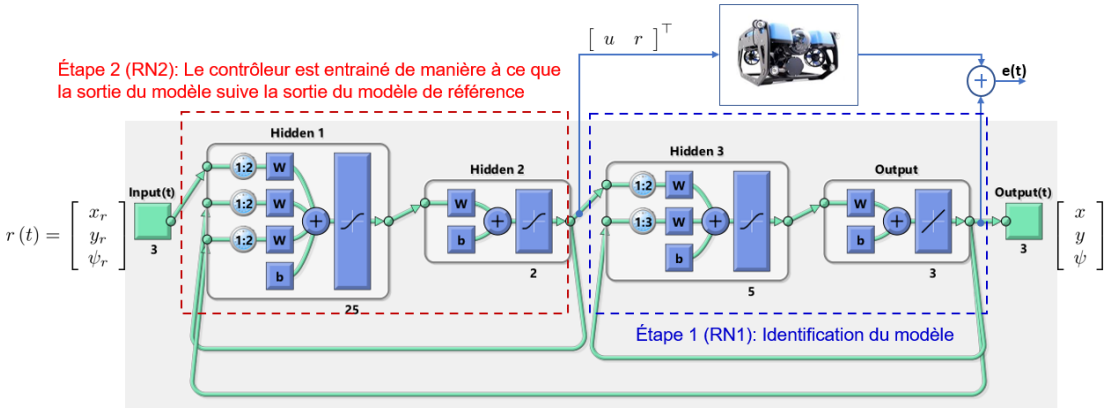


Figure 6.3 – Architecture CAMR utilisant Recurrent Neural Network

Nous commencerons par former un réseau NARX (en anglais *Nonlinear Autoregressive Exogenous*) qui deviendra le sous-réseau modèle du robot. Pour cette étude, nous utiliserons le BlueROV pour représenter le modèle du robot comme l'équation 6.7. Premièrement, les données recueillies seront créées à partir du modèle BlueROV dans Simulink (voir la figure 6.4) et utilisées pour former un réseau NARX. Pour ce problème simple, nous n'avons pas besoin de pré-traiter les données, et toutes les données peuvent être utilisées pour la formation, donc aucune division des données n'est nécessaire. Les signaux d'entrée u et r sont

des signaux aléatoire uniformément distribué de $(-1, 1)$ avec un temps d'échantillonnage de $0.5s$ (voir la figure 6.4).

En utilisation de l'équation 6.7, avec ces signaux d'entrée u et r correspondants, nous aurons les signaux de sortie : la position x, y et l'angle ψ via le modèle sur Simulink. A terme, 10000 échantillons seront créés pour la formation de réseau neuronal.

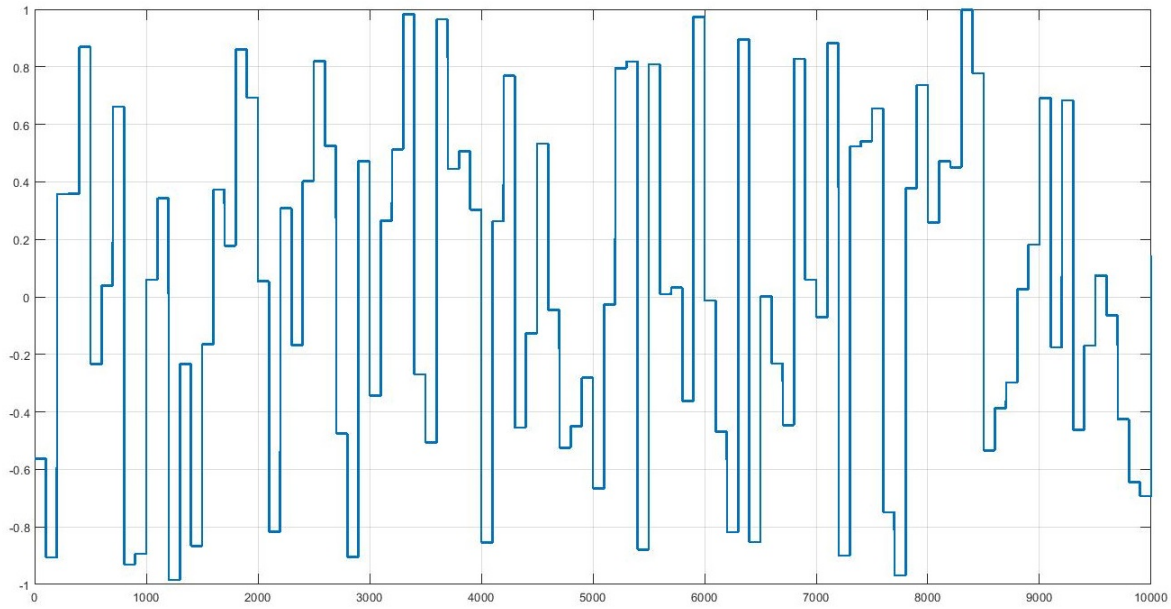


Figure 6.4 – Types de signaux d'entrée de u et r

Ensuite, nous définirons le réseau de neurones avec deux entrées et trois sorties comme indiqué sur la figure 6.3. Dans ce cas, deux entrées correspondent à u et r. Les trois sorties correspondent aux coordonnées sur les axes x, y et l'angle ψ. Ensuite, le réseau de neurones utilisera 10000 échantillons pour effectuer la formation et déterminer la matrice de poids. 6 neurones sélectionnés sont utilisés pour déterminer le modèle de robots. Après avoir déterminé la matrice de poids du réseau de neurones pour déterminer le modèle de robots, nous utiliserons ensuite l'architecture de réseau de neurones CARM, illustrée à la figure 6.1. Le réseau neuronal du système CARM est : 4 couches. Dans celle-ci, Les couche 3 et 4 sont l'architecture de réseau neuronal du modèle de robot. Ici, nous continuons à déterminer la matrice de poids des couches 1 et 2, ce qui équivaut au contrôleur du robot.

6.2.3 Résultats

Dans cette section, nous présenterons quelques résultats obtenus à partir de l'architecture de réseau de neurones présentée ci-dessus. Il se compose de deux étapes :

— Étape 1 : Identification du modèle BlueROV (Modèle de robot par réseau neuronal) : on utilise un réseau neuronal avec 6 neurones, 2 entrées et 3 sorties.

La figure 6.5 montre la distribution des erreurs. On voit que le nombre d'erreurs est principalement distribué dans une plage concentrée de -0.01001 à 0.004359.

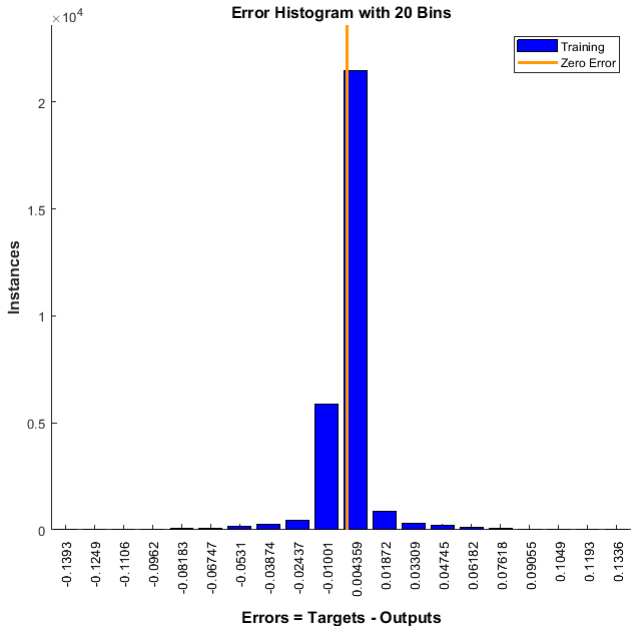


Figure 6.5 – Distribution d'erreur dans les réseaux de neurones d'apprentissage

La figure 6.6, 6.7, 6.8 montre le résultat de la comparaison de la sortie entre la valeur de coordonnée cible sur les axes x, y, l'angle ψ et la valeur donnée par le réseau de neurones.

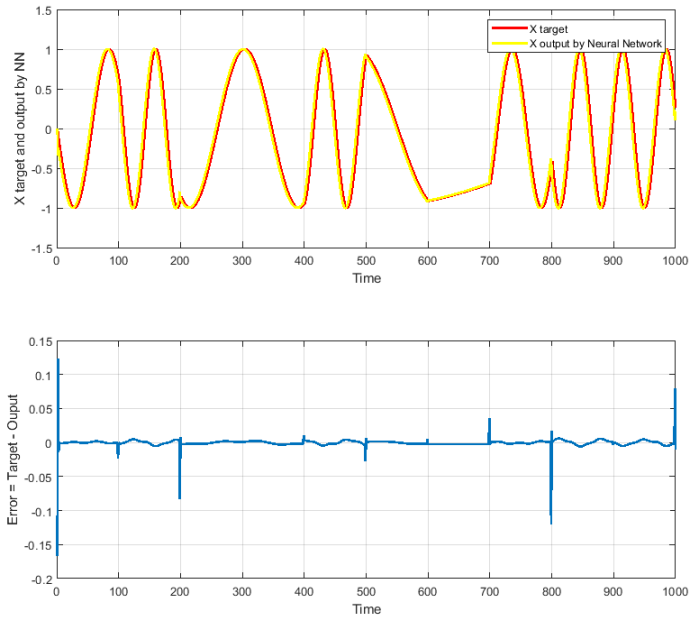


Figure 6.6 – Identification de X

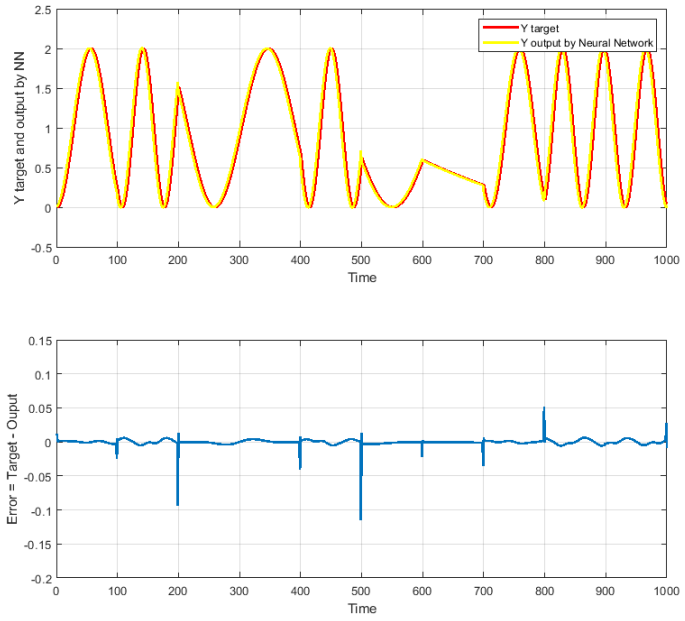


Figure 6.7 – Identification de Y

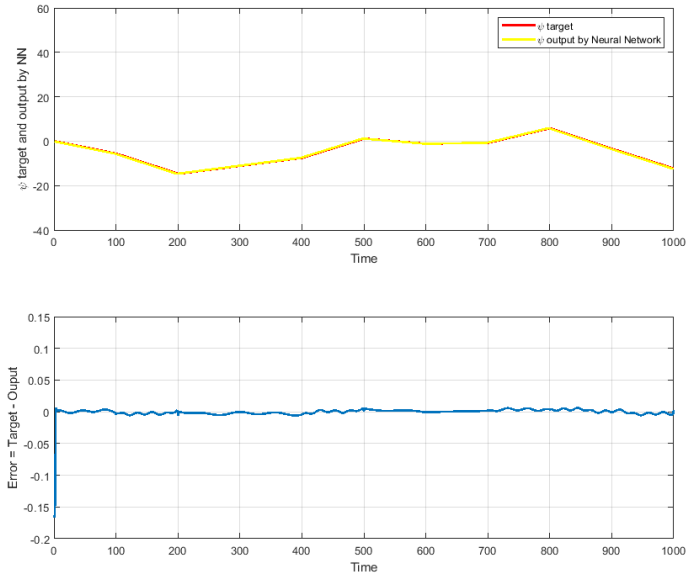


Figure 6.8 – Identification de ψ

On peut savoir que le réseau de neurones peut prédire avec précision le modèle du robot, qui est donné par l'équation 6.7. La valeur d'erreur est comprise entre -0.1 et 0.1.

— Étape 2 : Entraîner le contrôleur. Le contrôleur est entraîné de manière à ce que la sortie du modèle suive la sortie du modèle de référence. Le nombre de neurones utilisés : 25.

La figure 6.9 montre la distribution des erreurs pour la formation du contrôleur du robot. Les valeurs sont comprises entre -0.118 et 0.03336. On peut constater que la plage de résolution est plus large que le réseau neuronal pour l'identification du modèle de robot.

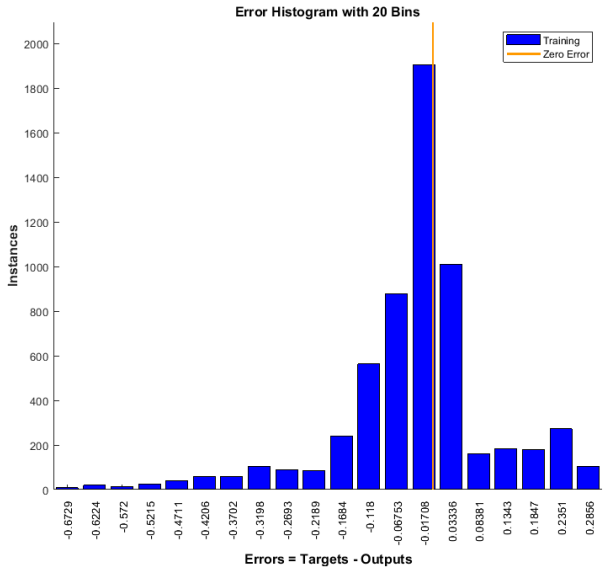


Figure 6.9 – Distribution d'erreur dans les réseaux de neurones d'apprentissage

La figure 6.10, 6.11, 6.12 montre le résultat de la comparaison de la sortie entre la valeur de coordonnée cible sur les axes x, y, l'angle ψ et la valeur donnée par le réseau de neurones.

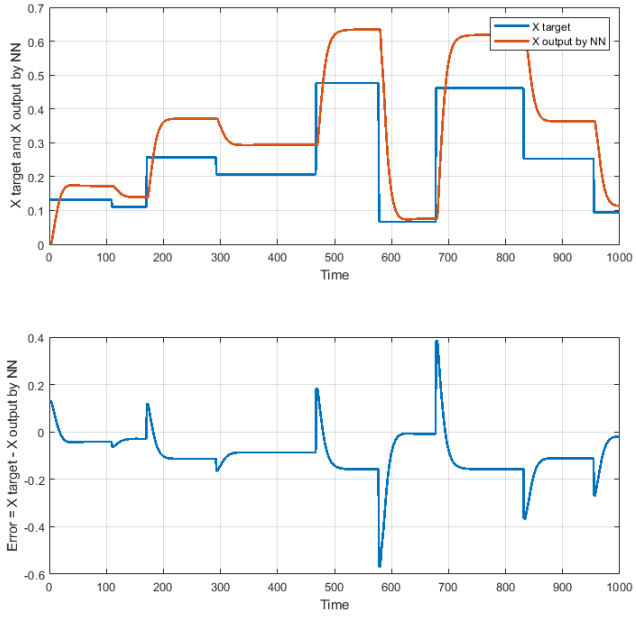


Figure 6.10 – Contrôle en x avec 40 neurones

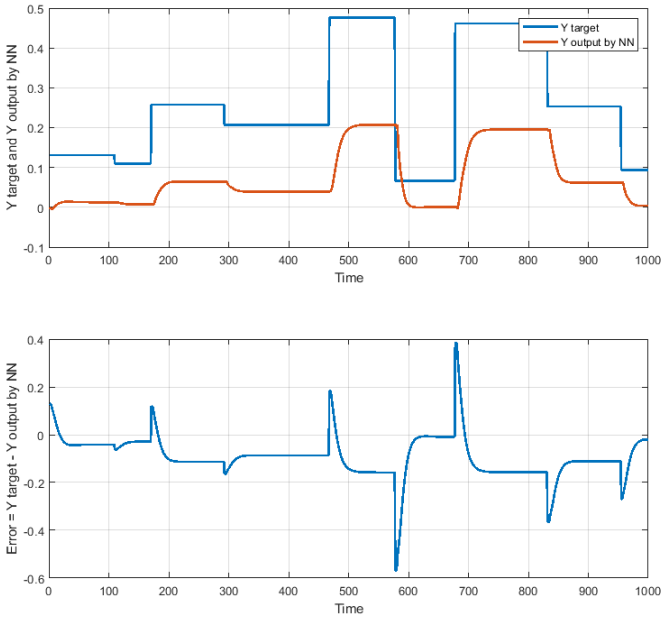


Figure 6.11 – Contrôle en y avec 40 neurones

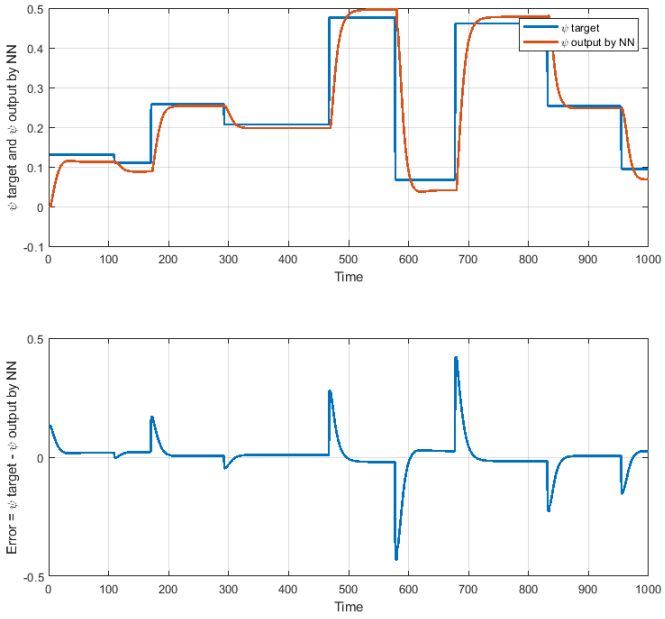


Figure 6.12 – Contrôle en ψ avec 40 neurones

Le réseau de neurones du contrôleur peut produire une valeur de sortie qui correspond à la forme du signal cible. Cependant, l'erreur est toujours élevée. La valeur donnée par le réseau de neurones ne suit pas vraiment la valeur souhaitée.

6.2.4 Conclusion sur l'utilisation RNN

Nous avons étudié l'utilisation d'un réseau de neurones RNN pour pouvoir déterminer les paramètres dynamiques du modèle non linéaire du robot sous-marin. Dans ce cas un modèle dynamique du robot peut être déterminé en fonction des ensembles de données d'entrée et de sortie pré-collectés. Cependant, la méthode que nous avons utilisée a ses limites : les paramètres sont appris *hors ligne*, ce qui signifie que nous devons collecter les données d'entrée et de sortie, puis via le réseau de neurones pour définir le modèle dynamique du robot.

En effet, l'utilisation de cette méthode d'apprentissage hors ligne entraîne le jeu de données obtenu uniquement dans une plage de valeurs apprises. Lorsqu'il y aura un petit changement (influences environnementales, changements de paramètres mécaniques, moteurs), les paramètres appris deviendront inexacts. De plus, cette méthode d'apprentissage hors ligne nécessite un temps de calcul très long. Il n'existe pas de méthode vraiment efficace d'entraînement du RNN dans le cas d'ensembles de données complexes.

6.3 Coordination pour modèle non linéaire du sous-marin en utilisant l'architecture de Radial Basis Network

Le but de cette section est de porter en particulier sur les problèmes de contrôle adaptatif par le réseau neuronal et les problèmes liés à la coordination de plusieurs sous-marins. Plus précisément, nous analyserons et utiliserons le réseau RBF, avec des algorithmes d'apprentissage en ligne pour être en mesure d'exclure des paramètres indéfinis du modèle de dynamique du robot sous-marin.

Les avantages du contrôleur de formation proposé sont les suivants : premièrement, la méthode proposée n'utilise que les mesures de la portée et de l'angle de visée par des capteurs locaux, aucune autre information sur le leader n'est requise pour la mise en œuvre du contrôle ; deuxièmement, le contrôleur de formation neuronale développé est capable de capturer la dynamique du véhicule sans informations exactes sur la force de Coriolis et la force centripète, l'amortissement hydrodynamique et les perturbations de l'environnement. Enfin, en utilisant l'algorithme d'entraîner en ligne, il n'est pas nécessaire de disposer de données d'entraînement. Cela peut étendre l'opérabilité du robot sous marin. Il convient également de noter que l'étude du contrôle de la formation de sous-marins par la méthode traditionnelle est introduite dans [102], [103], [120]. Cependant, dans [102], [103], [120], l'auteur n'a pas étudié le composant l'incertitude du robot. C'est donc la motivation de nos recherches.

6.3.1 Architecture de Radial Basis Function Network (RBF)

6.3.1.1 Introduction de RBF

L'architecture *Radial Basis Function* (RBF) a été formulé par Broomhead et Lowe en 1988 (voir la figure 6.13). Puisque RBF n'ont qu'une seule couche cachée, la convergence de l'objectif d'optimisation est beaucoup plus rapide. Le RBF présente les caractéristiques suivants :

— Les nœuds cachés implémentent un ensemble de fonctions de base radiales (par exemple des fonctions gaussiennes).
— Les nœuds de sortie implémentent des fonctions de sommation linéaire comme dans un *Multi-Layer Perceptron*.
— L'entraînement en réseau est divisé en deux étapes : d'abord les "poids" de la couche d'entrée à la couche cachée sont déterminés, et ensuite les poids de la couche cachée à la couche de sortie.
— L'apprentissage est très rapide.
— Les réseaux sont très bons pour l'interpolation.

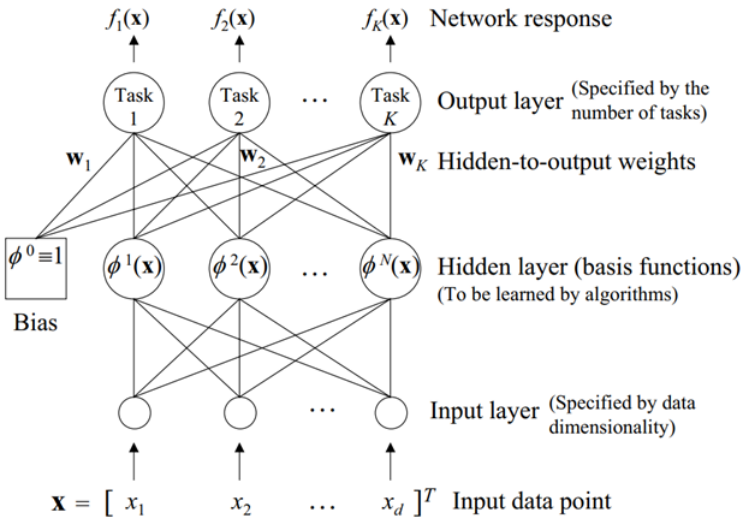


Figure 6.13 – Architecture de Radial Basis Function Network (RBF)

6.3.1.2 Algorithme d'apprentissage

Dans les applications RN en boucle ouverte telles que l'identification, la prédiction, la classification et la reconnaissance des formes, la limitation des poids RN implique à elle seule la stabilité du système global, puisque le système en boucle ouverte est supposé stable.

Cependant, dans les applications de contrôle par rétroaction en boucle fermée, la limite

des poids à elle seule ne démontre pas grand-chose. Par conséquent, les algorithmes standard d'entraînement au poids en boucle ouverte ne suffisent pas dans les systèmes de contrôle en boucle fermée. Là, il faut garantir que les poids RN restent limités et que l'erreur de suivi reste faible et que tous les états internes restent limités [121]. Il est maintenant nécessaire de montrer comment régler les poids RN en ligne afin de garantir un suivi stable. L'algorithme d'accord trouvé modifiera vraisemblablement les poids réels W de sorte qu'ils deviennent proches des poids idéaux W, qui sont inconnus. La figure 6.14 présente certains des algorithmes d'apprentissage actuels, ainsi qu'une comparaison de la vitesse de performance et des exigences matérielles requises.

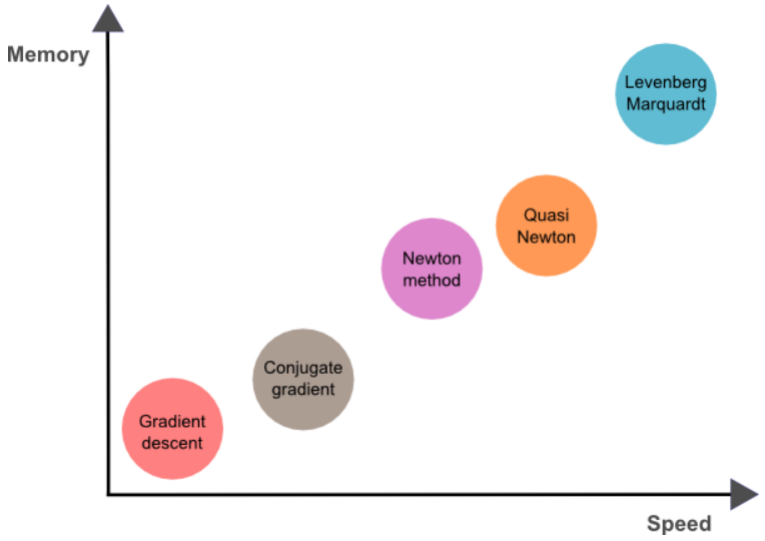


Figure 6.14 – Comparer les performances des algorithmes d'apprentissage

6.3.2 Proposition d'architecture de contrôle complète pour un groupe des UUVs

Cette section couvrira l'architecture de contrôle pour un groupe des UUVs. Pour compléter le modèle de recherche, nous considérons l'équation des robots comme étant non-linéaire avec les paramètres dynamiques inconnus et un bruit modélisant l'influence de l'environnement.

Ainsi, nous avons étudié la loi du contrôle des suiveurs de robots i, qui se compose de quatre éléments : l'entrée de contrôle de suivi de formation u_{iFC}, l'entrée de contrôle robuste u_{iR}, l'entrée de contrôle de réseau neuronal u_{iNN}, et l'évitement de collision u_{iCA} (voir dans la figure 6.15). Par conséquent, nous avons une loi de contrôle distribuée pour robot-follower qui prendra la forme :

$$\mathbf{u}_i = u_{iFC} - u_{iNN} - u_{iR} + u_{iCA} \tag{6.9}$$

où $u_{iFC} \in \mathbb{R}^3$ est une commande linéaire distribuée nominale ;
$u_{iNN} \in \mathbb{R}^3$ est un terme adaptatif en présence d'une dynamique non-linéaire (signifie dans ce cas il est l'incertitude inconnue du robot $\mathbf{f}_i(\mathbf{x}_i) \in \mathbb{R}^3$) ;
$u_{iR} \in \mathbb{R}^3$ est un terme robuste ;
$u_{CA} \in \mathbb{R}^3$ est un terme de contrôle pour éviter les collisions entre robots ;

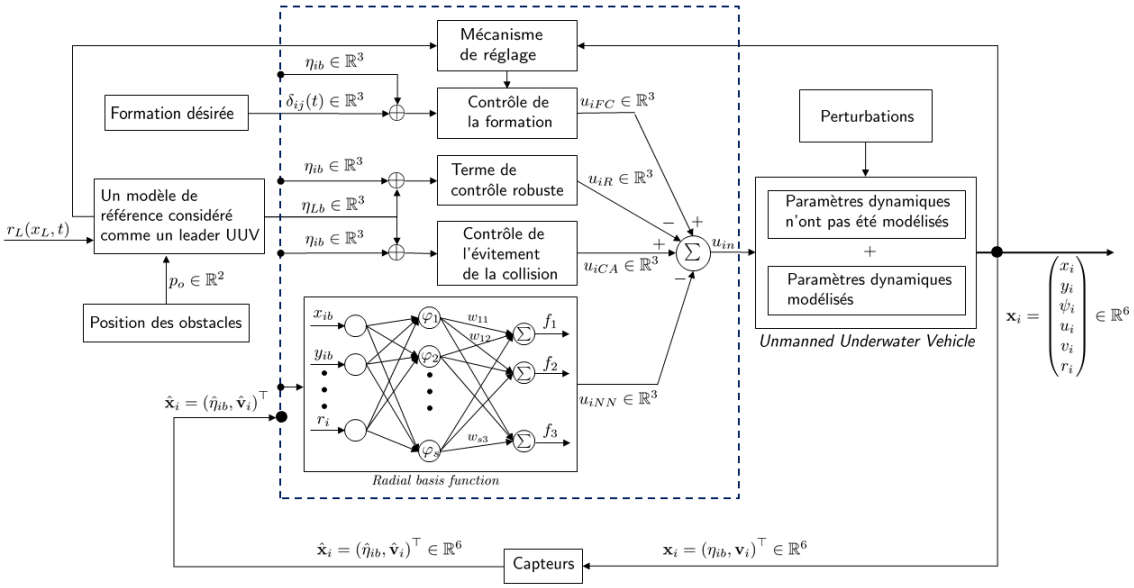


Figure 6.15 – Architecture de contrôle complète pour un groupe des UUVs

6.3.2.1 La commande de formation

Pour faire un contrôle de formation, nous définissons un vecteur variable auxiliaire, qui est une erreur de suivi position relative $\mathbf{e}_i \in \mathbb{R}^6$ pour l'UUV i comme

$$\mathbf{e}_i = \sum_{j \in N_i} a_{ij} \left(\mathbf{x}_i - \delta_i - (\mathbf{x}_j - \delta_j) \right) + a_{iL} \left(\mathbf{x}_i - \delta_i - \mathbf{x}_L \right) \tag{6.10}$$

Où $\delta_i - \delta_j, \forall i \neq j$, indique la formation souhaitée entre des robots sous-marins ; a_{ij} et a_{iL} sont définis dans l'équation 5.1 ; $\mathbf{x}_L \in \mathbb{R}^6$ représentent l'état du robot référence, qui est défini dans l'équation 6.5.

Une commande de suivi de formation adaptative distribuée est ensuite conçue sur la base du protocole de suivi consensuel et de la technique de linéarisation de la rétroaction, qui est donnée :

$$u_{iFC} = c_i K \mathbf{e}_i \tag{6.11}$$

où $K \in \mathbb{R}^{3\times 6}$ est un gain de contrôle de retour ; $c_i \in \mathbb{R}$ est un poids mis à jour sous la forme

$$\dot{c}_i = \Gamma_{ic} \Pr\left(c_i, \mathbf{e}_i^\top PBB^\top P\mathbf{e}_i\right) \tag{6.12}$$

où $\Gamma_{ic} \in \mathbb{R}$ est une constante positive, et P est la solution positive définitive de l'équation de Riccati algébrique suivante :

$$A^\top P + PA + Q - PBB^\top P = 0 \tag{6.13}$$

où $Q \in \mathbb{R}^{6\times 6}$ est une matrice positive-definite ; les matrices A et B sont définies dans l'équation 6.2.

Selon [122], une condition nécessaire et suffisante pour l'existence d'un P satisfaisant à l'équation de Riccati 6.13 est que (A, B) soit stabilisable.

6.3.2.2 Terme de contrôle des réseaux neuronaux

Comme la dynamique de l'UUV comporte des termes non modélisés et inconnus, un réseau neuronal de contrôle u_{iNN} est introduit pour compenser ces termes. Chaque UUV suivant aura un terme de réseau neuronal local pour suivre les estimations actuelles. Par conséquent, dans l'entrée de contrôle pour UUV i dans l'équation (6.11) est ajouté le terme u_{iNN}, qui est donné par (voir la figure 6.16)

$$u_{iNN} = \hat{W}_i^\top \varphi\left(\mathbf{x}_i\right) \tag{6.14}$$

où $\hat{W}_i \in R^{s_i\times m}$ est une estimation de W_i ;
s_i est le nombre de neurones du réseau neuronal pour l'UUV i ;
m est le nombre de valeurs d'entrée ; $\varphi(\mathbf{x}_i) \in R^{s\times 1}$ est la fonction d'activation.
$\mathbf{x}_i = (\eta_{ib}, \mathbf{v}_i)^\top = (x_{ib}, y_{ib}, \psi_{ib}, u_i, v_i, r_i)^\top \in R^6$ est le vecteur d'entrée ;

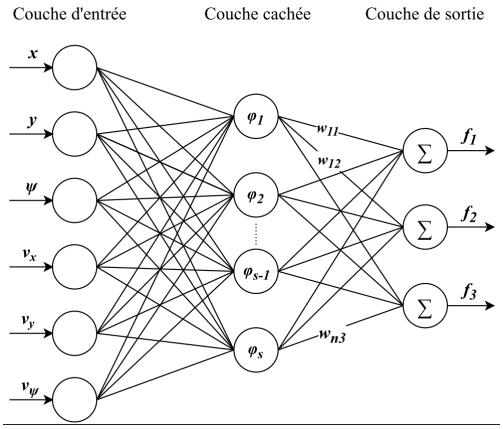


Figure 6.16 – Proposition un modèle d'utilisation du réseau RBF

Pour garantir que les matrices de poids du réseau de neurones restent bornées, l'algorithme de projection [92, 123] est adopté pour mettre à jour les paramètres

$$\dot{\hat{W}}_i = \Gamma_{iW} \Pr\left(\hat{W}_i, \varphi_i(\mathbf{x}_i)\, \mathbf{e}_i^\top PB\right) \tag{6.15}$$

où $\Gamma_{iW} \in \mathbb{R}$ est une constante positive ;
Pr est l'opérateur de projection (une matrice $n \times m$), avec la forme générale décrite comme

$$\Pr(\mathbf{\Theta}, \mathbf{Y}, \mathbf{\Phi}) = [\Pr(\theta_1, y_1, \phi_1), ..., \Pr(\theta_m, y_m, \phi_m)] \tag{6.16}$$

avec

$$\Pr(\theta_i, y_i, \phi_i) = \begin{cases} y_i - \dfrac{\bigtriangledown\phi_i(\theta_i)(\bigtriangledown\phi_i(\theta_i))^\top}{\| \bigtriangledown\phi_i(\theta_i) \|^2} y_i\phi_i(\theta_i) & \text{si } \phi_i(\theta_i) > 0 \wedge y_i^\top \bigtriangledown \phi_i(\theta_i) > 0 \\ y_i & \text{autrement} \end{cases} \tag{6.17}$$

où $i = 1$ à m ; $\Theta = \begin{bmatrix} \theta_1 & \cdots & \theta_m \end{bmatrix} \in \mathbb{R}^{s\times m}$; $\mathbf{Y} = \begin{bmatrix} y_1 & \cdots & y_m \end{bmatrix} \in \mathbb{R}^{s\times m}$; $\mathbf{\Phi} = \begin{bmatrix} \phi_1 & \cdots & \phi_m \end{bmatrix} \in \mathbb{R}^{s\times m}$; $\phi_i : \mathbb{R}^k \to \mathbb{R}$ est une fonction convexe ; $\bigtriangledown$: dérivées partielles.

Dans ce cas, ϕ_i est choisi comme

$$\phi_i(\theta_i) = \frac{\theta_i^\top\theta - \theta_{iM}^2}{\epsilon_{\theta_i}\theta_{iM}^2} \tag{6.18}$$

avec $\theta_{iM} \in R$ est une borne de norme de projection, et ϵ_{θ_i} est une borne de tolérance de projection. La fonction Pr est montrée détaillée dans l'annexe A.2.

L'objectif est d'utiliser les états des robots voisins de l'UUV i pour évaluer la performance du contrôle actuel de l'UUV i avec les estimations actuelles $\hat{f}_i.(\mathbf{x}_i)$ de la fonction $f_i(\mathbf{x}_i)$. Cela se fait par un ajustement en ligne du poids du réseau de neurones $\hat{W}_i$ dans l'équation (6.15).

6.3.2.3 Terme de contrôle robuste

Le but est la conception du contrôleur robuste u_{iR}, qui est conçu en utilisant uniquement les informations locales. Ceci est essentiel pour les applications de drones sous-marins à faible coût, où le positionnement et la communication sont limités. Selon [87], un terme robuste est conçu pour garantir que l'erreur de suivi réalise une convergence asymptotique. Il est donné comme :

$$u_{iR} = \frac{S_i^2 B^\top P\mathbf{e}_i}{S_i \|B^\top P\mathbf{e}_i\| + \mathbf{e}^{-t}} \tag{6.19}$$

où S_i est mis à jour en tant que :

$$S_i = \Gamma_{iS} \int_0^t \left\| \mathbf{e}_i^\top PB \right\| dt \tag{6.20}$$

où $\Gamma_{iS} \in \mathbb{R}$ est une constante positive.

6.3.2.4 Le changement de forme de formation du robot

En ce qui concerne la prévention des collisions, nous divisons en deux cas (voir la figure 6.17) :

— Cas 1 : Modifier la trajectoire de mouvement de la formation du robot afin qu'il peut éviter les collisions. Ceci a été étudié en le chapitre 5 en utilisant l'algorithme de Bézier pour calculer la nouvelle trajectoire lorsqu'un obstacle est rencontré.

— Cas 2 : Modifier la forme et la taille entre les robots pour éviter les collisions. Dans ce cas, il est supposé que le robot sous-marin doit se déplacer dans une zone étroite. Par conséquent, la formation du robot changera (zoom arrière ou zoom avant) pour pouvoir surmonter les obstacles.

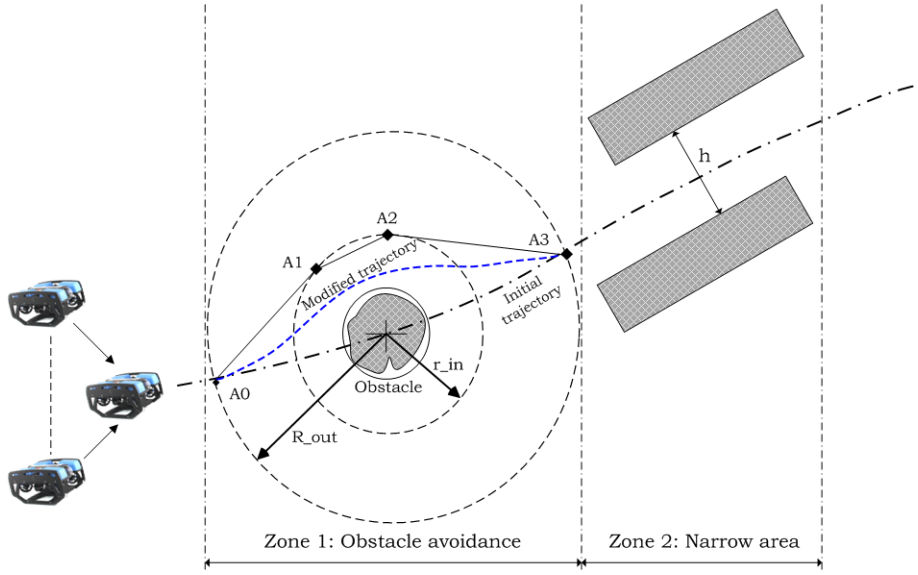


Figure 6.17 – Introduire deux cas d'évitement d'obstacles

En redéfinissant la fonction de configuration de formation δ_{ij}, qui peut changer dans le temps

$$\delta_{ij} = \delta_{ij}(t) = \delta_j(t) - \delta_i(t) \tag{6.21}$$

L'idée est de comparer la distance entre le robot leader et la zone étroite. Lorsque il se

trouve dans une zone étroite, un groupe d'UUV comparera une formation souhaitée $\delta_{ij}(t)$ avec la distance de la zone étroite h. Elle devra remplir les conditions suivantes

$$\delta_{ij}(t) = \begin{cases} \delta_{ij}(t) & \text{if } \delta_{ij}(t) < h - \Delta \\ h - \Delta & \text{if } \delta_{ij}(t) \geq h - \Delta \end{cases} \tag{6.22}$$

où Δ est une valeur donnée, qui dépend de la taille de l'UUV pour garantir que le groupe d'UUV se déplace dans une zone étroite sans collision, et h est la plus petite distance dans la zone étroite.

6.3.3 Résultats

L'objectif des expériences proposées est de coordonner les drônes sous-marins à faible coût en contrôlant simultanément certaines de leurs informations locales, et d'introduire un scénario de fonctionnement complet dans les environnements difficiles. Le contrôle distribué du réseau neuronal adaptatif et la prévention des collisions et des obstacles ont été testés et mis en œuvre dans le simulateur UUV [25], qui est basé sur ROS/Gazebo.

Nous considérons un schéma leader-suiveur composé de trois drônes sous-marins. Le modèle qui a été utilisé pour la simulation est un BlueROV-1, actionné en poussée, oscillation, soulèvement et lacet par une configuration à 6 propulseurs (le mouvement en roulis et tangage a été négligé car les deux DoF sont passifs et n'affectent que très peu les autres). Selon [124], les principaux paramètres inertiels et hydrodynamiques de BlueROV-1 sont donnés par :

$$M = M_{RB} + M_A = \begin{bmatrix} 12.81 & 0 & 0 \\ 0 & 20.01 & 0 \\ 0 & 0 & 0.12 \end{bmatrix}, \quad D = \begin{bmatrix} -28.27 & 0 & 0 \\ 0 & -134.74 & 0 \\ 0 & 0 & -0.07 \end{bmatrix} \tag{6.23}$$

puis

$$B = \begin{bmatrix} 0_{3\times 3} \\ M_i^{-1} \end{bmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.0781 & 0 & 0 \\ 0 & 0.0050 & 0 \\ 0 & 0 & 8.3333 \end{pmatrix} \tag{6.24}$$

et la masse de BlueROV-1 est $m = 7.31kg$, la longueur est $483mm$, la largeur est $330mm$, la hauteur est $267mm$. La dynamique non modélisée $\mathbf{f}_i$ et les perturbations $\mathbf{w}_i$ sont considérés

de manière aléatoire comme :

$$\mathbf{f}_i\left(\mathbf{x}_i\right)=\begin{bmatrix}-0.01u_iv_i^2\\-0.02u_iv_i\\-0.01r_iu_i\end{bmatrix}\text{ et }\mathbf{w}_i=\begin{bmatrix}-0.1\sin\left(0.5t\right)\cos\left(0.3t\right)\\-0.1\sin\left(0.2t\right)\cos\left(0.5t\right)\\-0.1\sin\left(0.1t\right)\cos\left(0.6t\right)\end{bmatrix} \tag{6.25}$$

La distance d'une zone étroite est choisie comme $h=8$m, $\Delta=2m$.
Le nombre de neurones du réseau neuronal pour chaque UUV a été ajusté à 10 neurones. Le nombre de valeurs d'entrée pour le réseau de neurones est 6. La fonction d'activation du réseau neuronal est choisie comme log-sigmoïde avec la forme $\phi(\mathbf{x}_i)=1/(1+e^{-\kappa t})$, où κ est une constante positive ($\kappa=2$ dans ce travail).
Une solution P de l'équation de Riccati avec $Q=\text{diag}\{1,1,1,1,1,1\}$ est

$$P=\begin{pmatrix}5.1595&0&0&12.8100&0&0\\0&6.4047&0&0&20.0100&0\\0&0&1.1136&0&0&0.1200\\12.8100&0&0&66.0926&0&0\\0&20.0100&0&0&128.1578&0\\0&0&0.1200&0&0&0.1336\end{pmatrix} \tag{6.26}$$

où P est une matrice à définition positive, puis

$$K=-B^\top P=\begin{pmatrix}-1.0000&0&0&-5.1595&0&0\\0&-1.0000&0&0&-6.4047&0\\0&0&-1.0000&0&0&-1.1136\end{pmatrix} \tag{6.27}$$

Dans cette simulation, K_r est conçu comme :

$$K_r=\begin{bmatrix}1&0&0&-7.7400&0&0\\0&1&0&0&-20.2800&-3.0345\\0&0&1&0&-0.6069&-8.2800\end{bmatrix} \tag{6.28}$$

On entre le modèle de référence comme suit

$$r=\begin{cases}\begin{bmatrix}0.05&0.05&0&.00\end{bmatrix} & 0\leq t\leq 90\\ \begin{bmatrix}0.00&0.10&0.00\end{bmatrix} & 90<t<130\\ \begin{bmatrix}0.00&0.00&0.00\end{bmatrix} & autrement\end{cases} \tag{6.29}$$

Les paramètres de commande adaptatifs sont pris comme suit $\Gamma_{iW}=1$; $\Gamma_{ic}=1$. Le nombre de neurones est 10. Nous allons d'abord développer des algorithmes sur Matlab que nous pouvons tester rapidement. Ensuite, nous développerons ces algorithmes afin de pouvoir expérimenter des modèles de simulation sur ROS/Gazebo.

6.3.3.1 Dans Matlab

Tout d'abord, nous effectuons des tests pour valider les algorithmes décrits ci-dessus sur le logiciel Matlab. La formation est conçue comme

$$\delta_1 = [0,0,0,0,0,0]^\top ; \delta_2 = [1,0,0,0,0,0]^\top ; \delta_3 = [0,2,0,0,0,0]^\top.$$

Les états initiaux des trois BlueROV-1 sont donnés par :

$$\mathbf{x}_1 = [0,-1,0,0,0,0]^\top ; \mathbf{x}_2 = [3,-2,0,0,0,0]^\top ; \mathbf{x}_3 = [-2,-2,0,0,0,0]^\top.$$

Les scénarios de simulation sont présentés comme suit :

— **Cas 1** : Contrôle total (y compris le contrôle de la vitesse et de la distance des robots) avec des composants : réseau de neurones pour éliminer les paramètres inconnus des robots, éviter les collisions entre les robots, changer de forme de formation pour éviter les obstacles. Cependant, la forme de la formation ne peut pas être tournée.

La figure 6.18 montre les trajectoires des 3 robots depuis la position initiale, puis la formation de formation, suivi du mouvement orbital. En plus de cela, ils peuvent également modifier la distance entre les robots pour pouvoir traverser une zone étroite (noter que dans ce cas, la distance et la position de la zone étroite sont déjà connues).

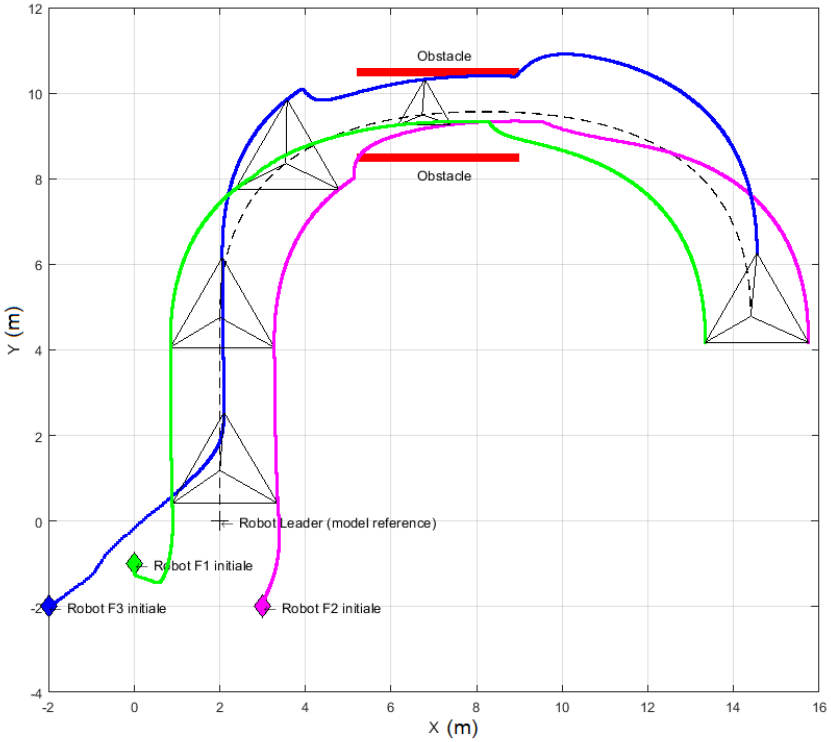


Figure 6.18 – Trajectoire des 3 robots dans cas 1

Sur la figure 6.19, on peut savoir que la distance entre les robots est toujours supérieure à $1m$. De plus, du temps $t = 0s$ à $t = 80s$, les distances convergent vers une valeur de $2,5m$. Cela garantit qu'il n'y a pas de collision entre les robots et en même temps ils gardent la formation (distance constante et égale à $2,5m$) pendant le mouvement. Du temps $t = 60s$ à $t = 80s$, c'est le moment où les robots modifient la distance entre eux pour pouvoir traverser une zone étroite (la distance entre les robots est de $1,4m$). Et après avoir traversé une zone étroite, ils se déplacent à nouveau pour obtenir une distance initiale de $2,5m$ entre eux.

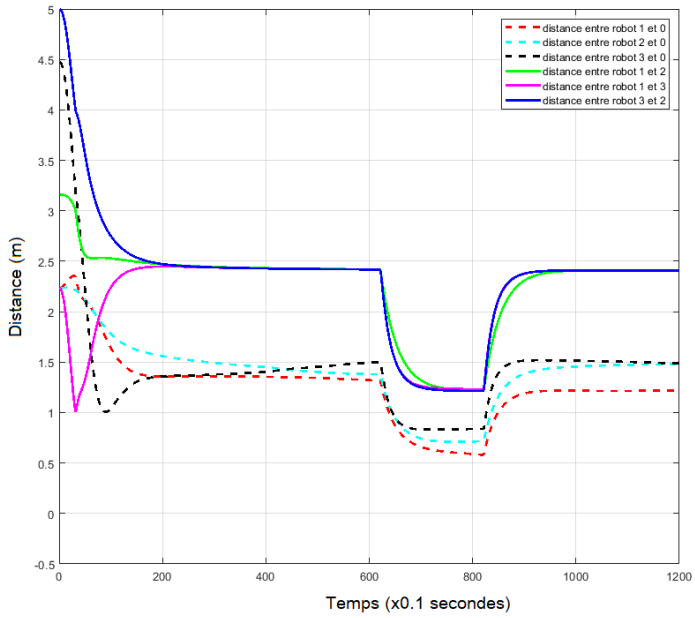


Figure 6.19 – Les distances entre 3 robots dans cas 1. (***La ligne continue montre la distance entre 3 robots sous-marins. La ligne en pointillés montre la distance entre les 3 robots sous-marins et le robot leader virtuel.***)

Les figures 6.20 et 6.21 montrent les vitesses des trois robots respectivement dans les axes x et y. Du temps $t = 0s$ à $t = 20s$, il y a la vitesse des robots pendant la formation. Du temps $t = 60s$ à $t = 80s$, il y a la vitesse des robots en train de changer la distance pour traverser une zone étroite.

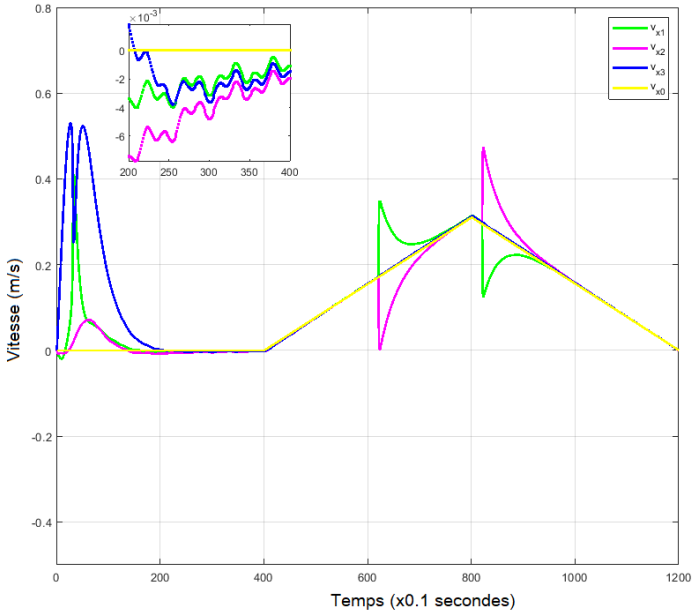


Figure 6.20 – La vitesse des robots sur l'axe x dans cas 1

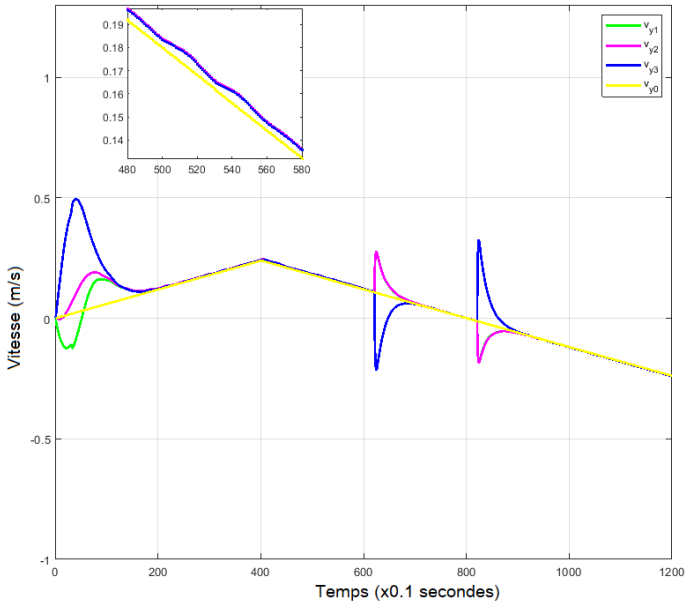


Figure 6.21 – La vitesse des robots sur l'axe y dans cas 1

— **Cas 2** : Identique au cas 1, mais n'utilise pas le composant réseau neuronal pour éliminer les paramètres inconnus des robots du temps $t = 20s$ à $t = 50s$.

La figure 6.22 montre les trajectoires des 3 robots depuis la position initiale, puis la mise de formation. En plus, ils peuvent modifier la distance entre les robots pour pouvoir traverser une zone étroite (on suppose que la distance et la position de la zone étroite sont connues).

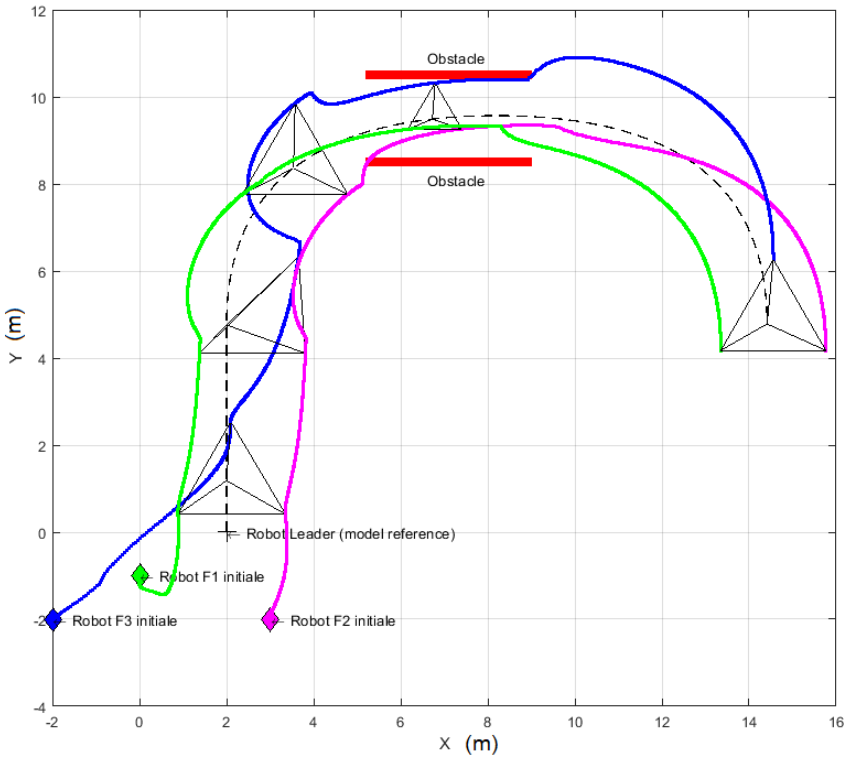


Figure 6.22 – Trajectoire des 3 robots dans cas 2

La figure 6.23 montre la distance entre les robots. Comme dans le cas 1, la distance entre les robots est toujours de $1m$ ou plus (pour s'assurer qu'il n'y a pas de collision entre eux) et converge à $2,5m$ (pour montrer que les robots peuvent maintenir la position relative constante). Cependant dans ce cas, du temps $t = 20s$ à $t = 50s$, on n'utilise pas le composant de contrôle RN pour compenser les paramètres inconnus des robots. Pour cette raison, nous pouvons voir que la distance entre les robots dans cette période n'est plus égale à $2,5m$.

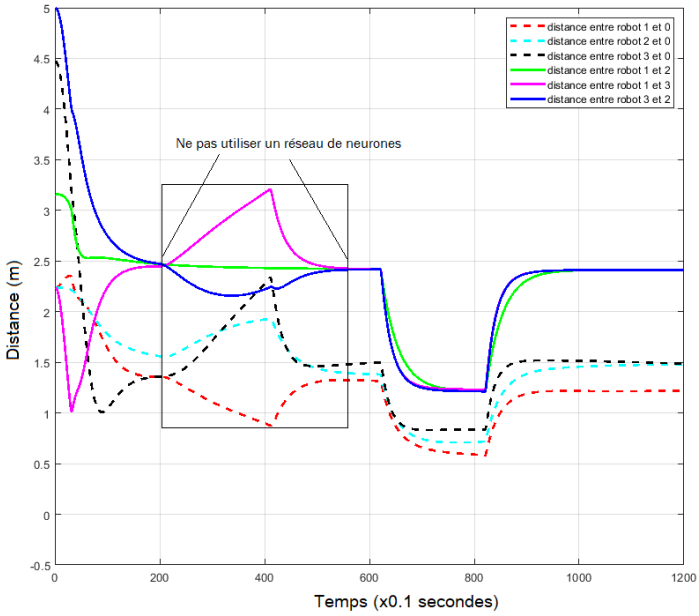


Figure 6.23 – Les distances entre 3 robots dans cas 2. (***La ligne continue montre la distance entre 3 robots sous-marins. La ligne en pointillés montre la distance entre les 3 robots sous-marins et le robot leader virtuel.***)

Les figures 6.24 et 6.25 sont les vitesses des robots dans leurs axes x et y respectifs. La vitesse des robots du temps $t = 20s$ à $t = 40s$ équivaut au temps où nous n'avons pas utilisé un composant de contrôle basé sur RN. Les temps $t = 60s$ et $t = 90s$ équivalent au début et à la fin du processus de changement de la distance entre eux pour traverser une zone étroite.

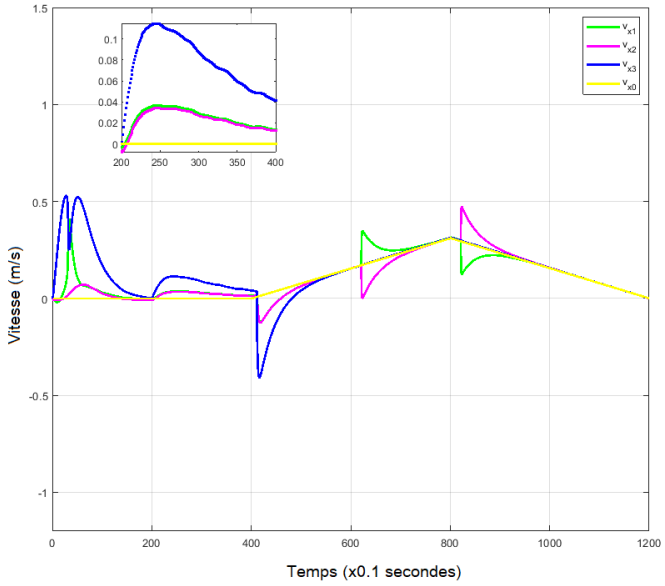


Figure 6.24 – Les vitesses des 3 robots dans leurs axes x dans cas 2

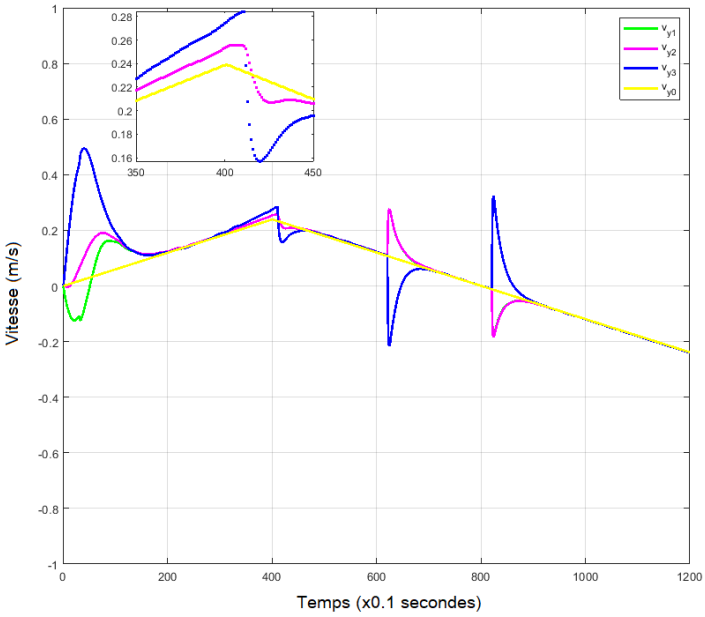


Figure 6.25 – Les vitesses des 3 robots dans leurs axes y dans cas 2

— **Cas 3** : Identique au cas 1, cependant la forme de la formation peut changer et tourner en fonction de la trajectoire du leader du robot.

La figure 6.26 montre les trajectoires des trois robots. Cependant, contrairement aux deux cas ci-dessus, dans ce cas, la formation des robots tourne autour du centre de la formation pour maintenir le robot leader au sommet. (Nous avons remarqué qu'en se déplaçant dans une zone étroite, le robot leader est toujours en première position par rapport aux deux autres robots).

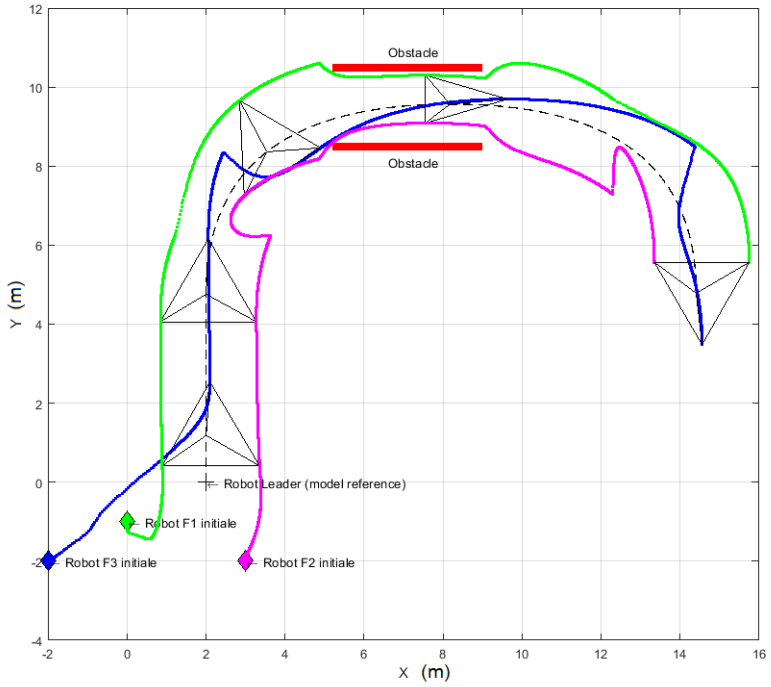


Figure 6.26 – Trajectoire des 3 robots sans évitement de collision entre robots dans cas 3

La figure 6.27 montre les distances de trois robots. Nous avons également remarqué que la distance entre les robots est toujours supérieure à $1m$ pour éviter toute collision entre les robots et converge à $2,5m$ pour dire que les robots peuvent maintenir une distance constante entre eux (équivaut à garder la formation). Du temps $t = 50s$ à $t = 60s$ et $t = 100s$ à $t = 110s$ sont le moment où la formation robotique tourne autour du centre de la formation pour garantir que la position du robot leader est toujours au sommet par rapport les deux robots restants. Temps $t = 70s$ à $t = 90s$, la distance entre les deux robots suiveurs 1 et 2 a changé (égale à $1,4m$) pour garantir que la formation du robot peut également traverser une zone étroite.

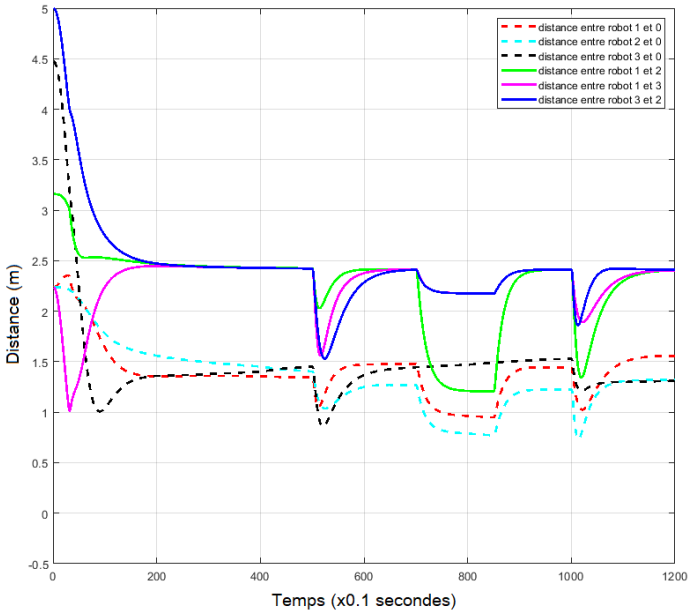


Figure 6.27 – Les distances entre 3 robots avec l'évitement de collision entre robots dans cas 3

Les figures 6.28 et 6.29 montrent les vitesses des robots sur les axes x et y respectivement. On remarque que la vitesse des robots change radicalement aux moments où la formation des robots tourne autour de son centre de gravité. Cela peut être résolu en réduisant les coefficients de contrôle, ou omis dans le cas de robots se déplaçant à faible vitesse.

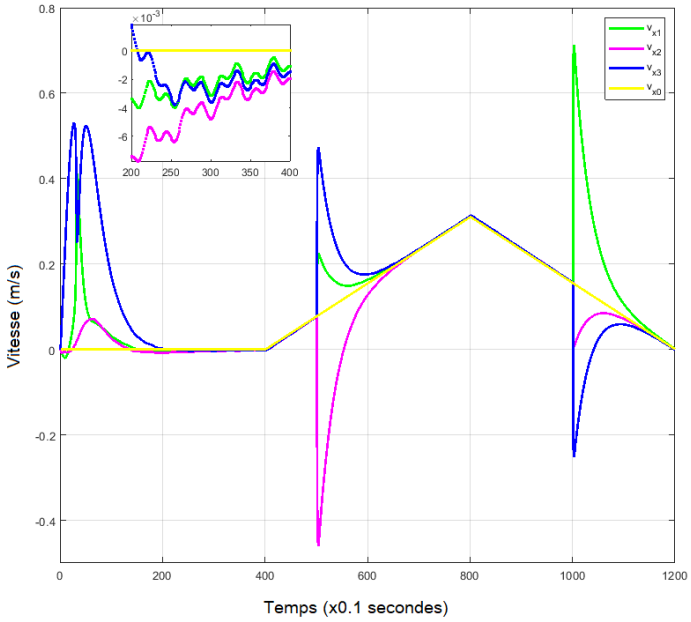


Figure 6.28 – Les vitesses des 3 robots avec l'évitement de collision entre robots dans cas 3

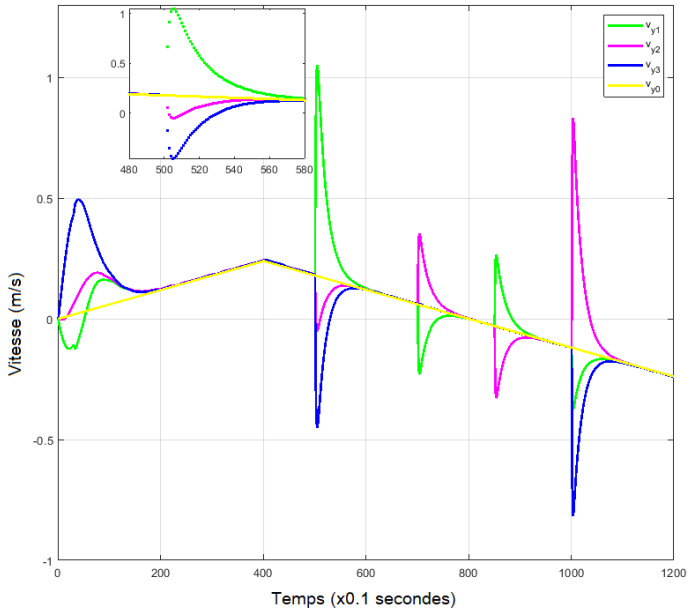


Figure 6.29 – Les vitesses des 3 robots avec l'évitement de collision entre robots dans cas 3

— **Cas 4** : Dans ce cas, supposons que les robots connaissent la position relative des robots adjacents et la vitesse des robots adjacents n'est pas déterminée. L'objectif de ce cas est d'étendre le contrôle des robots limités par les conditions de communication ainsi que des technologies de positionnement sous-marin.

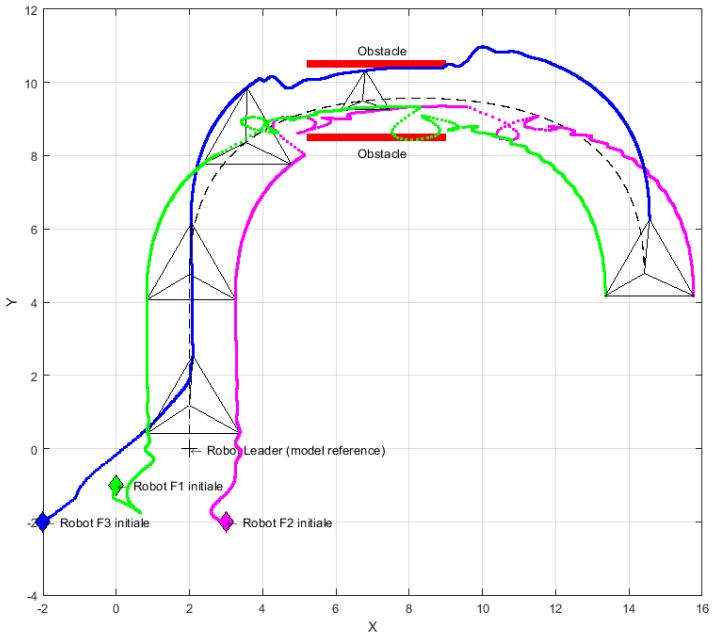


Figure 6.30 – Trajectoire des 3 robots sans évitement de collision entre robots dans cas 4

Comme les cas ci-dessus, la figure 6.30 montre la trajectoire de déplacement de 3 robots. Cependant, nous avons remarqué que les orbites des robots n'étaient pas *lisses* au cas où ils devraient changer de distance pour pouvoir traverser une zone étroite. En effet, dans ce cas, nous concevons le contrôleur en fonction uniquement de la position relative entre les robots sans connaître la vitesse des robots adjacents.

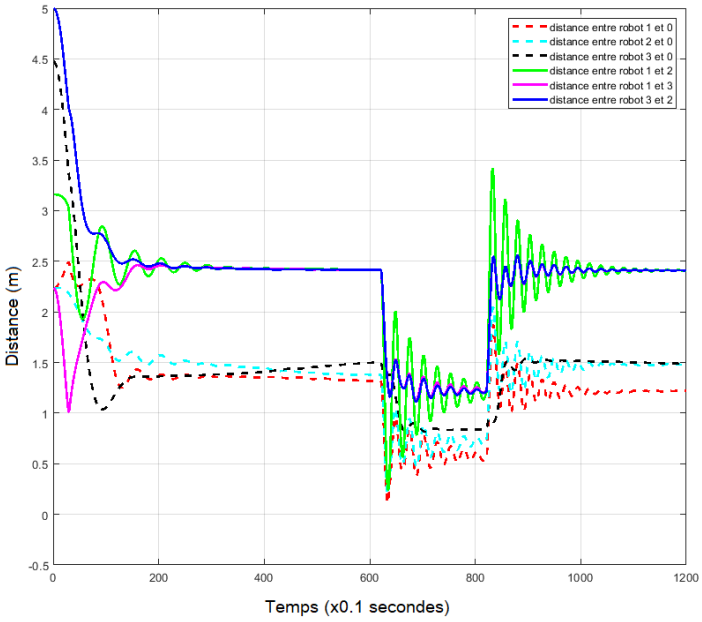


Figure 6.31 – Les distances entre 3 robots avec l'évitement de collision entre robots dans cas 4

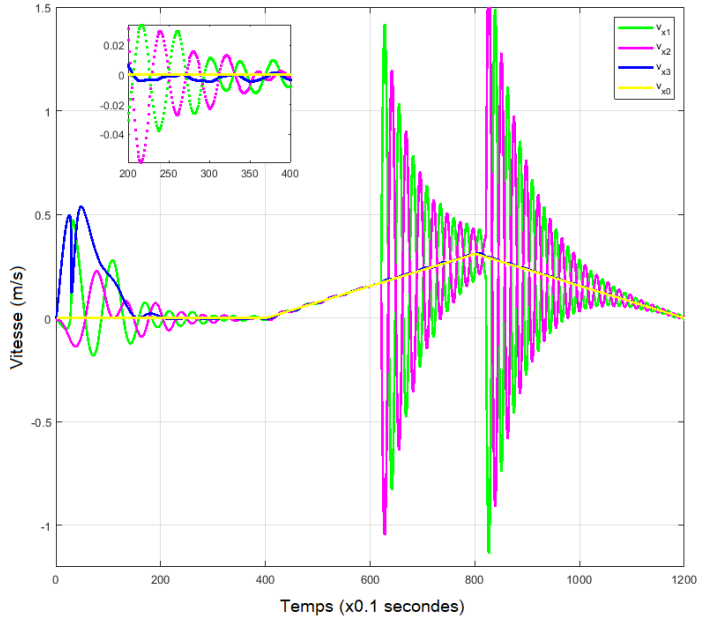


Figure 6.32 – Les vitesses des 3 robots avec l'évitement de collision entre robots dans cas 4

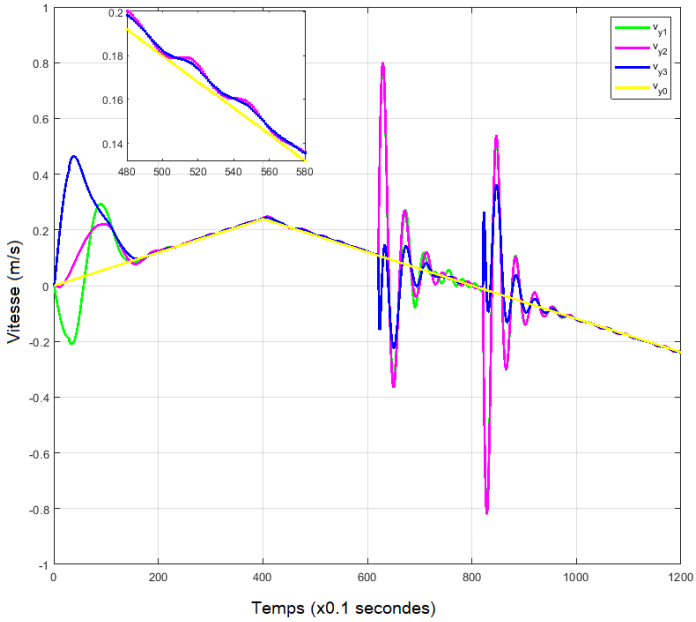


Figure 6.33 – Les vitesses des 3 robots avec l'évitement de collision entre robots dans cas 4

La figure 6.31 montre la distance entre les robots dans le cas 4. Cette distance est toujours supérieure à $1m$ et converge vers $2,5m$ pour s'assurer qu'il n'y a pas de collision entre les robots ainsi que les robots qui peuvent maintenir la formation en le processus de mouvement de trajectoire.

Les figures 6.32 et 6.33 montrent la vitesse des robots en se déplaçant respectivement sur les axes x et y. Nous avons également remarqué que la vitesse des robots dans ce cas fluctue tandis que les robots suiveurs doivent modifier leur distance pour traverser des zones étroites. Comme dans le cas 3, cela peut également être minimisé si l'on réduit les coefficients de contrôle ainsi que dans des conditions de test réelles à faible vitesse.

6.3.3.2 Dans ROS/Gazebo

Ensuite, nous testons les algorithmes de contrôle sur un modèle de simulation complet, qui a été développé sur la plate-forme de simulation ROS / Gazebo. La figure 6.34 présente l'architecture des nœuds et des topics sur ROS/Gazebo pour 3 robots sous-marins qui ont été développées.

Le leader UUV suppose qu'il a été contrôlé pour suivre la trajectoire avec les points de passage $(0, 6)$, $(0, 12)$, $(30, 12)$, $(30, 6)$. La position de l'obstacle est $(15, 9)$ et $D_{out} = 7m$, $d_{in} = 4m$. Les états initiaux des trois BlueROV-1 sont donnés par : $\mathbf{x}_1 = [6, 0, \frac{\pi}{2}, 0, 0, 0]^\top$; $\mathbf{x}_2 = [-6, 1, 0, 0, 0, 0]^\top$; $\mathbf{x}_3 = [0, 3, \frac{\pi}{2}, 0, 0, 0]^\top$.

La formation désirée est donnée par : $\delta_1 = [5\sqrt{3}, 0, 0, 0, 0, 0]^\top$; $\delta_2 = [0, 5, 0, 0, 0, 0]^\top$; $\delta_3 = [0, -5, 0, 0, 0, 0]^\top$.

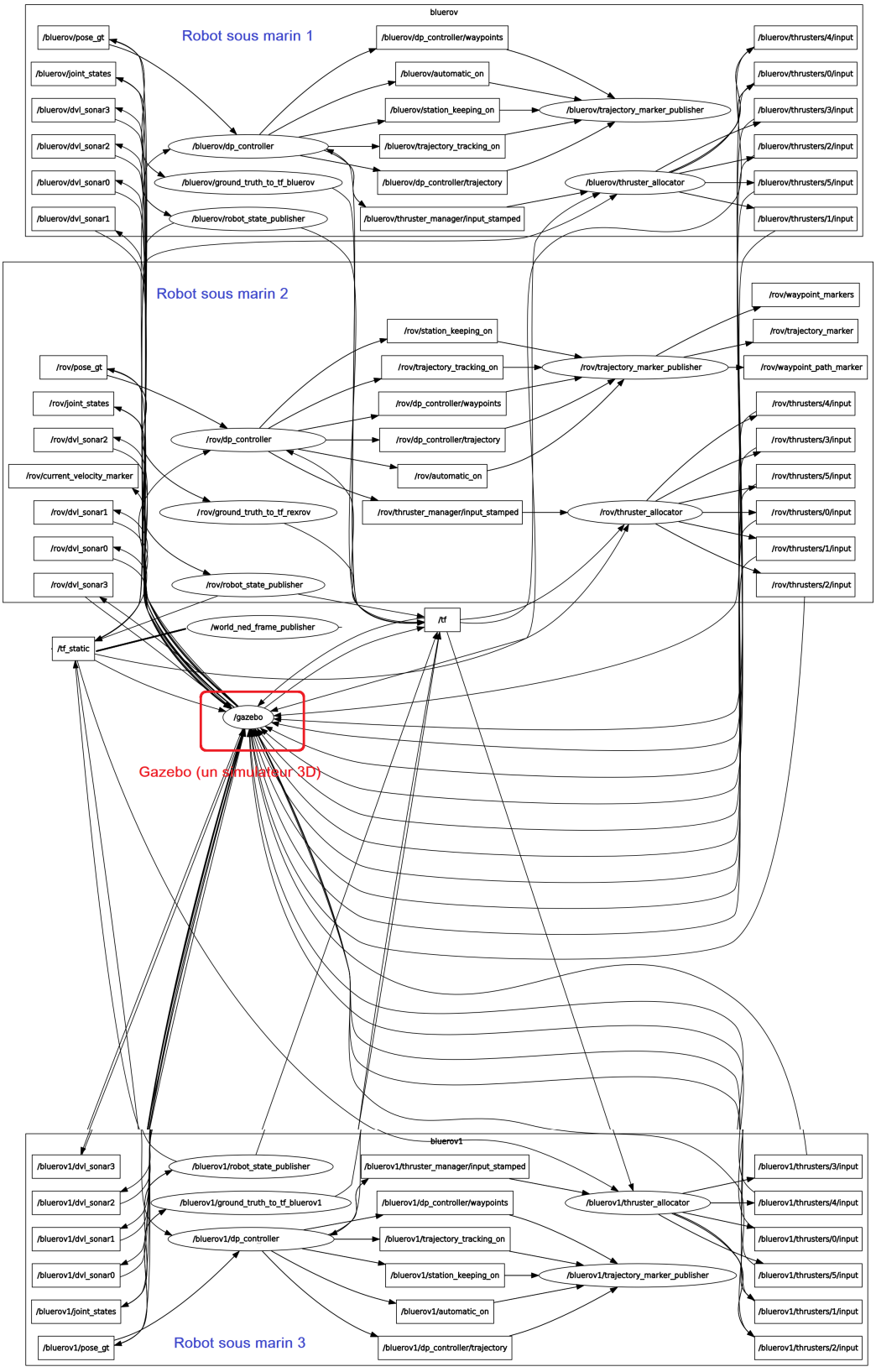


Figure 6.34 – Architecture des nœuds et des topics sur ROS/Gazebo pour 3 robots

Nous examinons d'abord le contrôle de formation sans terme d'évitement de collision. On peut observer que la distance minimale entre les UUV 2 et les UUV 3 est $d_{23} = 2m$ (voir figure 6.35). Cela signifie qu'il y a une collision entre les deux UUV. Nous examinons alors le contrôle de la formation en appliquant l'algorithme d'évitement de collision. La figure 6.36 montre que la distance minimale entre les robots est $d_{23} \approx 4m$. Cela garantit qu'aucune collision avec les paramètres sélectionnés est $D_{out} = 7m$, $d_{in} = 4m$, et $\sigma_{ij} = 3$.

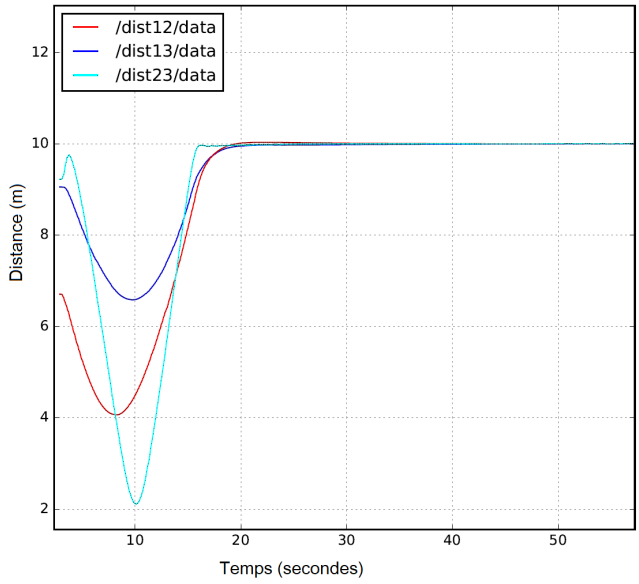


Figure 6.35 – Distances entre trois robots sous-marins sans terme d'évitement de collision. (*/dist ij/data* signifie la distance entre le robot i et j)

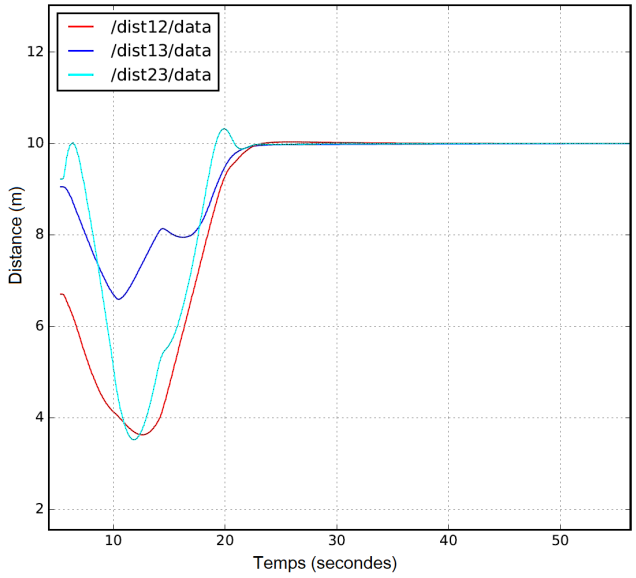


Figure 6.36 – Distances entre trois robots sous-marins avec un terme d'évitement de collision. (*/dist ij/data* signifie la distance entre le robot i et j)

Nous examinons plus en détail le contrôle de suivi des formations distribuées avec collision dans les environnements difficiles. Un groupe d'UUV part d'une configuration aléatoire, atteint une formation souhaitée, puis suit une trajectoire de leader UUV. Pendant un déplacement, un groupe d'UUV peut changer la forme de la formation pour traverser une zone étroite. La distance entre les UUV est indiquée sur la figure 6.37, qui converge vers $10m$. Cela prouve que la formation entre les robots est maintenue pendant le déplacement.

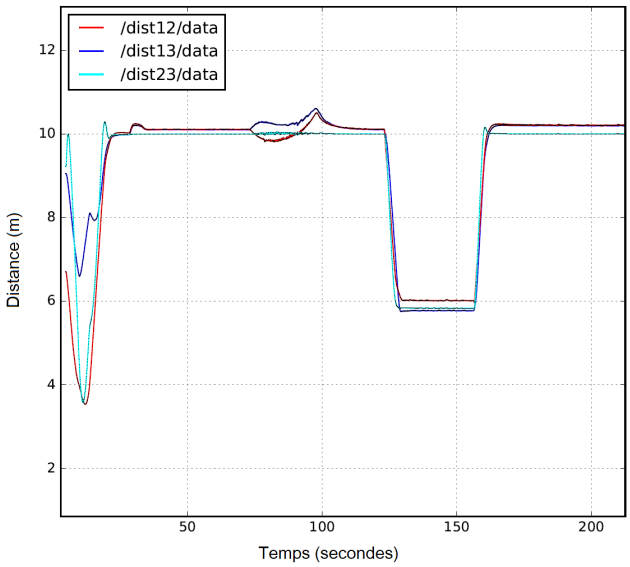


Figure 6.37 – Distances entre les trois BlueROV. (*Dans la période de* $t = 120s$ *à* $t = 170s$*, les robots ont changé leur distance entre eux pour pouvoir traverser une zone étroite.*)

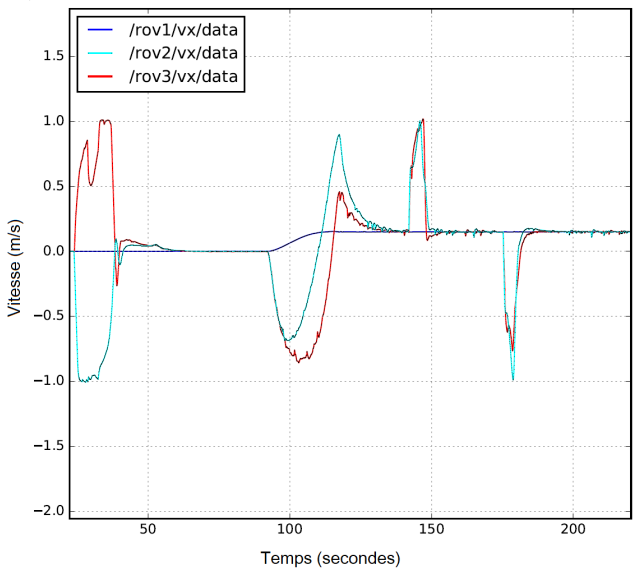


Figure 6.38 – Vélocités des trois robots dans l'axe x. (*/rov i/vx/data* signifie la vitesse du robot i, $(i = 1, 2, 3)$)

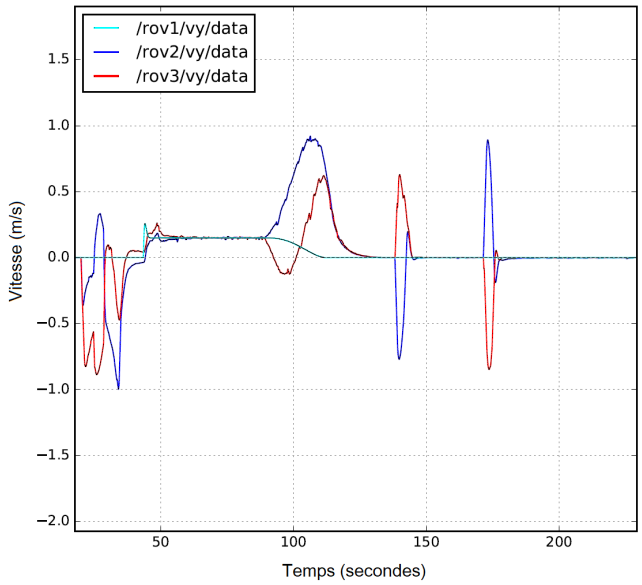


Figure 6.39 – Vélocités des trois robots dans l'axe y. (*/rov i/vy/data* signifie la vitesse du robot i, $(i = 1, 2, 3)$)

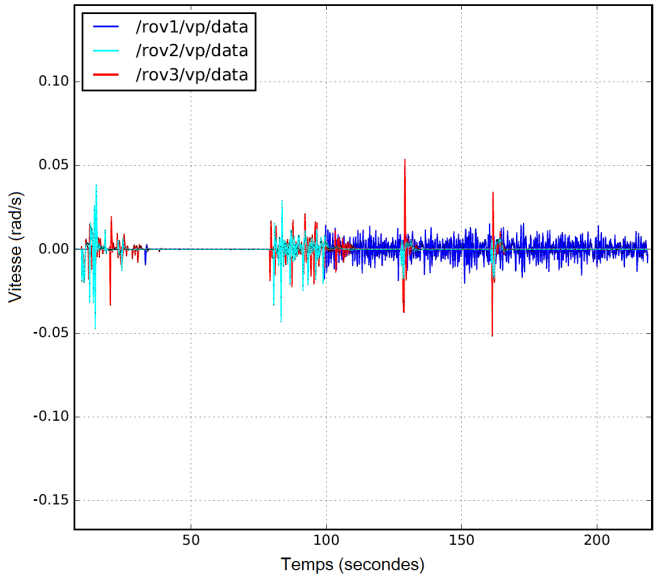
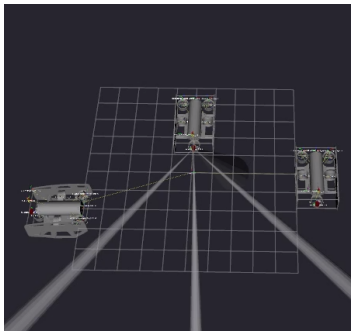


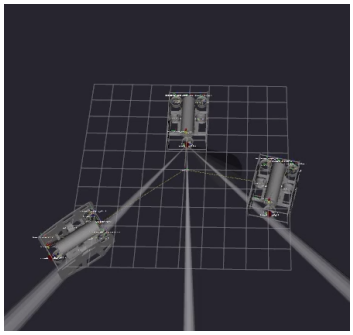
Figure 6.40 – Vitesse angulaire des trois robots $r_i \simeq 0$. (*/rov i/vp/data* signifie la vitesse angulaire du robot i, $(i = 1, 2, 3)$)

Les vitesses des trois BlueROV-1 sont illustrées sur la figure 6.38, 6.39, et 6.40, respectivement. Pendant la période allant de $t = 120s \rightarrow t = 170s$, on peut observer que le groupe de l'UUV a modifié la distance $d_{ij} = 6m$ pour traverser des zones étroites. Il convient de noter que la vitesse des UUV n'est pas régulière selon la longueur des axes, mais ce résultat est tout de même acceptable.

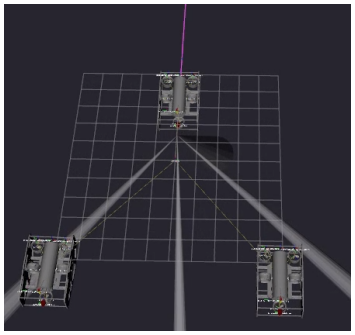
Un exemple de l'évolution de trois BlueROV-1 dans Gazebo est présenté dans la figure 6.41, où une visualisation 3D de la dynamique de BlueROV-1 est présentée sous les effets hydrostatiques et hydrodynamiques sous-marins.



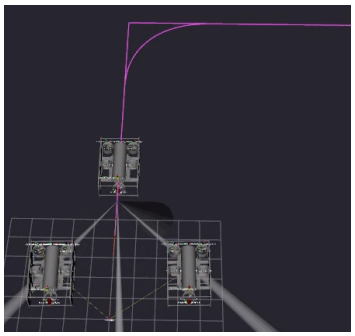
(a) $t = 0s$ **- états initiaux**



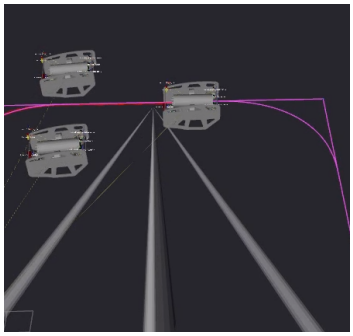
(b) $t = 10s$ **- contrôle de formation**



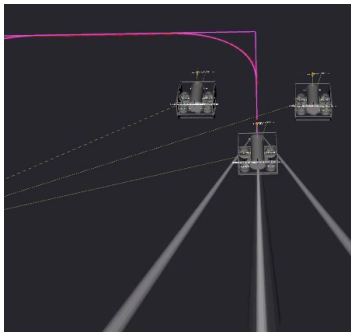
(c) $t = 20s$ **- contrôle de formation**



(d) $t = 30s$ **- changement de formation**



(e) $t = 125s$ **- suivi de trajectoire**



(f) $t = 200s$

Figure 6.41 – Un exemple de l'évolution de trois BlueROV-1 sur ROS/Gazebo

6.3.4 Conclusion sur l'utilisation de l'architecture RBF

Nous avons étudié l'utilisation de réseau RBF pour concevoir avec apprentissage en ligne un contrôleur qui compense la partie non linéaire du modèle de robot sous-marin. De plus, nous avons également introduit une architecture de contrôle complète pour aider le groupe de robots sous-marin à se former, suivre la trajectoire, changer la distance entre les robots sous-marins pour traverser une zone étroite, tout en contrôlant qu'ils évitent les collisions entre eux.

6.4 Conclusion du chapitre

Dans ce chapitre, nous avons considéré l'équation d'un robot du groupe comme étant non linéaire avec des paramètres dynamiques inconnus et un bruit modélisant l'influence de l'environnement. L'application de réseau neuronal pour résoudre les problèmes liés à la

stabilité du contrôleur a été étudiée. Nous avons réalisé et terminé la phase 2 du modèle d'étude proposé au chapitre 3.

7 Expérimentation pour modèle robot sous-marin réel

Ce chapitre présente un ensemble d'expérimentations et de preuves de concept réalisées dans un contexte sanitaire où le bassin d'essais était très peu disponible. Il présente les tests d'algorithmes de contrôle pour 3 robots BlueROV avec positionnements relatifs. En effet, les robots ne sont pas équipés de dispositifs de positionnement tels que transpondeurs et modems acoustiques. C'est pourquoi nous choisirons une approche basée sur la vision pour obtenir la position relative entre les robots (dans ce cas la position relative entre le suiveur et le leader du robot). Cependant, avec l'utilisation de la méthode basée sur la vision dans l'environnement sous-marin, la position relative obtenue est interrompue car le leader du robot est hors de la vue de la caméra ou la caméra sur le suiveur de robot n'a pas suffisamment de points de référence sur le robot-leader. Pour cette raison, nous utilisons un filtre de Kalman étendu pour combiner le signal IMU (en anglais *Inertial Measurement Unit*) et Caméra pour améliorer les résultats obtenus.

7.1 Filtre de Kalman étendu et Vision-based

7.1.1 Filtre de Kalman étendu

Le filtre de Kalman étendu est l'un des algorithmes d'estimation les plus importants et les plus courants. Le filtre de Kalman produit des estimations de variables cachées basées sur des mesures inexactes et incertaines. De plus, le filtre de Kalman fournit une prédiction de l'état futur du système, basée sur les estimations passées. Les notations standards sont utilisées pour le filtre de Kalman, et sont listées sur le tableau 7.1. Ces symboles sont couramment utilisés lorsqu'on travaille avec des filtres Kalman. Par conséquent, il est nécessaire d'éviter toute confusion avec les symboles des chapitres précédents.

Le processus peut être décrit comme un système dynamique non linéaire, avec

$$\mathbf{x}_n = \mathbf{f}_{kf}\left(\mathbf{x}_{n-1}\right) + \mathbf{w}_{n-1} \tag{7.1}$$

Tableau 7.1 – Les notations standards sont utilisées pour le filtre de Kalman étendu

Notation	Nom	Dimensions
$\mathbf{x}$	Vecteur d'État	$n_x \times 1$
$\mathbf{z}$	Vecteur de sortie	$n_z \times 1$
F	Matrice de transition des États	$n_x \times n_x$
$\mathbf{u}$	Variable d'entrée	$n_u \times 1$
G	Matrice de contrôle	$n_x \times n_u$
P	Estimation de l'incertitude	$n_x \times n_x$
Q	Incertitude sur le bruit des processus	$n_x \times n_x$
R	Incertitude des mesures	$n_z \times n_z$
$\mathbf{w}$	Vecteur de bruit de processus	$n_x \times 1$
$\mathbf{v}$	Vecteur de bruit de mesure	$n_z \times 1$
H	Matrice d'observation	$n_z \times n_x$
K	Kalman Gain	$n_x \times n_z$
n	Indice de temps discret	–

où $\mathbf{x}_n$ est l'état du système du robot (c'est-à-dire la pose 3D) au temps n, $\mathbf{f}_{kf}$ est une fonction de transition d'état non linéaire, et $\mathbf{w}_{n-1}$ est le bruit du processus, qui est supposé être normalement distribué. En outre, la forme des mesures est

$$\mathbf{z}_n = h\left(\mathbf{x}_n\right) + \mathbf{v}_n \tag{7.2}$$

Le filtre de Kalman fonctionne en "boucle prédictive" (voir la figure 7.1). Une fois initialisé, le filtre de Kalman doit prédire l'état du système au prochain pas de temps. De plus, le filtre de Kalman fournit l'incertitude de la prédiction.

$$\hat{\mathbf{x}}_{n+1,n} = F\hat{\mathbf{x}}_{n,n} + G\hat{\mathbf{u}}_{n,n} \tag{7.3}$$

$$P_{n+1,n} = FP_{n,n}F^T + Q \tag{7.4}$$

Une fois la mesure reçue, le filtre de Kalman met à jour (ou corrige la prédiction) et l'incertitude de l'état actuel. De plus, le filtre de Kalman prédit les états suivants, et ainsi de suite.

$$K_n = P_{n,n-1}H^\top\left(HP_{n,n-1}H^\top + R_n\right)^{-1} \tag{7.5}$$

$$\hat{\mathbf{x}}_{n,n} = \hat{\mathbf{x}}_{n,n-1} + K_n\left(\mathbf{z}_n - H\hat{\mathbf{x}}_{n,n-1}\right) \tag{7.6}$$

$$P_{n,n} = (I - K_n H)\, P_{n,n-1} (I - K_n H)^\top + K_n R_n K_n^\top \tag{7.7}$$

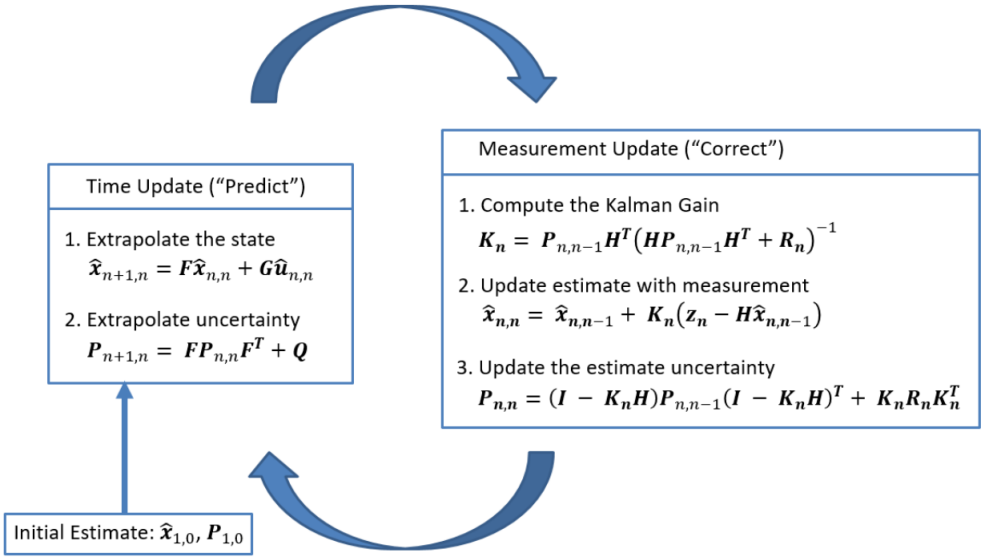


Figure 7.1 – Une structure complète de l'opération de filtre de Kalman

7.1.2 Théorie de l'estimation de la position basée sur la vision

Dans cette section, nous supposons qu'un modèle 3D de la scène est disponible ou peut être estimé en ligne. La position doit être estimée en connaissant les correspondances entre les mesures 2D dans les images et les caractéristiques 3D du modèle.

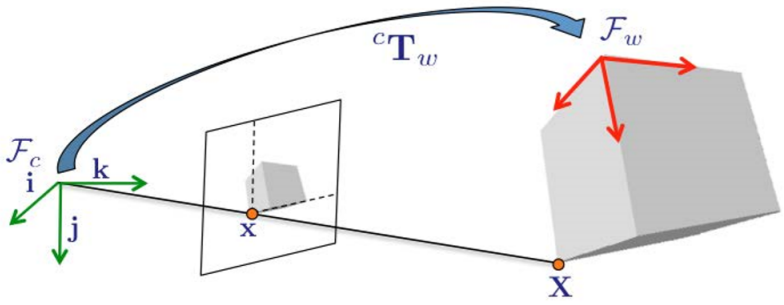


Figure 7.2 – Transformation rigide ${}^c\mathbf{T}_w$ entre l'image du monde F_w et l'image de la caméra F_c et la projection en perspective [125]

Appelons F_c le cadre de la caméra et ${}^c\mathbf{T}_w$ la transformation qui définit la position de F_w par rapport à F_c.

$${}^c\mathbf{T}_w = \begin{pmatrix} {}^c\mathbf{R}_w & {}^c\mathbf{t}_w \\ \mathbf{0}_{3\times 1} & 1 \end{pmatrix} \tag{7.8}$$

où ${}^{c}R_{\mathrm{w}}$ et ${}^{c}t_{\mathrm{w}}$ sont la matrice de rotation et le vecteur de translation que définit la position de la caméra dans un repère global.

La projection en perspective $\bar{\mathbf{x}} = (u, v, 1)^{\top}$ d'un point ${}^{w}\mathbf{X} = ({}^{w}X, {}^{w}Y, {}^{w}Z, 1)^{\top}$ sera donnée par

$$\bar{\mathbf{x}} = \mathbf{K}\Pi {}^{c}\mathbf{T}_{w} {}^{w}\mathbf{X} \tag{7.9}$$

où $\bar{\mathbf{x}}$ est la projection en perspective d'un point $\mathbf{X}$, $\mathbf{K}$ est la matrice des paramètres intrinsèques de la caméra.

$$\mathbf{K} = \begin{pmatrix} p_x & 0 & u_0 \\ 0 & p_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7.10}$$

$$\Pi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{7.11}$$

Les paramètres intrinsèques peuvent être facilement obtenus par une étape de calibration hors ligne. Par conséquent, nous allons considérer les coordonnées de l'image exprimées dans l'espace métrique normalisé

$$\mathbf{x} = \mathbf{K}^{-1}\bar{\mathbf{x}} \tag{7.12}$$

Si nous avons N points ${}^{w}\mathbf{X}_i, i = 1..N$ dont les coordonnées exprimées en F_w sont donnés par ${}^{w}\mathbf{X}_i = ({}^{w}X_i, {}^{w}Y_i, {}^{w}Z_i, 1)^{\top}$, la projection $\bar{\mathbf{x}}_i = (x_i, y_i, 1)^{\top}$ de ces points dans le plan de l'image est alors donnée par :

$$\mathbf{x}_i = \Pi {}^{c}\mathbf{T}_{w} {}^{w}\mathbf{X}_i \tag{7.13}$$

Connaître les correspondances de points 2D-3D, $\mathbf{x}_i$ et ${}^{w}\mathbf{X}_i$, l'estimation de la pose consiste à résoudre le système donné par l'ensemble des équations 7.13 pour ${}^{c}\mathbf{T}_{w}$. Il s'agit d'un problème inverse connu sous le nom de problème de perspective à partir de N points ou PnP (*Perspective-n-point*) [125], [126].

7.2 Mise en oeuvre

Dans cette section, nous présenterons l'utilisation du filtre EKF pour combiner le signal de caméra reçu et le signal IMU dans le but de prédire la position du robot Leader au cas où la caméra montée sur le robot suiveur ne peut pas détecter 4 Leds montées sur le robot leader.

Hypothèse 1. *Nous utilisions 3 types BlueROV : BlueROV-1, BlueROV-2, BlueROV-Heavy.*

Cependant, dans ce scénario de test, nous supposons que les robots sous-marins BlueROV ont le même modèle dynamique.

Hypothèse 2. *Les algorithmes qui ont été étudiés dans les chapitres 5 et 6 ne distinguent pas le robot sous-marin leader et le suiveur. Cependant, dans ces conditions de test, nous devons choisir 1 robot leader et 2 robots suiveurs pour pouvoir utiliser l'approche caméra pour détecter 4 LED, qui sont montées sur le robot sous-marin leader.*

En utilisant la méthode PnP [125], [126], on obtient la position relative de 4 Leds par rapport à la caméra, ou la position relative du robot sous-marin Leader par rapport le robot sous-marin suiveur (voir la figure 7.3).

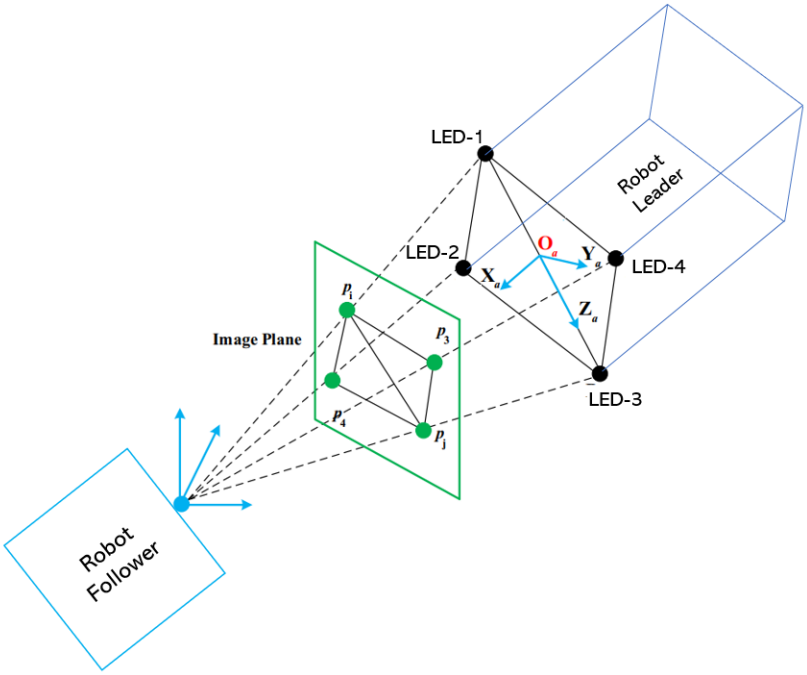


Figure 7.3 – Déterminer la position relative de 4 points à l'aide de la caméra

Ensuite, en combinant la position relative obtenue par la caméra avec le signal mesuré par le capteur via le filtre EKF, nous obtenons la position relative entre le robot sous-marin Follower et le Leader. Les détails de l'utilisation du filtre EKF sont présentés dans la figure 7.4.

Dans ce test, les robots ont 4 degrés de liberté. Il convient de noter que lors des tests, il n'y a pas eu de communication entre les robots. Les câbles collectent et transmettent uniquement les données de contrôle à partir d'ordinateurs situés à terre.

Nous avons effectué le test à la piscine SeaTech de l'Université de Toulon. Longueur de nage x largeur x profondeur = 10 mx 3 mx 1,5 m. La taille des 4 LED équipées sur le leader du robot est longueur x largeur = 0,42 mx 0,24 m (comme indiqué sur la figure 7.5).

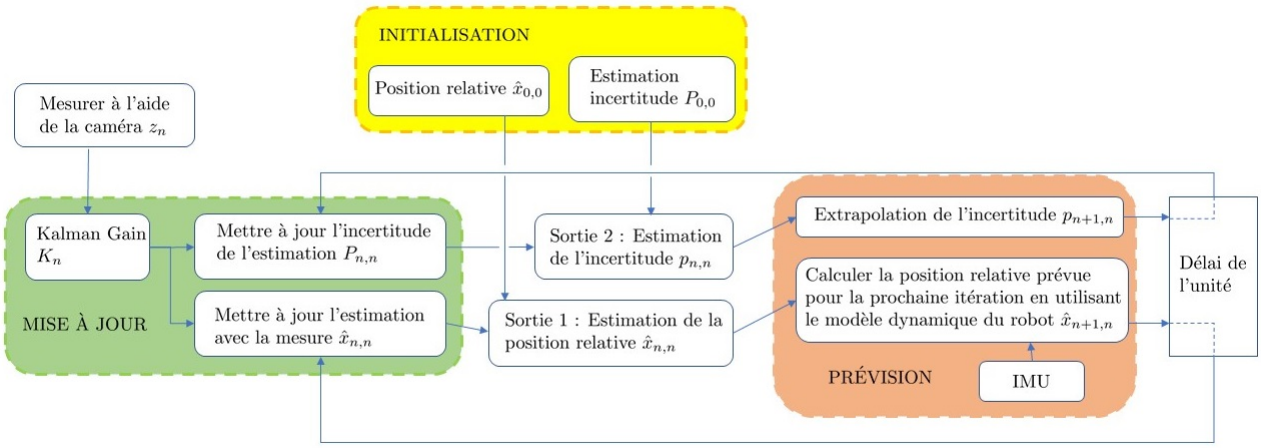


Figure 7.4 – Schéma de principe de la combinaison de la caméra et de l'IMU à l'aide du filtre KF

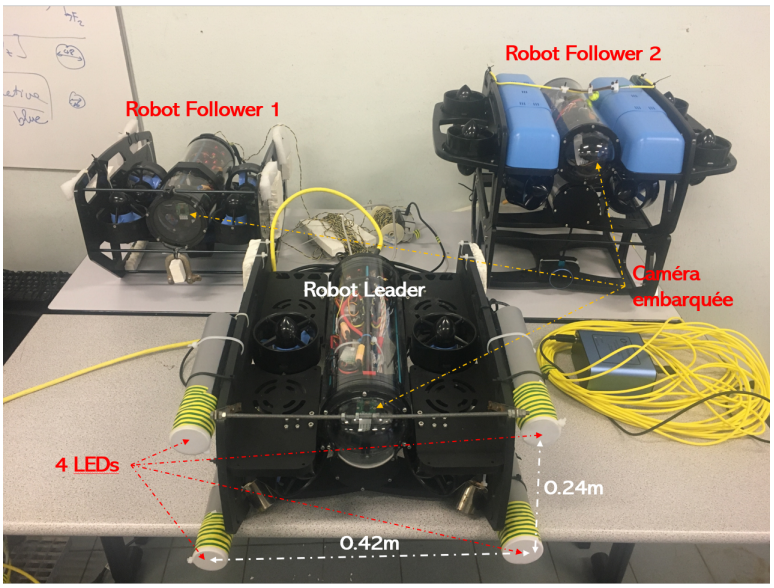


Figure 7.5 – Trois robots BlueROV

Les trois robots sont placés dans une position arbitraire au niveau de la piscine et doivent remplir la condition initiale que le chef de robot doit être dans la zone visible des deux caméras situées sur le suiveur de robot. Le robot suiveur doit détecter 4 LED placées sur le leader du robot puis calculer la position relative entre la caméra et 4 LED. Sans perte de généralité, on suppose que la position relative entre le robot suiveur et le robot leader est également la position relative entre la caméra robot suiveur et les 4 Leds montées sur le robot leader.

7.3 Résultats

Nous avons uniquement eu le temps d'expérimenter la mise en formation et le suivi. La position relative initiale entre le robot suiveur BlueROV-1 et le robot BlueROV leader est $(x, y) = (1.6m, 0.5m)$. La position relative initiale entre le robot suiveur BlueROV-2 et le robot BlueROV leader est $(x, y) = (1.6m, -0.5m)$. Le but est que les robots suiveurs 1 et 2 doivent se déplacer automatiquement pour assurer la position relative entre eux égales à $= 1, 2m$. Ensuite, le robot leader se déplace selon une trajectoire donnée (dans ce cas, il est contrôlé via un joystick. Cependant, il est également possible d'adopter une approche basée sur la vision dans une application de suivi de pipeline [127]). Dans le même temps, les robots suiveurs 1 et 2 doivent se déplacer et toujours assurer leurs distances entre eux et le robot leader égales à $1.2m$. De plus, nous intégrons également un contrôleur de profondeur (PD) pour nous assurer que les trois robots ont toujours la même profondeur pendant l'expérience.

Les figures 7.6, 7.7 montrent la distance entre le robot suiveur 1 et 2 par rapport le robot leader. Nous constatons que ces distances convergent toutes vers $1, 2m$. Cela peut prouver que les robots suiveurs peuvent garder une distance constante du robot leader.

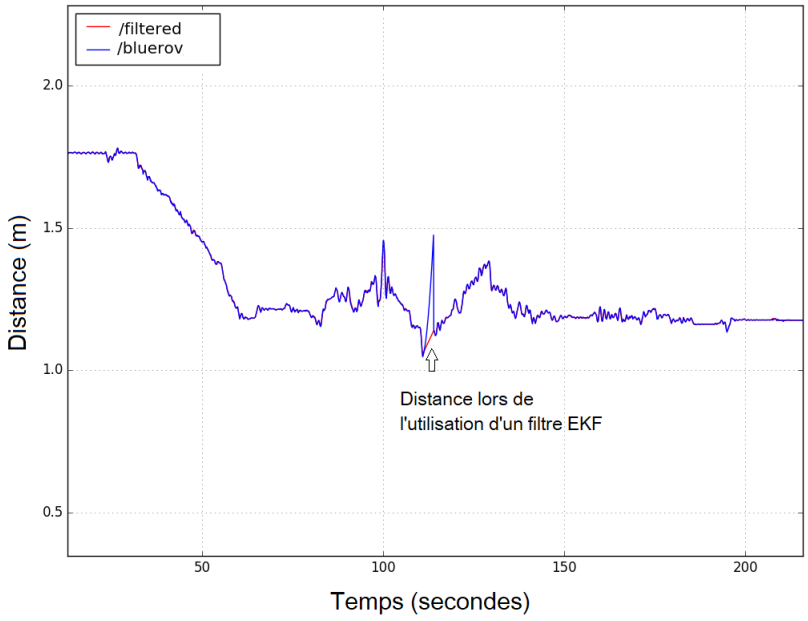


Figure 7.6 – Distance entre robot BlueROV suiveur-1 et robot BlueROV leader. (***La ligne rouge indique la valeur de la distance entre le suiveur du robot et le leader du robot après avoir utilisé le filtre EKF pour combiner les valeurs de vision et d'IMU. La ligne bleue montre les valeurs de distance entre deux robots lorsque le filtre EKF n'est pas utilisé.***)

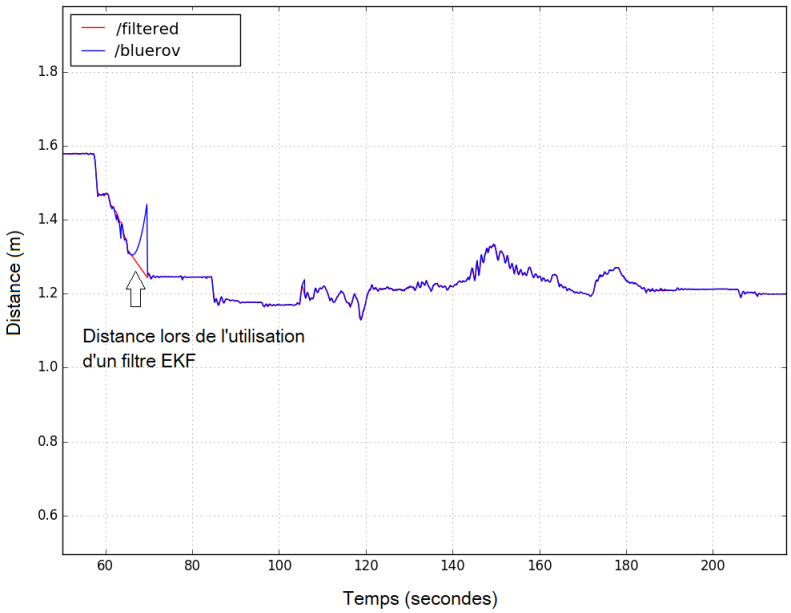


Figure 7.7 – Distance entre robot BlueROV suiveur-2 et robot BlueROV leader. (*La ligne rouge indique la valeur de la distance entre le suiveur du robot et le leader du robot après avoir utilisé le filtre EKF pour combiner les valeurs de vision et d'IMU. La ligne bleue montre les valeurs de distance entre deux robots lorsque le filtre EKF n'est pas utilisé.*)

Les figures 7.8, 7.9 montrent les signaux de commande du robot suiveur 1 et 2.

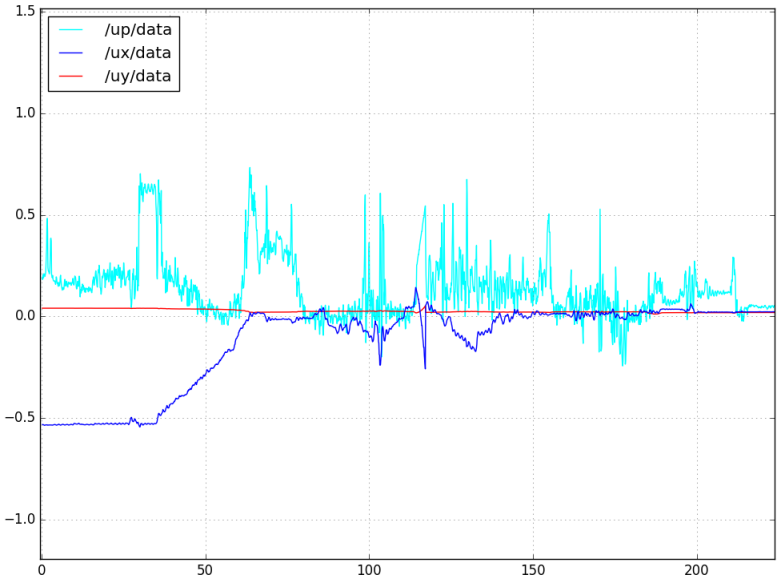


Figure 7.8 – Signaux de commande selon 3 axes du robot BlueROV suiveur-1. (*Une ligne bleue, rouge et bleu ciel représente un signal de commande sur l'axe des x; axe y et signal de commande d'angle correspondant.*)

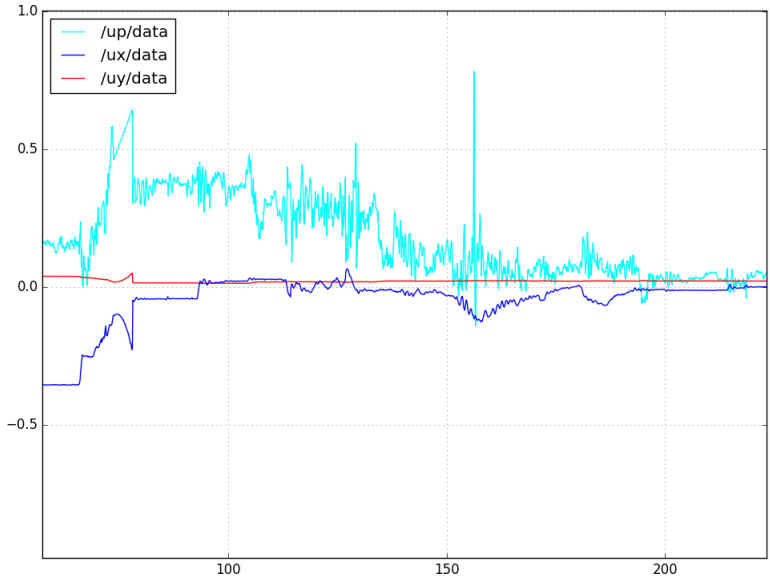


Figure 7.9 – Signaux de commande selon 3 axes du robot BlueROV suiveur-2. (***Une ligne bleue, rouge et bleu ciel représente un signal de commande sur l'axe des x; axe y et signal de commande d'angle correspondant.***)

La figure 7.10 montre le test de 3 robots sous-marins BLueROVs au bassin de Seatech (Université de Toulon). L'objectif était la formation des robots suiveurs BlueROV à une distance de $1,2m$ entre eux par rapport le robot leader BlueROV. Puis, le robot leader se déplace et les robots suiveurs doivent suivre le robot leader. De plus, ils doivent encore maintenir une distance de $= 1.2m$ pour le robot leader. La figure 7.10 $a)$ montre la position initiale des robots sous-marins BlueROV. La figure 7.10 $b)$ montre les robots sous-marins suiveur BLueROVs déplacent en formation. Les figures 7.10 $c)$ et $d)$ montrent que le robot leader en mouvement, parallèlement à cela, les deux robots suiveurs suivront le robot leader et maintiendront une distance de $1.2m$.

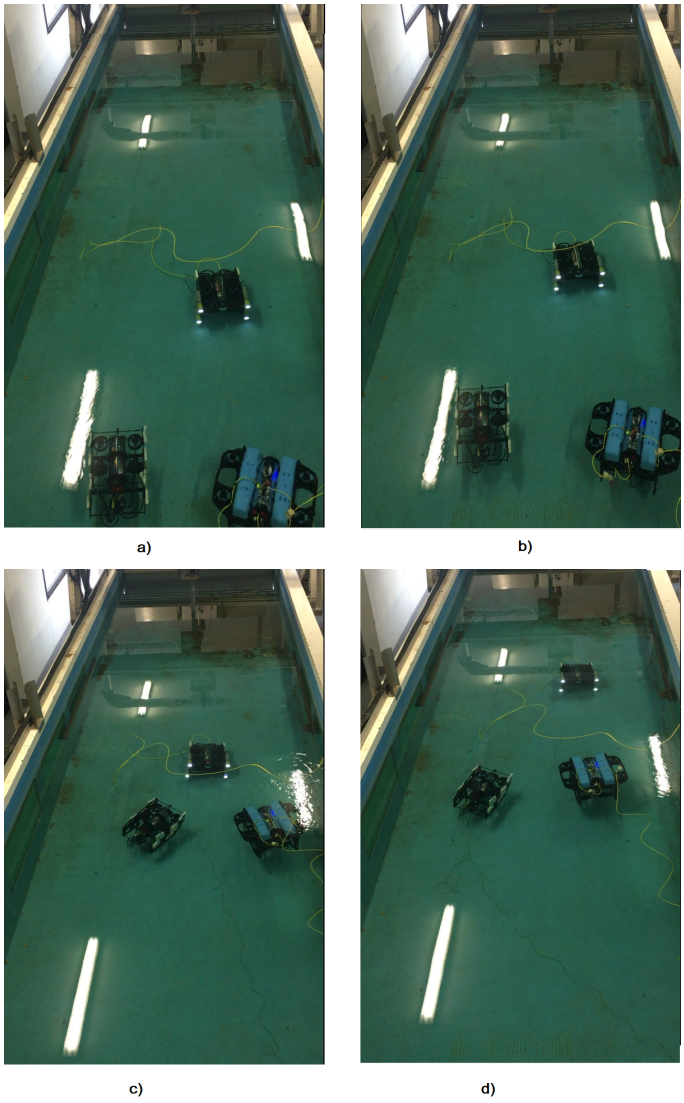


Figure 7.10 – Le test de 3 robots a formé une formation et s'est déplacé selon le robot leader

La figure 7.11 montre des photos des 2 robots sous-marins BlueROV suiveurs, qui est vue depuis la caméra montée sur le robot leader.

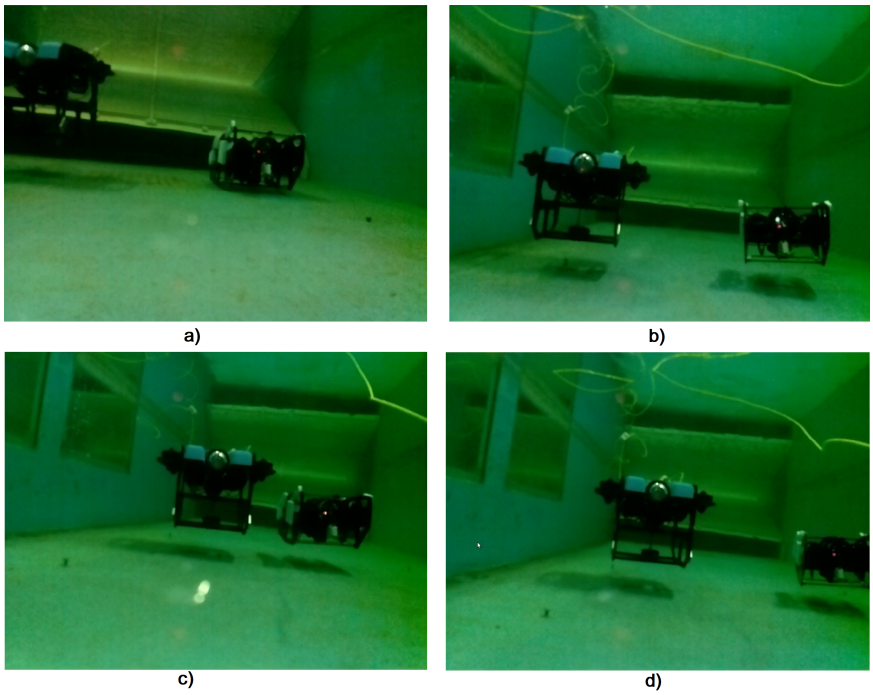


Figure 7.11 – Les deux robots suiveurs sont vus depuis la caméra du robot leader

La Figure 7.12 montre l'image de 4 LED détectées sur le robot Leader depuis le robot BlueROV suiveur 1 et 2 respectivement.

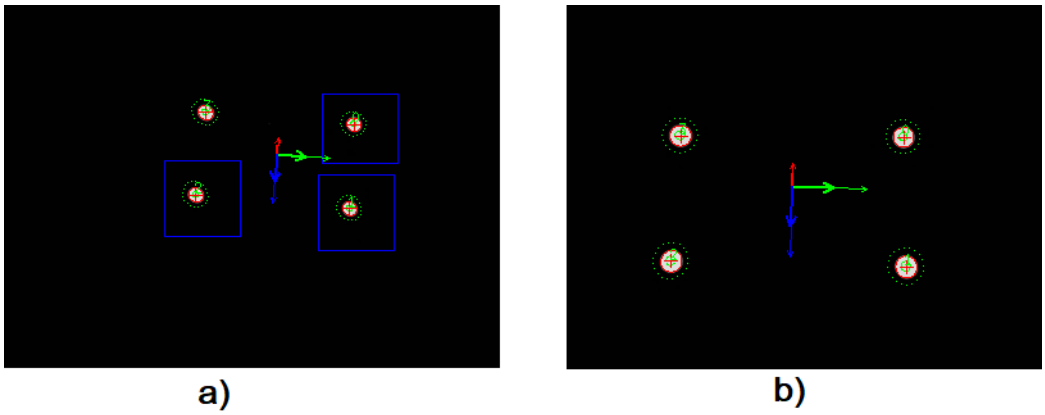


Figure 7.12 – Les deux robots suiveurs regardent le robot leader

La figure 7.13 présente une structure des nœuds ROS pour un robot follower i. Par rapport à l'architecture simulée sur Gazebo (Figure 6.34), ici, nous avons développé des nœuds supplémentaires pour pouvoir détecter 4 LED du robot leader ; ainsi que l'utilisation de filtres EKF supplémentaires.

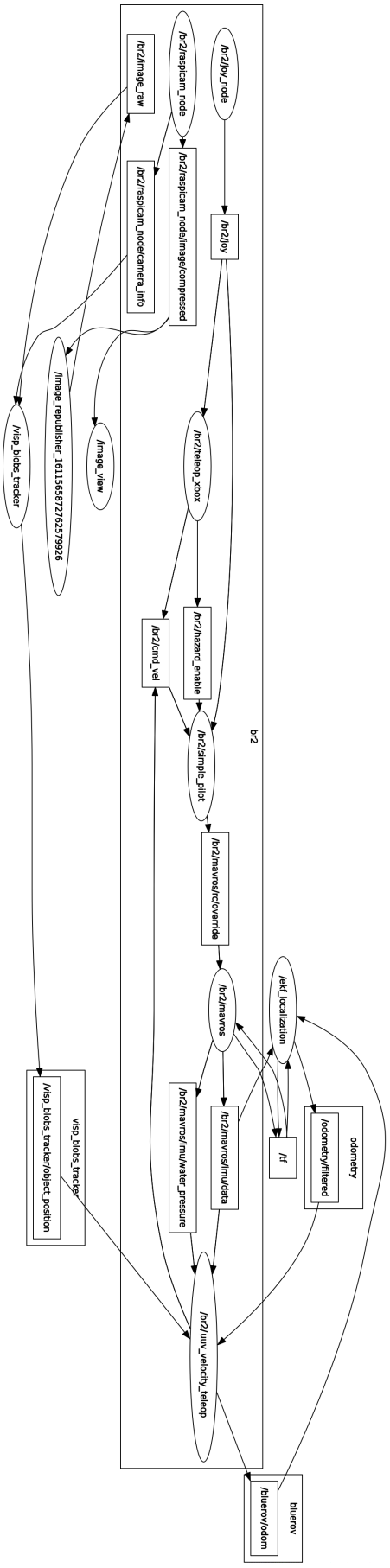


Figure 7.13 – Structure des nœuds ROS pour un robot follower *i*

7.4 Conclusion du chapitre

Les robots suiveurs peuvent suivre le robot leader ; se mettre en formation et maintenir une distance relative entre eux. Ceci démontre le fonctionnement des algorithmes de contrôle présentés et étudiés dans les chapitres 6 et 7. Ces approches conviennent aux applications où un espacement entre petits robots est nécessaire.

Dans ce cas de test, nous devons choisir un robot leader et deux robots suiveurs en raison de la limitation des dispositifs de positionnement pour les robots sous-marins et pour s'adapter à l'utilisation de la caméra.

De plus, les problèmes de robustesse doivent être étudiés davantage pour augmenter la précision. La position relative obtenue entre les robots est entièrement due à la combinaison de la caméra et de l'IMU utilisant le filtre EKF. Cependant, la nature de l'utilisation de caméras pour localiser sous l'eau présente encore de nombreux défis en matière d'erreur, de détection d'objets (dans ce cas 4 LEDs), ainsi que de changements d'intensité de la lumière de l'environnement sous-marin.

En raison du temps limité de travail dans la piscine durant tout année 2020 (en raison de l'état de sanitaire actuel), nous n'avons pas été en mesure de procéder à des évaluations d'évitement de collisions entre robots sous-marins ni à des tests d'évitement des obstacles en se déplaçant dans la trajectoire du groupe de robots sous-marins. De plus, les évaluations de l'utilisation des réseaux RBF pour exclure les paramètres non linéaires du modèle de dynamique du robot sous-marin n'ont pas été réalisées dans ces tests.

À l'avenir, il est nécessaire d'utiliser d'autres méthodes de positionnement pour comparer et référencer les résultats obtenus à partir de la caméra.

8 Conclusion générale

8.1 Conclusion

Dans ce manuscrit, nous avons abordé le thème de la coordination d'un groupe de robot sous-marins autonomes basée sur une méthodologie intégrée dans un environnement Open-source. En recherche d'une méthodologie intégrée, nous avons créé un framework qui permet de concevoir et simuler des commandes de robots sous-marins low-cost avec différentes hypothèses de modèle de complexité croissante (linéaire, non-linéaire, et enfin non-linéaire avec des incertitudes). Sur la base de ce framework articulant plusieurs outils, nous avons étudié des algorithmes pour résoudre le problème de la mise en formation d'un essaim, puis celui de l'évitement de collisions entre robots et celui du contournement d'obstacle d'un groupe de robots sous-marins. Plus précisément, nous considérons d'abord les modèles de robot sous-marin comme des systèmes linéaires de type simple intégrateur, à partir duquel nous pouvons construire un contrôleur de mise en formation en utilisant des algorithmes de consensus et d'évitement. Nous élargissons ensuite ces algorithmes pour le modèle dynamique non linéaire d'un robot Bluerov dans un processus de conception itératif. Nous intégrons ensuite un réseau de neurones de type RBF (Radial Basis Function), déjà éprouvé en convergence et stabilité, avec le contrôleur algébrique pour pouvoir estimer et compenser des incertitudes du modèle du robot. Enfin, nous décrivons les tests de ces algorithmes sur un essaim de robots sous-marins réels BlueROV en environnement Opensource de type ROS et programmés en mode autonome. Ce travail permet également de convertir un ROV téléopéré en un hybride ROV-AUV autonome. Nous présentons des résultats de simulation et des essais réels en bassin validant les concepts proposés.

8.2 Perspective

A l'avenir, il est nécessaire d'étudier la robustesse pour augmenter la stabilité du groupe de robots sous-marin.

Il faudra améliorer les algorithmes de traitement d'image dans les environnements sous-marins, où l'intensité lumineuse est inégale et varie avec la profondeur, ainsi que les effets de la diffraction de la lumière dans les environnements aquatiques.

En outre, il sera nécessaire d'étudier l'optimisation, qui traite la vitesse de mis en formation du groupe de robots sous-marins.

Il faudra étudier l'intégration de programmes de contrôle sur un ordinateur embarqué, qui sont placés directement sur des robots sous-marins.

La disponibilité future d'équipements de positionnement à bas coût permettra d'implémenter les algorithmes exacts des trois prototypes virtuels sans nécessiter de formation leader-followers.

L'approche AADL basée sur OSATE permettra de mesurer finement les temps de latence des modules correspondants aux algorithmes, et également de prévoir les conséquences de certaines pannes d'équipement.

A

Algorithme d'apprentissage pour les réseaux de neurones

A.1 Méthode du gradient

Dans cette section, nous utilisons la méthode du gradient et deux fonctions de coût différentes pour élaborer des lois adaptatives pour estimer θ^* dans le modèle paramétrique.

$$z = W(s)\,\theta^{*\top}\psi \tag{A.1}$$

L'utilisation de la méthode du gradient implique le développement d'une équation d'erreur d'estimation algébrique qui motive le choix d'une fonction de coût appropriée $J(\theta)$ qui est convexe sur l'espace de $\theta(t)$, l'estimation de θ^* au temps t, pour chaque temps t. La fonction $J(\theta)$ est alors minimisée par rapport à θ pour chaque fois t en utilisant la méthode du gradient (*The method of Steepest Descent*). La méthode de descente la plus raide procède d'une approximation initiale θ_0 pour le minimum θ^* à des points successifs $\theta_1, \theta_2, ...$ en R^n de manière itérative jusqu'à ce qu'une condition d'arrêt soit remplie. Compte tenu du point actuel θ_k, le point θ_{k+1} est obtenu par une recherche linéaire dans la direction d_k où

$$d_k = -\nabla J(\theta_k) \tag{A.2}$$

d_k est la direction à partir de θ_k dans laquelle le taux initial de diminution de $J(\theta)$ est le plus élevé. Par conséquent, la séquence $\{\theta_k\}$ est définie par

$$\theta_{k+1} = \theta_k + \lambda_k d_k = \theta_k - \lambda_k \nabla J(\theta_k), \qquad (k = 0, 1, 2, \cdots) \tag{A.3}$$

En considérant des longueurs de pas infiniment petites A.3, on peut les convertir en équation différentielle en temps continu :

$$\dot{\theta} = -\nabla J(\theta(t)), \qquad \theta(t_0) = \theta_0 \tag{A.4}$$

A.2 Méthode de projection du gradient

Dans la méthode de la descente la plus raide (méthode du gradient), la recherche du minimum de la fonction $J(\theta)$ a été effectuée pour tous les $\theta \in \mathcal{R}^n$. Dans certains cas, θ est contraint d'appartenir à un certain ensemble convexe.

$$S \triangleq \{\theta \in \mathcal{R}^n \,|g(\theta) \leq 0\} \tag{A.5}$$

dans $\mathcal{R}^n$. Où $g(\bullet)$ est une fonction à valeur scalaire s'il n'y a qu'une seule contrainte et une fonction à valeur vectorielle s'il y a plus d'une contrainte. Dans ce cas, la recherche du minimum est limitée à l'ensemble convexe défini par (A.5) au lieu de $\mathcal{R}^n$.

Considérons le cas simple où nous avons une contrainte d'égalité, c'est-à-dire que nous avons

$$\begin{array}{l} \text{minimiser } J(\theta) \\ \text{sujet à } g(\theta) = 0 \end{array} \tag{A.6}$$

où $g(\theta)$ est une fonction à valeur scalaire. L'une des techniques les plus courantes pour gérer les contraintes est d'utiliser une méthode de descente dans laquelle le sens de descente est choisi pour réduire la fonction $J(\theta)$ en restant dans la région contrainte. Cette méthode est généralement appelée méthode de projection du gradient.

Nous commençons par un point θ_0 satisfaisant la contrainte, c'est-à-dire, $g(\theta_0) = 0$.

Pour obtenir un vecteur amélioré θ_1, nous projetons le gradient négatif de J à θ_0, c'est-à-dire, $-\nabla J(\theta_0)$ sur le plan tangent $M(\theta_0) = \left\{\theta \in R^n \,\middle|\, \nabla g^\top(\theta_0)\,\theta = 0\right\}$ obtenir le vecteur directionnel $\Pr(\theta_0)$.

Puis θ_1 est considérée comme $\theta_0 + \lambda_0 \Pr(\theta_0)$ où λ_0 est choisi de manière à réduire au minimum $J(\theta_1)$. La forme générale de cette itération est donnée par

$$\theta_{k+1} = \theta_k + \lambda_k \Pr(\theta_k) \tag{A.7}$$

où λ_k est choisi de manière à réduire au minimum $J(\theta_k)$ et $\Pr(\theta_k)$ est le nouveau vecteur de direction après la projection $-\nabla J(\theta_k)$ sur $M(\theta_k)$. L'expression explicite pour $\Pr(\theta_k)$ peut être obtenue comme suit :

+ Le vecteur $-\nabla J(\theta_k)$ peut être exprimée sous la forme d'une combinaison linéaire du vecteur $\Pr(\theta_k)$ et le vecteur normal $N(\theta_k) = \nabla g(\theta_k)$ au plan tangent $M(\theta_k)$ à θ_k, c'est-à-dire.,

$$-\nabla J(\theta_k) = \alpha \nabla g(\theta_k) + \Pr(\theta_k) \tag{A.8}$$

$$\Rightarrow -\nabla g^\top \nabla J = \alpha \nabla g^\top \nabla g + \nabla g^\top \Pr \tag{A.9}$$

Puisque $\Pr(\theta_k)$ se trouve sur le plan tangent $M(\theta_k)$, nous avons également $\nabla g^\top \Pr = 0$

$$\Rightarrow -\nabla g^t \nabla J = \alpha \nabla g^t \nabla g \tag{A.10}$$

$$\Rightarrow \alpha = -\left(\nabla g^\top \nabla g\right)^{-1} \nabla g^\top \nabla J \tag{A.11}$$

Ainsi, à partir de (A.9) , nous obtenons

$$\Pr(\theta_k) = -\left[I - \nabla g\left(\nabla g^\top \nabla g\right)^{-1} \nabla g^\top\right] \nabla J \tag{A.12}$$

Nous nous référons à $\Pr(\theta_k)$ comme la direction projetée sur l'installation tangente $M(\theta_k)$. La méthode de projection du gradient est illustrée à la figure A.1.

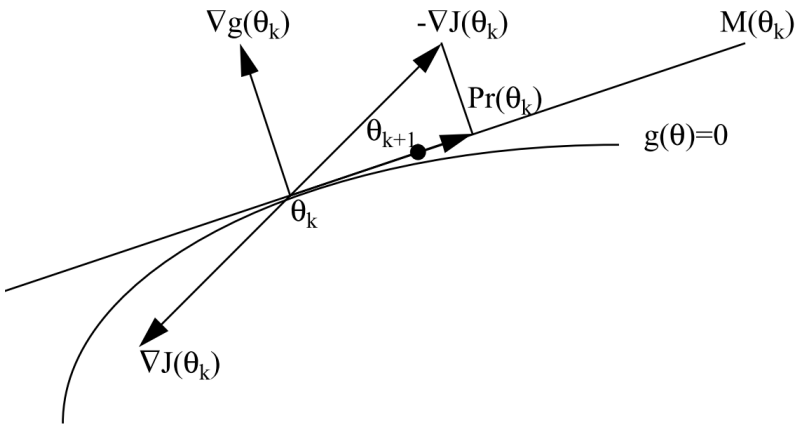


Figure A.1 – Méthode de projection de gradient

Il est évident d'après la figure A.1 que lorsque $g(\theta)$ n'est pas une fonction linéaire de θ, le nouveau vecteur θ_{k+1} donné par (A.8) peut ne pas satisfaire la contrainte, donc il doit être modifié. Il existe plusieurs techniques d'approximation successives qui peuvent être utilisées pour déplacer

$$\theta_{k+1}$$

de $M(\theta_k)$ à la surface de contrainte $g(\theta) = 0$. Un cas particulier, que l'on rencontre souvent dans les applications de contrôle adaptatif, est celui où θ est contraint de rester dans une bille avec un centre et un rayon donnés, c'est-à-dire., $g(\theta) = (\theta - \theta_0)^\top (\theta - \theta_0) - M^2$, où θ_0 est un vecteur constant fixe et $M > 0$ est un vecteur scalaire. Dans ce cas, l'algorithme de projection discrète qui garantit que $\theta_k \in S \quad \forall k$ est

$$\bar{\theta}_{k+1} = \theta_k + \lambda_k \nabla J \tag{A.13}$$

$$\theta_{k+1} = \begin{cases} \bar{\theta}_{k+1} \quad if \quad \left|\bar{\theta}_{k+1} - \theta_0\right| \leq M \\ \theta_0 + \frac{\bar{\theta}_{k+1} - \theta_0}{\left|\bar{\theta}_{k+1} - \theta_0\right|} M \quad if \quad \left|\bar{\theta}_{k+1} - \theta_0\right| \geq M \end{cases} \tag{A.14}$$

En laissant la longueur du pas λ_k devenir infiniment petite, on obtient la version en temps continu de (A.8)

$$\dot{\theta} = \Pr(\theta) = -\left[I - \nabla g\left(\nabla g^\top \nabla g\right)^{-1} \nabla g^\top\right] \nabla J \tag{A.15}$$

En raison de la longueur de pas suffisamment petite, la trajectoire $\theta(t)$, si elle existe, satisfera $g(\theta(t)) \, \forall t \geq 0$ à condition que $\theta(0) = \theta_0$ satisfasse $g(\theta_0) = 0$.

Le problème de minimisation A.6 peut maintenant être étendu à :

$$\begin{array}{l} \text{minimiser } J(\theta) \\ \text{sujet à } g(\theta) \leq 0 \end{array} \tag{A.16}$$

où $S = \{\theta \in R^n \,|g(\theta) \leq 0\}$ est un sous-ensemble convexe de R^n.

La solution à A.16 découle directement de celle du problème sans contrainte et A.6. Nous partons d'un point initial $\theta_0 \in S$. Si le point actuel se trouve à l'intérieur de S, défini comme $S_0 \triangleq \{\theta \in R^n \,|g(\theta) < 0\}$, alors l'algorithme non contraint est utilisé. Si le point actuel se trouve à la limite de S, défini comme $\delta(S) = \{\theta \in R^n \,|g(\theta) = 0\}$ et la direction de la recherche donnée par l'algorithme non contraint est orientée à l'opposé de S, alors nous utilisons l'algorithme de projection du gradient. Si la direction de la recherche pointe à l'intérieur de S, nous gardons l'algorithme sans contrainte. Compte tenu de ce qui précède, la solution au problème d'optimisation contrainte A.16 est donnée par :

$$\dot{\theta} = \begin{cases} -\nabla J(\theta) \quad \text{si} \quad \theta \in S_0 \quad \text{ou} \quad \theta \in \delta(S) \quad \text{et} \quad -\nabla J^\top \nabla g \leq 0 \\ -\nabla J(\theta) + \frac{\nabla g \nabla g^\top}{\nabla g^\top \nabla g} \nabla J(\theta) \quad \text{autrement} \end{cases} \tag{A.17}$$

Glossaire

AUV : Autonomous Underwater Vehicle
ROV : Remotely Operated underwater Vehicle
UUV : Unmanned underwater Vehicles
GPS : Global Positioning System
MBSE : Model Based System Engineering
UML : Unified Modeling Language
SysML : Systems Modeling Language
MARTE : Modeling and Analysis of Real Time and Embedded systems
PIM : Platform Independent Model
PSM : Platform Specific Model
SDLC : Software Development Life Cycle
AADL : Architecture Analysis and Design Language
EFR : Enhanced Full Rate
MDE : Model Driven Engineering
DET : Design exploration Technologies
SSML : Solution Space Modeling Language
DSML : Domain-Specific Modeling Languages
ROS : Robot Operating System
RN : Réseau Neuronal
CAMR : Contrôle avec modèle de référence
RNN : Recurrent Neural Network
RBF : Radial Basis Function Network
NARX : Nonlinear Autoregressive Exogenous

Bibliographie

[1] Intel, "Intel drone light shows," *https://www.intel.com/content/www/us/en/technology-innovation/aerial-technology-light-show.html*, 2019.

[2] Amazon, "Amazon robots warehouse," *https://spectrum.ieee.org/automaton/robotics/industrial-robots/amazon-introduces-two-new-warehouse-robots*, 2019.

[3] Aquabotix, "Aquabotix's swarmdivers," *https://www.aquabotix.com/*, 2019.

[4] D. Akdur, V. Garousi, and O. Demirörs, "A survey on modeling and model-driven engineering practices in the embedded software industry," *Journal of Systems Architecture*, vol. 91, pp. 62–82, nov 2018.

[5] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice, Second Edition*. Morgan and Claypool, 2017.

[6] S. Mallek, N. Daclin, and V. Chapurlat, "The application of interoperability requirement specification and verification to collaborative processes in industry," *Computers in Industry*, vol. 63, pp. 643–658, sep 2012.

[7] R. F. Paige, N. Matragkas, and L. M. Rose, "Evolving models in model-driven engineering : State-of-the-art and future challenges," *Journal of Systems and Software*, vol. 111, pp. 272–280, jan 2016.

[8] M. Torchiano, F. Tomassetti, F. Ricca, A. Tiso, and G. Reggio, "Relevance, benefits, and problems of software modelling and model driven techniques : A survey in the italian industry," *Journal of Systems and Software*, vol. 86, pp. 2110–2126, aug 2013.

[9] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice Second Edition*, vol. 3. Synthesis Lectures on Software Engineering, 2017.

[10] F. Cottet, E. Grolleua, S. Gérard, J. Hugues, Y. Ouhammou, and S. Tucci, *Systèmes temps réel embarqués - 2e édition : Spécification, conception, impémentation et validation temporelle*. Dunod, 2014.

[11] F. Herrera, H. Posadas, P. Peñil, E. Villar, F. Ferrero, R. Valencia, and G. Palermo, "The COMPLEX methodology for UML/MARTE modeling and design space exploration of embedded systems," *Journal of Systems Architecture*, vol. 60, pp. 55–78, jan 2014.

[12] M. Rashid, M. W. Anwar, and A. M. Khan, "Toward the tools selection in model based system engineering for embedded systems : A systematic literature review," *Journal of Systems and Software*, 2015.

[13] A. Ramaswamy, *A Model-Driven Framework Development Methodology for Robotic Systems*. PhD thesis, Université Paris-Saclay, 2017.

[14] P. C, P. G, G. R, and B. G, "Applying MDA and OMG robotic specification for developing robotic systems," in *System Analysis and Modeling (SAM), Technology-Specific Aspects of Models, Lecture Notes in Computer Science, Springer, Cham*, vol. 9959, 2016.

[15] Y. Hua, S. Zandery, M. Bordignonz, and B. Hein, "From automationML to ROS : A model-driven approach for software engineering of industrial robotics using ontological reasoning," in *IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, no. concrete, IEEE, 2016.

[16] D. Brugali, "Model-driven software engineering in robotics : Models are designed to use the relevant things, thereby reducing the complexity and cost in the field of robotics," *IEEE Robotics & Automation Magazine*, vol. 22, pp. 155–166, sep 2015.

[17] P. Trojanek, I. of Control, and C. Engineering, "Model-driven engineering approach to design and implementation of robot control system," *arXiv :1302.5085*, 2013.

[18] T. Soriano, N. Hien, K. Tuan, and T. Anh, "An object-unified approach to develop controllers for autonomous underwater vehicles," *Mechatronics*, vol. 35, pp. 54 – 70, 2016.

[19] B. Geoffrey, F. Kiyoshi, and A. Keiju, "Modelling and analysis of a redundant mobile robot architecture using AADL," in *Simulation, Modeling, and Programming for Autonomous Robots* (D. Brugali, J. F. Broenink, T. Kroeger, and B. A. MacDonald, eds.), (Cham), pp. 146–157, Springer International Publishing, 2014.

[20] S.-G. Chitic, *Middleware and programming models for multi-robot systems*. PhD thesis, Université de Lyon - L'Institut National des Sciences Appliquées de Lyon (INSA Lyon), 2018.

[21] F. Ciccozzi, D. D. Ruscio, I. Malavolta, and P. Pelliccione, "Adopting MDE for specifying and executing civilian missions of mobile multi-robot systems," *IEEE Access*, vol. 4, pp. 6451–6466, 2016.

[22] J. M. Gascueña, F. J. Garijo, A. Fernández-Caballero, M.-P. Gleizes, and A. Machonin, "Deliberative control components for eldercare robot team cooperation," *Journal of Intelligent & Fuzzy Systems*, 2015.

[23] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics : a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, pp. 1–41, jan 2013.

[24] J. M. Gascueña, E. Navarro, and A. Fernández-Caballero, "Model-driven engineering techniques for the development of multi-agent systems," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 1, pp. 159 – 173, 2012.

[25] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, "Uuv simulator : A gazebo-based package for underwater intervention and multi-robot simulation," in *OCEANS 2016 MTS/IEEE Monterey*, pp. 1–8, Sept 2016.

[26] N. Koenig, A. Howard, R. R. Labs, and U. of Southern California, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems September 28 - October 2, 2004, Sendai, Japan*, IEEE, 2004.

[27] J. Hutchinson, J. Whittle, and M. Rouncefield, "Model-driven engineering practices in industry : Social, organizational and managerial factors that lead to success or failure," *Science of Computer Programming*, vol. 89, pp. 144–161, sep 2014.

[28] A. Gokhale, K. Balasubramanian, A. S. Krishna, J. Balasubramanian, G. Edwards, G. Deng, E. Turkay, J. Parsons, and D. C. Schmidt, "Model driven middleware : A new paradigm for developing distributed real-time and embedded systems," *Science of Computer Programming*, vol. 73, pp. 39–58, sep 2008.

[29] Roadmap, "Model-driven development of complex software : A research," No. has, IEEE, 2007.

[30] G. Karsai, S. Neema, and D. Sharp, "Model-driven architecture for embedded software : A synopsis and an example," *Science of Computer Programming*, vol. 73, pp. 26–38, sep 2008.

[31] H. Espinoza, D. Cancila, B. Selic, and S. Gérard, "Challenges in combining SysML and MARTE for model-based design of embedded systems," in *Model Driven Architecture - Foundations and Applications* (R. F. Paige, A. Hartman, and A. Rensink, eds.), (Berlin, Heidelberg), pp. 98–113, Springer Berlin Heidelberg, 2009.

[32] F. Mallet and R. de Simone, "Chapter 2 MARTE vs. AADL for discrete-event and discrete-time domains," 2009.

[33] M. Faugère, T. Bourbeau, R. D. Simone, and S. Gérard, "MARTE : Also an UML profile for modeling AADL applications," in *12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*, IEEE, 2007.

[34] S. Turki, E. Senn, and D. Blouin, "Mapping the MARTE and UML profile to AADL," in *MoDELS 2010 ACES-MB Workshop Proceedings*, 2010.

[35] H. Andersson, *Aircraft Systems and Modeling : Model Based Systems Engineering in Avionics Design and Aircraft Simulation*. PhD thesis, Linkoping University, Sweden, 2009.

[36] H. Andersson, E. Herzog, G. Johansson, and O. Johansson, "Experience from introducing unified modeling language/systems modeling language at saab aerosystems," *Systems Engineering*, pp. n/a–n/a, 2009.

[37] H. Graves and Y. Bijan, "Using formal methods with SysML in aerospace design and engineering," *Annals of Mathematics and Artificial Intelligence*, vol. 63, pp. 53–102, sep 2011.

[38] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos : An agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, pp. 203–236, May 2004.

[39] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[40] E. Group, "H1000," *https ://www.ecagroup.com/en/solutions/autonomous-underwater-vehicle-pipeline-inspection*.

[41] S. Tech, "Rov guardian," *https ://www.subsea-tech.com/*.

[42] Hydroid, "Remus 100-auv," *https ://www.hydroid.com/*.

[43] E. Group, "A9-e et a18d," *https ://www.ecagroup.com/en/solutions/autonomous-underwater-vehicle-pipeline-inspection*.

[44] Aquabotix, "Aquabotix hybride auv-rov," *https ://www.aquabotix.com/*.

[45] A. D. Bowen and D. R. Y. et al., "The nereus hybrid underwater robotic vehicle for global ocean science operations to 11,000m depth," in *Oceans*, IEEE, 2008.

[46] Saab, "Double eagle mkii/mkiii," *https ://www.saab.com/products/double-eagle*.

[47] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV navigation and localization : A review," *IEEE Journal of Oceanic Engineering*, vol. 39, pp. 131–149, jan 2014.

[48] L.-H. Nguyen, M.-D. Hua, G. Allibert, and T. Hamel, "A homography-based dynamic control approach applied to station keeping of autonomous underwater vehicles without linear velocity measurements," *IEEE Transactions on Control Systems Technology*, 2020.

[49] F. Shkurti, W.-D. Chang, P. Henderson, M. J. Islam, J. C. G. Higuera, J. Li, T. Manderson, A. Xu, G. Dudek, and J. Sattar, "Underwater multi-robot convoying using visual tracking by detection," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) September 24–28, 2017, Vancouver, BC, Canada*, IEEE, 2017.

[50] H. Yamaguchi, T. Arai, and G. Beni, "A distributed control scheme for multiple robotic vehicles to make group formations," *Robotics and Autonomous Systems*, vol. 36, no. 4, pp. 125 – 147, 2001.

[51] Z. Sun, M.-C. Park, B. D. Anderson, and H.-S. Ahn, "Distributed stabilization control of rigid formations with prescribed orientation," *Automatica*, vol. 78, pp. 250–257, apr 2017.

[52] J. T. Feddema, C. Lewis, and D. A. Schoenwald, "Decentralized control of cooperative robotic vehicles : theory and application," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 852–864, Oct 2002.

[53] S. B. Saad, *Conception d'un algorithme de coordination hybride de groupes de robots sous-marins communicants. Application : acquisition optique systematique et detaillee des fonds marins*. PhD thesis, Université de Bretagne Occidentale, France, 2016.

[54] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, mar 2015.

[55] Z. Jia, L. Wang, J. Yu, and X. Ai, "Distributed adaptive neural networks leader-following formation control for quadrotors with directed switching topologies," *ISA Transactions*, mar 2019.

[56] C.-W. Kuo, C.-C. Tsai, and C.-T. Lee, "Intelligent leader-following consensus formation control using recurrent neural networks for small-size unmanned helicopters," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, pp. 1–14, 2019.

[57] D. Wang, Q. Zong, B. Tian, S. Shao, X. Zhang, and X. Zhao, "Neural network disturbance observer-based distributed finite-time formation tracking control for multiple unmanned helicopters," *ISA Transactions*, vol. 73, pp. 208–226, feb 2018.

[58] E. Montijano, D. Zhou, M. Schwager, and C. Sagues, "Distributed formation control without a global reference frame," in *2014 American Control Conference (ACC) June 4-6, 2014. Portland, Oregon, USA*, 2014.

[59] W. Yu, W. Ren, W. X. Zheng, G. Chen, and J. Lü, "Distributed control gains design for consensus in multi-agent systems with second-order nonlinear dynamics," *Automatica*, vol. 49, pp. 2107–2115, jul 2013.

[60] Q. Yang, M. Cao, H. G. de Marina, H. Fang, and J. Chen, "Distributed formation tracking using local coordinate systems," *Systems & Control Letters*, vol. 111, pp. 70–78, jan 2018.

[61] M. Deghat, B. D. O. Anderson, and Z. Lin, "Combined flocking and distance-based shape control of multi-agent formations," *IEEE Transactions on Automatic Control*, vol. 61, pp. 1824–1837, July 2016.

[62] E. Montijano, E. Cristofalo, D. Zhou, M. Schwager, and C. Sagues, "Vision-based distributed formation control without an external positioning system," *IEEE Transactions on Robotics*, vol. 32, pp. 339–351, apr 2016.

[63] Wei Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, 2005.

[64] W. Ren, R. W. Beard, and T. W. McLain, *Coordination Variables and Consensus Building in Multiple Vehicle Systems*, pp. 171–188. Berlin, Heidelberg : Springer Berlin Heidelberg, 2005.

[65] W. Ren, "Consensus seeking in multi-vehicle systems with a time-varying reference state," in *2007 American Control Conference*, pp. 717–722, 2007.

[66] W. Ren, "On consensus algorithms for double-integrator dynamics," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1503–1509, 2008.

[67] W. Ren, "Second-order consensus algorithm with extensions to switching topologies and reference models," in *2007 American Control Conference*, pp. 1431–1436, 2007.

[68] W. Ren, "Distributed attitude alignment in spacecraft formation flying," *International Journal of Adaptive Control and Signal Processing*, vol. 21, no. 2-3, pp. 95–113, 2007.

[69] W. Ren, "Distributed cooperative attitude synchronization and tracking for multiple rigid bodies," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 383–392, 2010.

[70] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, pp. 215–233, Jan 2007.

[71] R. B. Wei Ren, *Distributed Consensus in Multi-vehicle Cooperative Control Theory and Applications*. Springer-Verlag London, 2008.

[72] Z. Lin, B. Francis, and M. Maggiore, "Necessary and sufficient graphical conditions for formation control of unicycles," *IEEE Transactions on Automatic Control*, vol. 50, pp. 121–127, Jan 2005.

[73] A. Tahbaz-Salehi and A. Jadbabaie, "A necessary and sufficient condition for consensus over random networks," *IEEE Transactions on Automatic Control*, vol. 53, pp. 791–795, April 2008.

[74] Y. Zhang and Y.-P. Tian, "Consentability and protocol design of multi-agent systems with stochastic switching topology," *Automatica*, vol. 45, no. 5, pp. 1195 – 1201, 2009.

[75] Z. Qu, J. Wang, and R. A. Hull, "Cooperative control of dynamical systems with application to autonomous vehicles," *IEEE Transactions on Automatic Control*, vol. 53, pp. 894–911, May 2008.

[76] S. E. Tuna, "Conditions for synchronizability in arrays of coupled linear systems," *IEEE Transactions on Automatic Control*, vol. 54, pp. 2416–2420, Oct 2009.

[77] R. Li and G.-H. Yang, "Consensus control of a class of uncertain nonlinear multiagent systems via gradient-based algorithms," *IEEE Transactions on Cybernetics*, pp. 1–10, 2018.

[78] Keyou You, Nan Xiao, and Lihua Xie, *Analysis and Design of Networked Control Systems*. No. XII, 321 in Communications and Control Engineering, Springer-Verlag London, 1 ed., 2015.

[79] M. Jafarian and C. D. Persis, "Formation control using binary information," *Automatica*, vol. 53, pp. 125 – 135, 2015.

[80] H. Su, M. Z. Q. Chen, J. Lam, and Z. Lin, "Semi-global leader-following consensus of linear multi-agent systems with input saturation via low gain feedback," *IEEE Transactions on Circuits and Systems I : Regular Papers*, vol. 60, pp. 1881–1889, July 2013.

[81] B. Zhou, G. Duan, and Z. Lin, "A parametric lyapunov equation approach to the design of low gain feedback," *IEEE Transactions on Automatic Control*, vol. 53, pp. 1548–1554, July 2008.

[82] C. Altafini, "Consensus problems on networks with antagonistic interactions," *IEEE Transactions on Automatic Control*, vol. 58, pp. 935–946, apr 2013.

[83] K. Shojaei, "Leader-follower formation control of underactuated autonomous marine surface vehicles with limited torque," *Ocean Engineering*, vol. 105, pp. 196–205, sep 2015.

[84] H. Liu, Y. Wang, and F. L. Lewis, "Robust distributed formation controller design for a group of unmanned underwater vehicles," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, pp. 1–9, 2019.

[85] B. S. Park, "Adaptive formation control of underactuated autonomous underwater vehicles," *Ocean Engineering*, vol. 96, pp. 1–7, mar 2015.

[86] X. Yang, J. Yan, C. Hua, and X. Guan, "Trajectory tracking control of autonomous underwater vehicle with unknown parameters and external disturbances," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, pp. 1–10, 2019.

[87] Z. Peng, H. Wang, D. Wang, G. Sun, and H. Zhang, "Distributed model reference adaptive control for cooperative tracking of uncertain dynamical multi-agent systems," *IET Control Theory & Applications*, vol. 7, pp. 1079–1087, may 2013.

[88] Z. Peng, D. Wang, Z. Chen, X. Hu, and W. Lan, "Adaptive dynamic surface control for formations of autonomous surface vehicles with uncertain dynamics," *IEEE Transactions on Control Systems Technology*, vol. 21, pp. 513–520, mar 2013.

[89] X. Xu, Z. Li, and L. Gao, "Distributed adaptive tracking control for multi-agent systems with uncertain dynamics," *Nonlinear Dynamics*, vol. 90, pp. 2729–2744, oct 2017.

[90] Y. Li, C. Wang, X. Cai, L. Li, and G. Wang, "Neural-network-based distributed adaptive asymptotically consensus tracking control for nonlinear multiagent systems with input quantization and actuator faults," *Neurocomputing*, vol. 349, pp. 64–76, jul 2019.

[91] Z. Chen, Z. Li, and C. L. P. Chen, "Adaptive neural control of uncertain MIMO nonlinear systems with state and input constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 1318–1330, jun 2017.

[92] Y. Liu, P. Huang, F. Zhang, and Y. Zhao, "Distributed formation control using artificial potentials and neural network for constrained multiagent systems," *IEEE Transactions on Control Systems Technology*, pp. 1–8, 2018.

[93] C. L. P. Chen, G.-X. Wen, Y.-J. Liu, and F.-Y. Wang, "Adaptive consensus control for a class of nonlinear multiagent time-delay systems using neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, pp. 1217–1226, jun 2014.

[94] L. Chen, C. Li, Y. Sun, and G. Ma, "Distributed finite-time containment control for multiple euler-lagrange systems with communication delays," *International Journal of Robust and Nonlinear Control*, vol. 29, pp. 332–352, oct 2018.

[95] K. Chen, J. Wang, Y. Zhang, and Z. Liu, "Leader-following consensus for a class of nonlinear strick-feedback multiagent systems with state time-delays," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, pp. 1–11, 2018.

[96] H. Ma, Z. Wang, D. Wang, D. Liu, P. Yan, and Q. Wei, "Neural-network-based distributed adaptive robust control for a class of nonlinear multiagent systems with time delays and external noises," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, vol. 46, pp. 750–758, jun 2016.

[97] H. Wei, Q. Lv, N. Duo, G. Wang, and B. Liang, "Consensus algorithms based multi-robot formation control under noise and time delay conditions," *Applied Sciences*, vol. 9, p. 1004, mar 2019.

[98] B. Cui, T. Ma, Y. Song, C. Feng, F. L. Lewis, and C. Zhao, "Distributed adaptive consensus control of heterogeneous multi-agent chaotic systems with unknown time delays," *IET Control Theory & Applications*, vol. 9, pp. 2414–2422, oct 2015.

[99] C. Yuan, S. Licht, and H. He, "Formation learning control of multiple autonomous underwater vehicles with heterogeneous nonlinear uncertain dynamics," *IEEE Transactions on Cybernetics*, vol. 48, pp. 2920–2934, oct 2018.

[100] X. Qi, "Adaptive coordinated tracking control of multiple autonomous underwater vehicles," *Ocean Engineering*, vol. 91, pp. 84–90, nov 2014.

[101] C. P. Bechlioulis, F. Giagkas, G. C. Karras, and K. J. Kyriakopoulos, "Robust formation control for multiple underwater vehicles," *Frontiers in Robotics and AI*, vol. 6, sep 2019.

[102] P. C. Abreu and A. M. Pascoal, "Formation control in the scope of the MORPH project. part i : Theoretical foundations," *IFAC-PapersOnLine*, vol. 48, no. 2, pp. 244–249, 2015.

[103] P. C. Abreu, M. Bayat, A. M. Pascoal, J. Botelho, P. Góis, J. Ribeiro, M. Ribeiro, M. Rufino, L. Sebastião, and H. Silva, "Formation control in the scope of the MORPH project.

part II : Implementation and results," *IFAC-PapersOnLine*, vol. 48, no. 2, pp. 250–255, 2015.

[104] J. Bosch, N. Gracias, P. Ridao, K. Istenic, and D. Ribas, "Close-range tracking of underwater vehicles using light beacons," *Sensors*, 2016.

[105] J. Yu, J. Ji, Z. Miao, and J. Zhou, "Neural network-based region reaching formation control for multi-robot systems in obstacle environment," *Neurocomputing*, vol. 333, pp. 11–21, mar 2019.

[106] Y. Xia, X. Na, Z. Sun, and J. Chen, "Formation control and collision avoidance for multi-agent systems based on position estimation," *ISA Transactions*, vol. 61, pp. 287–296, mar 2016.

[107] A. Mondal, L. Behera, S. R. Sahoo, and A. Shukla, "A novel multi-agent formation control law with collision avoidance," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 3, pp. 558–568, 2017.

[108] J. Alonso-Mora, E. Montijano, M. Schwager, and D. Rus, "Distributed multi-robot formation control among obstacles : A geometric and optimization approach with consensus," in *2016 IEEE International Conference on Robotics and Automation (ICRA) Stockholm, Sweden, May 16-21, 2016*, IEEE, 2016.

[109] X. Li and D. Zhu, "An adaptive som neural network method to distributed formation control of a group of auvs," *IEEE Transactions on Industrial Eletronics*, 2018.

[110] G. Bardaro, A. Semprebon, and M. Matteucci, "Aadl for robotics a general approach for system architecture modeling and code generation," *Journal of Software Engineering for Robotics*, 2017.

[111] A. Semprebon, "Model based robot development : From aadl to ros through code generation," Master's thesis, Polytechnic University of Milan, Italy, 2017.

[112] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley, 2011.

[113] T. I. Fossen, *Guidance and Control of Ocean Vehicles*. John Wiley & Sons, 1994.

[114] T. I. Fossen and T. Perez, "Kalman filtering for positioning and heading control of ships and offshore rigs," in *IEEE CONTROL SYSTEMS MAGAZINE*, IEEE, 2009.

[115] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, "Uuv simulator : A gazebo-based package for underwater intervention and multi-robot simulation," in *OCEANS 2016 MTS/IEEE Monterey*, pp. 1–8, Sept 2016.

[116] A. Aili and E. Ekelund, "Model-based design and development and control and of and an underwater and vehicle," Master's thesis, Master of Science Thesis in Automatic Control Department of Electrical Engineering, Linköping University, 2016.

[117] O. Alexander and Eidsvik, *Identification of Hydrodynamic Parameters for Remotely Operated Vehicles*. PhD thesis, Norwegian University of Science and Technology, 2015.

[118] K. Jolly, R. S. Kumar, and R. Vijayakumar, "A bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits," *Robotics and Autonomous Systems*, vol. 57, pp. 23–33, jan 2009.

[119] A. H. Jafari and M. T. Hagan, "Application of new training methods for neural model reference control," *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 312–321, sep 2018.

[120] L. Briñón-Arranz, A. Seuret, and A. Pascoal, "Circular formation control for cooperative target tracking with limited information," *Journal of the Franklin Institute*, vol. 356, pp. 1771–1788, mar 2019.

[121] R. Fierro and F. L. Lewis, "Control of a nonholonomic mobile robot using neural networks," in *IEEE TRANSACTIONS ON NEURAL NETWORKS*, IEEE, 1998.

[122] P. Lancaster and L. Rodman, *Algebraic Riccati equations*. Oxford University Press, 1995.

[123] E. Lavretsky, T. E. Gibson, and A. M. Annaswamy, "Projection operator in adaptive systems," *arXiv*, 2012.

[124] S. Skaalvik and Sandøy, "System identification and state estimation for rov udrone," Master's thesis, Department of Marine Technology, Norwegian University of Science and Technology, 2016.

[125] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for augmented reality and a hands-on survey," *IEEE Trans. on Visualization and computer graphics.*, 2016.

[126] P. Wang, G. Xu, Y. Cheng, and Q. Yu, "A simple, robust and fast method for the perspective-n-point problem," *Pattern Recognition Letters*, vol. 108, pp. 31–37, jun 2018.

[127] G. Allibert, M.-D. Hua, S. Krupínski, and T. Hamel, "Pipeline following by visual servoing for autonomous underwater vehicles," *Control Engineering Practice*, vol. 82, pp. 151 – 160, 2019.

Table des figures

Coordination de systèmes sous-marins autonomes basée sur une méthodologie intégrée dans un environnement Open-source

Cette thèse étudie la coordination de robots sous-marins autonomes dans le contexte d'exploration de fonds marins côtiers ou d'inspections d'installations. En recherche d'une méthodologie intégrée, nous avons créé un framework qui permet de concevoir et simuler des commandes de robots sous-marins low-cost avec différentes hypothèses de modèle de complexité croissante. Sur la base de ce framework articulant plusieurs outils, nous avons étudié des algorithmes pour résoudre le problème de la mise en formation d'un essaim, puis celui de l'évitement de collisions entre robots et celui du contournement d'obstacle d'un groupe de robots sous-marins. Nous décrivons les tests de ces algorithmes sur un essaim de robots sous-marins réels en environnement Opensource et programmés en mode autonome. Ce travail permet également de convertir un ROV téléopéré en un hybride ROV-AUV autonome. Enfin, nous présentons des résultats de simulation et des essais réels en bassin validant les concepts proposés.

mots clés : Contrôle distribué, Réseau neuronal RBF, ROS/Gazebo, Hybride AUV-ROV, Robot sous-marin autonome

Coordination of autonomous underwater systems based on an integrated methodology in an open-source environment

This thesis studies the coordination of autonomous underwater robots in the context of coastal seabed exploration or facility inspections. In search of an integrated methodology, we have created a framework that allows the design and simulation of low-cost underwater robot controls with different model assumptions of increasing complexity. Based on this framework, we have studied algorithms to solve the problem of swarm formation, collision avoidance between robots and obstacle avoidance of a group of underwater robots. We have described the tests of these algorithms on a swarm of real underwater robots in Opensource environment and programmed in autonomous mode. This work also aims to convert a remotely operated ROV into an autonomous ROV-AUV hybrid. Finally, we have presented simulation results and real basin tests validating the proposed concepts.

keywords: Distributed control, Neural Network RBF, ROS/Gazebo, Hybrid AUV/ROV