UNIVERSITÉ DE LILLE

ECOLE DOCTORALE MADIS

UMR 9189 - Laboratoire CRIStAL

Thèse présentée pour obtenir le grade de Docteur

Discipline : Automatique informatique

Présentée par : Zaynab EL MAWAS

Localisation coopérative tolérante aux fautes : apport de l'apprentissage pour le diagnostic

Soutenue le 18/12/2023 devant le jury :

Présidente du jury :

Nadine PIAT	Professeure, École Nationale Supérieure de Mécanique et des Microtechniques

Rapporteurs :

Roland CHAPUIS	Professeur, Université Clermont-Auvergne
Vincent FREMONT	Professeur, École Centrale de Nantes

Examinateurs :

Philippe BONNIFAIT	Professeur, Université de Technologie de Compiègne
Nicole EL ZOGHBY	Docteure Ingénieure, Groupe Renault
Fawzi NASHASHIBI	Directeur de recherche, Inria Paris

Co-encadrante de thèse :

Cindy CAPPELLE	Maîtresse de Conférences, Université de Lille

Directeur de thèse :

Maan EL BADAOUI EL NAJJAR	Professeur, Université de Lille

Résumé

La mobilité autonome et connectée est devenue un enjeu socio-économique majeur. Cependant, la sûreté et la sécurité des véhicules autonomes sont des freins au déploiement de ce type de véhicules. Dans ce contexte, les travaux de cette thèse ont pour objectifs d'une part de contribuer au développement de méthodes de diagnostic des défauts des capteurs et d'autre part de mettre en œuvre une coopération entre les véhicules pour améliorer les performances de la localisation des véhicules autonomes.

Concernant l'aspect diagnostic, nous proposons de coupler des techniques basées à la fois sur des modèles et des données afin de produire une solution de localisation coopérative tolérante aux défauts capteurs. Étant donnée la nature stochastique des mesures, nous avons choisi le formalisme informationnel, qui fournit des mesures de dissimilarité entre des distributions de probabilité appelées divergences. Dans le cadre de cette thèse, nous utilisons ainsi la divergence de Jensen-Shannon pour synthétiser des indicateurs de défauts, les résidus. Le seuillage de ces résidus permet alors de détecter et d'isoler les défauts des capteurs. Par ailleurs, l'apport de l'apprentissage a été étudié pour la prise de décision du diagnostic. Deux modèles, l'un pour la détection et l'autre pour l'isolation, ont été entraînés, avec différents outils de l'apprentissage machine (perceptron multi-couches, arbre de décision et régression logistique).

La coopération entre les véhicules a mené à la mise en place d'une architecture décentralisée pour la fusion de données multi-capteurs et le diagnostic. Cet aspect coopératif inter-véhicules permet une redondance informationnelle contribuant à l'amélioration des performances de l'estimation de la pose et du diagnostic. Les données issues de cette architecture ont permis de mettre en place un paradigme fédéré pour l'apprentissage.

Les méthodes proposées ont été développées, testées et évaluées sur un ensemble de scénarios avec des défauts capteurs réels et injectés. Ces scénarios ont été créés en utilisant une base de données réelles acquises à l'aide d'une plateforme robotique conçue durant la thèse. Cet équipement de la plateforme PRETIL est constitué de trois robots communicants et instrumentés.

Mots clés : Tolérance aux fautes, Localisation coopérative, Apprentissage machine, Véhicules autonomes et connectés, Fusion multi-capteur, Robotique.

Abstract

Autonomous and connected mobility has become a major socio-economic challenge. However, the safety and security of autonomous vehicles are barriers to the deployment of this type of vehicle. In this context, the aim of this thesis is to contribute to the development of sensor fault diagnosis methods, and to implement inter-vehicle cooperation to improve the localization performance of these autonomous vehicles.

Concerning the diagnostic aspect, we propose to couple both model- and data-based techniques to produce a sensor fault-tolerant cooperative localization solution. Given the stochastic nature of the measurements, we have chosen the informational formalism, which provides measures of dissimilarity between probability distributions called divergences. In this thesis, we use the Jensen-Shannon divergence to synthesize fault indicators, the residuals. Thresholding these residuals enables us to detect and isolate sensor faults. We have also studied the contribution of learning to diagnostic decision-making. Two models, one for detection and the other for isolation, were trained, using different machine learning tools (multi-layer perceptron, decision tree and logistic regression).

Inter-vehicle cooperation has led to the implementation of a decentralized architecture for multi-sensor data fusion and diagnosis. This inter-vehicle cooperative aspect enables informational redundancy, contributing to improved performance in pose estimation and diagnosis. The data generated by this architecture has been used to implement a federated learning paradigm.

The proposed methods were developed, tested and evaluated on a set of scenarios with real and injected sensor faults. These scenarios were created using a database of real data acquired with a robotic platform designed during the thesis. The PRETIL platform comprises three communicating and instrumented robots.

Keywords : Fault tolerance, Cooperative localization, Machine learning, Autonomous and connected vehicles, Multi-sensor fusion, Robotics.

Remerciements

Les travaux présentés dans ce manuscrit ont été effectués au Centre de Recherche en Informatique, Signal et Automatique de Lille "CRIStAL", dans le cadre du lot de travail (WP) 3 du projet ANR LOCSP No 2019-CE22-0011 en collaboration avec l'université Gustave Eiffel et M3Systems.

Je tiens à exprimer ici ma respectueuse gratitude au Professeur Roland CHAPUIS de l'Université Clermont Auvergne et de l'Institut Pascal, et au Professeur Vincent FREMONT de l'Ecole Centrale de Nantes et du laboratoire LS2N qui ont accepté de juger et d'examiner mon travail.

Je remercie également les membres de jury : Professeur Philippe BONNIFAIT, de l'Université de Technologie de Compiègne et du laboratoire Heudiasyc, Docteure Nicole EL ZOGHBY du groupe Renault, Docteur Fawzi NASHASHIBI directeur de recherche à l'INRIA (Paris-Rocquencourt), et Professeure Nadine PIAT de l'École nationale supérieure de mécanique et des microtechniques et de l'institut FEMTO-ST d'avoir accepté d'examiner ma thèse.

Mes sincères remerciements vont à mon directeur de thèse Professeur Maan EL BADAOUI EL NAJJAR, et à ma co-encadrante de thèse Madame Cindy CAPPELLE pour m'avoir soutenu et encadré tout au long de cette thèse, en me faisant confiance et en croyant en mes capacités. Votre soutien continu, votre intérêt au caractère humain comme professionnel, à m'apprendre à être plus pédagogique lorsque j'étais plutôt attirée et plongée dans les détails techniques et l'expérimentation des outils m'ont aidé à développer l'esprit de chercheuse dans ce domaine, et d'avoir une bonne vision pour mon futur.

Un grand merci à tous les doctorants, stagiaires et étudiants de projets de fin d'étude grâce à qui j'ai appris beaucoup sur la gestion d'équipe et la vie professionnelle. Chacune de ces expériences, qu'elle ait été positive ou moins favorable, a contribué à forger la personne que je suis fière d'être aujourd'hui.

Mes derniers remerciements les plus sincères vont à ma famille. À mon cher père depuis que je prends ma motivation, ma curiosité technique et l'intérêt de transformer et d'appliquer les connaissances abstraites en matières concrètes. À ma chère mère, qui m'a appris le dévouement et la perfection dans le travail, et qui a investi beaucoup d'efforts pour que mes sœurs et moi puissions affronter le monde et y avoir un impact positif. À ma grande sœur, son mari et sa fille, qui ont été pour moi une véritable famille loin de chez moi. À ma seconde sœur, qui m'a soutenue et redirigée pour mon parcours universitaire. À ma troisième sœur, ma meilleure amie du monde, pour le soutien inestimable et les conseils précieux apportés tout au long de ma vie. C'est grâce à vous que je me trouve ici, et je suis infiniment reconnaissante pour tout ce que vous avez fait pour moi.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

CRIStAL	Centre de Recherche en Informatique, Signal et Automatique de Lille
ToSyMA	Tolérance aux fautes des Systèmes Mobiles Autonomes
FDE	Fault Detection and Exclusion
FDI	Fault Detection and Isolation
GPS	Global Positionning System
GNSS	Global Navigation Satellite System
LiDAR	Light Detection and Ranging
IMU	Inertial Measurement Unit
CPS	Cooperative Positionning System
NHTSA	National Highway Traffic Safety Administration
KF	Kalman Filter
EKF	Extended Kalman Filter
IF	Informational Filter
EIF	Extended Informational Filter
CIF	Covariance Intersection Filter
B-CIF	Batch Covariance Intersection Filter
S-CIF	Split Covariance Intersection Filter
B-ICIF	Batch Inverse Covariance Intersection Filter
KL	Kullback-Leibler
JS	Jensen-Shannon
UKF	Unscented Kalman Filter
PDF	Probability Density Function
ML	Machine Learning
IA	Intelligence Artificielle
ROC	Receiver Operating Characteristic
SLAM	Simultaneous Localization And Mapping
DOS	Dedicated Observer Scheme
GOS	Generalized Observer Scheme
UIO	Unknown Input Observer
NN	Neural Network
MLP	MultiLayer Perceptron

Chapitre 1

Introduction

De nos jours, nous sommes témoins d'une compétition acharnée vers l'ère de la numérisation et de l'intelligence artificielle. Cette course à l'innovation et à la transformation numérique s'étend bien au-delà des secteurs de l'urbanisme et de l'industrie. L'intérêt pour les véhicules autonomes et semi-autonomes s'est intensifié, marquant une révolution dans la manière dont nous concevons et utilisons la mobilité. Dans un monde de plus en plus connecté et basé sur les données, l'automatisation et l'IA s'imposent comme des piliers essentiels pour répondre aux défis actuels. Les véhicules autonomes, en particulier, incarnent cette révolution en redéfinissant la façon dont nous envisageons le transport.

Que ce soit dans le secteur des transports en commun ou pour les conducteurs individuels, l'automatisation, grâce à des algorithmes exerçant un contrôle complet, offre la possibilité d'améliorer l'efficacité, d'accroître la sécurité grâce à une régulation précise du comportement, et d'encourager la durabilité par une utilisation responsable qui prolonge la durée de vie des composants. Les véhicules autonomes sont spécialement conçus pour prendre des décisions en temps réel en se basant sur des données actuelles, s'adapter à leur environnement et réduire les erreurs humaines.

Dans l'urbanisme, dont l'avenir est étroitement lié au concept des villes intelligentes, l'introduction de ces véhicules offre des perspectives passionnantes pour la gestion du trafic, la réduction de la congestion et la création de villes plus durables. Avec la croissance de la connectivité au sein de ces zones urbaines, les véhicules autonomes ont la possibilité de partager leurs informations entre eux, ce qui leur permet d'améliorer leur compréhension de l'environnement et de rectifier avec précision leurs estimations de position. De plus, ces véhicules génèrent une quantité considérable de données relatives aux modèles de circulation, aux conditions routières, et bien plus encore. Ces données peuvent servir de base pour élaborer des comportements qui bénéficieront aux nouveaux véhicules entrant dans le réseau. Dans l'industrie, l'automatisation et l'IA révolutionnent la production, la logistique et la maintenance. Les usines intelligentes tirent parti de la robotique avancée pour augmenter l'efficacité et la précision des processus de fabrication.

À l'heure actuelle, ces machines se trouvent à la phase d'expérimentation et de conceptualisation. Pour passer à la phase d'implémentation, il est impératif

de démontrer leur fiabilité opérationnelle. La contribution de cette thèse réside dans la promotion de ces véhicules en communication en proposant une solution tolérante aux défauts capteurs, garantissant ainsi la précision de la localisation du véhicule, une donnée cruciale pour la planification du trajet et le contrôle du véhicule.

1.1 Contexte et problématique

La proximité des véhicules autonomes avec l'infrastructure et les usagers donne lieu à des besoins légitimes surtout en termes de sûreté de fonctionnement. En effet, la probabilité de dysfonctionnement croît en même temps que la complexité, la multiplicité des capteurs embarqués et la diversité des environnements d'évolution des véhicules. Dans le but d'atteindre une localisation sûre et précise, un effort doit être porté sur une solution multi-capteurs[1], multi-véhicules[2], de localisation coopérative[3], associée à une couche de diagnostic[4] qui permet de détecter, d'isoler et d'exclure les capteurs défaillants dans l'objectif d'assurer un fonctionnement tolérant aux défauts.

Cette thèse s'inscrit dans le lot de travail 3 du projet ANR LOCSP (LOCalisation Sûre et Précise), dont l'objectif est de développer une méthode de localisation coopérative pour un ensemble de véhicules capable de détecter et de gérer les défauts des capteurs afin d'améliorer la précision de l'estimation de la position des véhicules pour une tâche de navigation autonome sûre et sécurisée.

Nous nous focalisons sur :

1. multi-capteurs : L'algorithme développé utilise la fusion de données multi-capteurs, car nous ne pouvons pas toujours nous fier uniquement à une seule source d'information pour la localisation, puisque parfois, une source ne suffit pas pour obtenir une compréhension adéquate de l'environnement et présente des risques d'accident, en particulier pour les modifications soudaines de l'infrastructure qui affectent la connaissance de la solution de navigation cartographique. En effet, en septembre 2022, un homme est décédé à la suite d'un accident provoqué par des indications trompeuses de Google Maps. Alors qu'il se rendait chez lui, il a traversé un pont en ruines vers lequel l'application l'avait dirigé.
2. multi-véhicules : L'algorithme élaboré exploite la présence de plusieurs véhicules interconnectés, ce qui lui permet de profiter de l'observation mutuelle des autres véhicules évoluant dans le même environnement.
3. localisation coopérative : Chaque véhicule de l'ensemble a sa propre destination, et partage les informations relatives à celle-ci. Cette coopération est intégrée au sein du système sensoriel, qui constitue l'un des composants essentiels du fonctionnement des véhicules autonomes (voir Fig.1.1).
4. diagnostic : Ces nouvelles solutions cherchent à garantir d'une part la **précision** nécessaire à l'application, et d'autre part, répondre au besoin

croissant au niveau de la **sécurité de fonctionnement** du système, plus spécifiquement à la **disponibilité** de l'estimation de la géo-position des véhicules, de son **intégrité**.

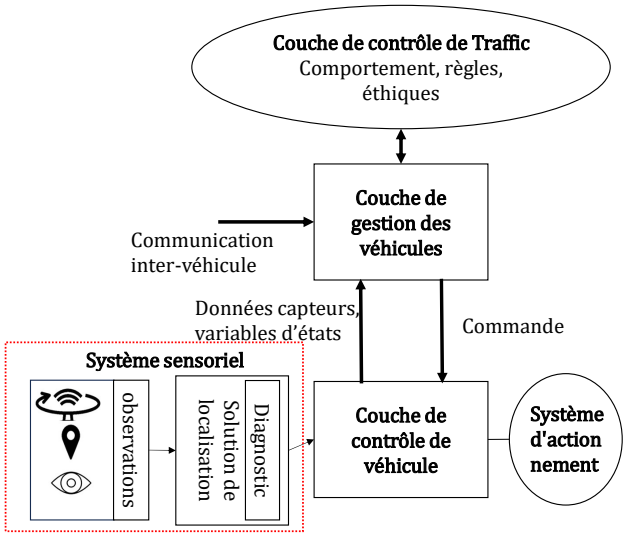


Figure 1.1 – Composants de fonctionnement des véhicules autonomes, inspiré de TSUGAWA et al., 2000

1.2 Contributions

1.2.1 Liste des contributions

L'objectif initial de cette thèse est d'établir une solution de localisation, d'un système multi-véhicule communicants, menée en coopération, qui sera tolérante aux défauts capteurs. Cette solution est basée sur le filtrage stochastique afin d'appliquer une fusion multi-capteurs. Ce filtrage produit une estimation de la pose basée sur des mesures susceptibles d'avoir des défauts. Afin de valider cette estimation, une couche de diagnostic basée sur des résidus informationnels est développée. Cette méthode intègre deux types de diagnostic : à base de modèle et guidé par les données. Son but est de détecter et isoler les défauts des capteurs dans le système. Dans ce contexte, les contributions visées par cette thèse sont les suivantes :

1. Localisation multi-véhicules et diagnostic coopératifs : L'estimation de la pose des véhicules est réalisée par la fusion des données de deux types de capteurs : ceux observant directement l'état du véhicule et ceux mesurant la distance à d'autres véhicules. L'intégration de cette distance avec la pose des autres véhicules, considérés comme repères mobiles, se

fait en mode coopératif. Ce mode sert à enrichir la solution de localisation d'une part, et d'autre part à mettre en œuvre un diagnostic mutuel en ayant un retour sur les informations partagées entre les véhicules.

2. Architecture adaptative et décentralisée : Chaque véhicule estime sa pose et applique le diagnostic sur ses données en local, suivant donc une architecture décentralisée. Cette architecture permet au système multi-véhicule de s'adapter au nombre de véhicules et de capteurs utilisés.
3. Réduction de l'impact de la propagation des erreurs : Chaque véhicule vérifie, à l'aide de sa couche de diagnostic, la précision de sa localisation en s'assurant qu'elle n'inclut pas les données de capteurs défectueux. Grâce à l'exclusion locale de ces capteurs défectueux, ainsi qu'à la validation de la solution obtenue à partir des données partagées entre les véhicules, les erreurs ne sont pas intégrées. En conséquence, ces défauts restent confinés localement et ne se propagent pas vers d'autres systèmes.
4. Intégration de méthodes d'apprentissage machine : Dans cette contribution méthodologique, nous avons intégré et comparé différentes méthodes d'apprentissage pour la prise de décision de détection et d'isolation des défauts capteurs dans le cadre du formalisme informationnel de diagnostic développé au sein de l'équipe ToSyMA.
5. Amélioration de la sécurité de partage en adoptant des modèles fédérés : En adoptant l'architecture de fusion de données multi-capteurs multi-véhicules décentralisée, ainsi que le formalisme d'apprentissage fédéré, le flux de données est diminué et la confidentialité des informations entre les véhicules est préservée, ainsi qu'à l'égard du serveur central.
6. Tester et valider sur des données réelles : Les données sur lesquelles le test et la validation sont réalisés, sont réelles. Elles proviennent d'une plateforme d'acquisition de données capteurs de trois robots communicants et instrumentés. À partir de ces données capteurs, nous générons une base de données de défauts capteurs utilisée pour l'entraînement et le test des modèles d'apprentissage.

1.2.2 Liste des publications

Conférences internationales

— Zaynab El Mawas, Cindy Cappelle, Maan El Badaoui El Najjar. Decision Tree Based Diagnosis for Hybrid Model-Based/Data-Driven Fault Detection and Exclusion of a Decentralized Multi-Vehicle Cooperative Localization System. IFAC World Congress, Jul 2023, Yokohama, Japan.

— Juliette Marais, Syed Ali Kazim, Zaynab El Mawas, Maan El Badaoui El Najjar, Jeremy Skelton, Contributions to the development of safe and

accurate localisation solutions : The LOCSP project. Transport Research Arena (TRA), Nov 2022, Lisbonne, Portugal. 8p.

— Zaynab El Mawas, Cindy Cappelle, Maan El Badaoui El Najjar, Fault tolerant cooperative localization using diagnosis based on Jensen Shannon divergence. 25th International Conference on Information Fusion (FUSION), Jul 2022, Linköping, Sweden. p.1-8.

Revue internationale

— Zaynab El Mawas, Cindy Cappelle, Mohammed Daher, Maan El Badaoui El Najjar. Comparative analysis of centralized and federated learning techniques for sensor diagnosis applied to cooperative localization for multi-robot systems. MDPI Sensors 2023, 23, 7351. https ://doi.org/10.3390/s23177351
— (en cours de soumission) Zaynab El Mawas, Cindy Cappelle, Maan El Badaoui El Najjar, Sensor fault detection and exclusion based on information theory for cooperative localization, in IEEE Robotics and Automation Letters.

Section de livre

— Zaynab El Mawas, Cindy Cappelle, and Maan El Badaoui El Najjar, "Hybrid Model/Data-Driven Fault Detection and Exclusion for a Decentralized Cooperative Multi-robot System," in Recent Developments in Model-Based and Data-Driven Methods for Advanced Control and Diagnosis, D. Theilliol, J. Korbicz, and J. Kacprzyk, Eds., in Studies in Systems, Decision and Control, vol. 467. Cham : Springer Nature Switzerland, 2023, pp. 261–271. doi : 10.1007/978-3-031-27540-1_23.

Séminaires

— Journée des Jeunes Chercheurs en Robotique (JJCR21) en Octobre 2021, organisée par le GDR-CNRS à Sorbonne Université - Campus Pierre et Marie Curie.
— Journée Régionale des Doctorants en Automatique (JRDA21) Novembre 2021 du GRAISyHM, organisée par le laboratoire Heudiasyc de l'Université de Technologie de Compiègne.
— L'école d'été de la Société de robotique et d'automatisation de l'IEEE (RAS) sur les systèmes multi-robots (MRS) en août 2022, organisée par l'université technique tchèque à Prague.
— Journée Outils Logiciels et Matériels pour la Recherche sur les Véhicules Terrestres Autonomes en Octobre 2023, organisée par le GDR-CNRS à l'ENS Paris Saclay.
— Journées Nationales de la Recherche en Robotique (JNRR) Octobre 2023, organisée par le GDR-Robotique à Moliets.

1.3 Organisation du manuscrit

Ce manuscrit est divisé en 5 chapitres :

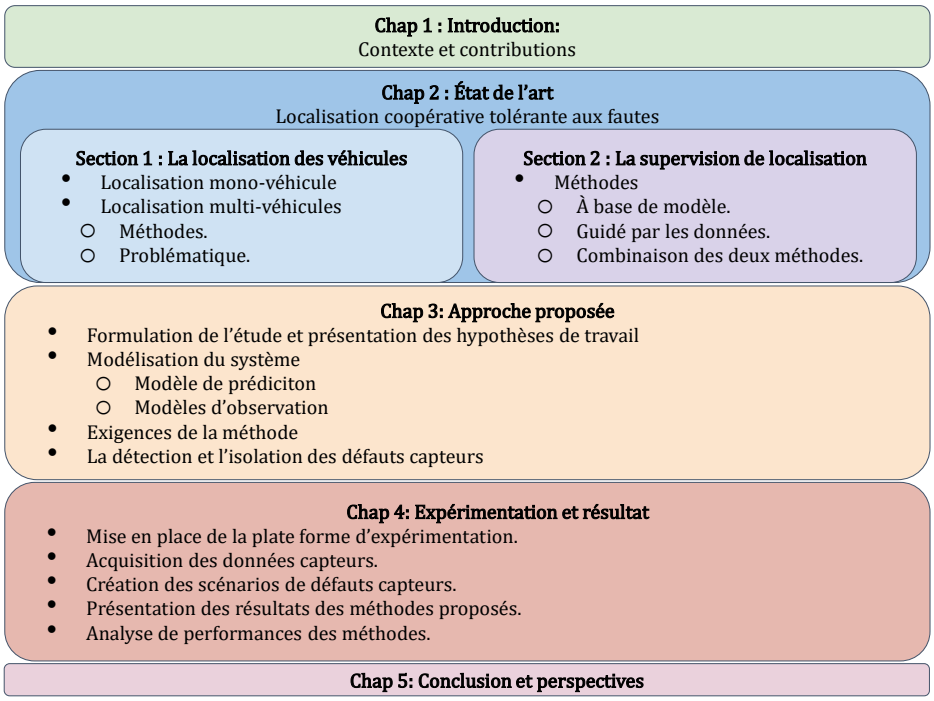


Figure 1.2 – Organisation des chapitres

1. Le présent chapitre, constitue une introduction, et présente le contexte, les contributions et la liste de publications.
2. Le chapitre 2 présente un état de l'art des domaines suivants :
   — La localisation des véhicules autonomes : Dans cette section, une étude du développement au cours des années des capteurs et des usages est présentée, aboutissant à la mise en place des premiers véhicules autonomes.
   — Les systèmes multi-véhicules : Tout d'abord, une description des systèmes multi-véhicules distinguant leurs modes de fonctionnement est présentée. Parmi ces modes, notre intérêt est accordé à la localisation coopérative, pour laquelle nous décrivons les architectures de fusion de données déployées, la manière avec laquelle la communication et la perception dans un tel système sont établies.
   — La supervision de la localisation : Dans cette section, nous présentons une vision globale de l'évolution du domaine de la supervision, avec son vocabulaire et sa terminologie. Ensuite, nous étudions les différentes méthodes de diagnostic, que nous développons ensuite.

Les méthodes à base de modèles et de la théorie de l'information constituent le socle des travaux de notre équipe. Afin de la décrire, nous commençons par l'historique du développement de cette approche, ensuite nous décrivons chacune de ses étapes. En premier lieu sont décrits les outils fondamentaux pour l'estimation d'état. Ensuite l'outil de base de la phase de diagnostic est présenté, suivi par la manière avec laquelle le diagnostic est évalué et son usage dans le cadre de la détection de défauts.

— L'usage de l'intelligence artificielle pour la tolérance aux fautes : Dans un premier temps, nous commencerons par présenter les différents types d'outils de l'intelligence artificielle ainsi que leurs domaines d'application. Ensuite, nous aborderons l'utilisation de ces outils dans le contexte de la localisation et du diagnostic, en examinant les diverses approches pour résoudre les problèmes liés à la précision de l'estimation de l'état et à la détection de défauts. Enfin, nous explorerons les méthodes existantes pour combiner les approches de diagnostic basées sur des modèles et celles guidées par les données, en mettant en évidence les avantages et les limites de chaque approche individuelle, ainsi que les bénéfices potentiels de leur combinaison.

3. Le chapitre 3 présente les méthodes de localisation tolérante aux fautes proposées. D'abord, les hypothèses de travail et la formulation du problème sont présentées, puis les protocoles de communication mis en place et l'architecture admise. Ensuite la modélisation du système, l'évolution et les observations et la façon d'estimer la position sont abordées. Une fois le système modélisé et l'évolution claire, la partie de diagnostic est présentée. Initialement, la conception et le choix des résidus est justifié. En passant à la partie applicative et matérielle, les exigences des capteurs afin de concevoir la base de données sur laquelle l'étude sera validée sont détaillés. Enfin, sont présentées chaque méthode proposée, à savoir la méthode de seuillage des résidus classique c'est-à-dire basée sur le modèle uniquement, puis la méthode combinant des deux méthodes à base de modèle et guidée par les données.
4. Le chapitre 4 traite l'expérimentation, les matériels et les plateformes utilisés et le traitement de la base de données. Puis les résultats de l'application de la méthode proposée sur la base de données sauvegardée.
5. Le chapitre 5 constitue une conclusion et avec une présentation des perspectives du travail.

Chapitre 2

État de l'art

2.1 Introduction

Depuis l'aube de l'humanité, l'homme s'est élevé au-dessus de la Terre et a contemplé les merveilles célestes qui scintillent dans l'infini, nourrissant sa curiosité innée et sa soif de comprendre l'univers qui l'entoure.

Les progrès technologiques réalisés par l'humanité sont souvent nés de cette volonté de comprendre et d'explorer l'univers. Que ce soit en observant les mouvements des étoiles dans le ciel nocturne, en scrutant les particules infiniment petites ou en étudiant les mécanismes complexes de la nature, l'homme a cherché à décoder les mystères de l'univers et à apporter son influence sur lui.

L'imagination humaine est un puissant moteur de progrès. Elle a inspiré des esprits brillants à inventer des machines volantes, des télécommunications sans fil, des satellites en orbite et bien d'autres inventions incroyables. De la théorie de la relativité d'Einstein à la conquête de la Lune, les frontières de la connaissance ont été repoussées grâce à cette capacité unique de l'homme à rêver, à concevoir et à innover.

Cette capacité de l'imagination à inspirer les avancées scientifiques s'est manifestée de manière frappante dans la pièce de théâtre tchèque *(Rossum's Universal Robots)*, où le mot "robot" a été utilisé pour la première fois. Ce terme a rapidement gagné en popularité et est devenu couramment utilisé pour décrire des machines autonomes et programmables capables d'accomplir des tâches spécifiques. Cette notion révolutionnaire a catalysé des progrès majeurs dans le domaine de l'automatisation industrielle, qui était en plein essor à cette époque, qui, jusqu'à aujourd'hui continue à évoluer grâce à l'augmentation de l'intelligence de ces machines. Lorsqu'on parle d'*intelligence*, on désigne la capacité de reproduire des comportements et des capacités intellectuelles semblables à ceux des êtres humains, telles que la résolution de problèmes, l'apprentissage, la reconnaissance de formes et le langage naturel. Le premier de ce genre était le robot *Shakey*, développé au *Stanford Research Institute* dans les années 1960 et 1970. *Shakey* était équipé de capteurs, de caméras et de capacités de planification. Il utilisait des techniques d'apprentissage pour explorer son environnement, éviter les obstacles et accomplir des tâches spécifiques.

Cependant, lorsque des machines puissantes sont impliquées, il est crucial

de s'assurer que leur comportement est maîtrisable et qu'elles ne représentent aucun danger lorsqu'elles sont en interaction avec leur environnement. C'est dans cette optique que le domaine de la supervision a été créé, ayant pour objectif de surveiller et de contrôler en permanence ces machines afin de garantir leur bon fonctionnement, leur sécurité et leurs performances.

Ce chapitre se concentre sur une revue de la littérature de la localisation des véhicules autonomes, qu'ils soient uniques ou multiples, et explore ensuite le concept de supervision dans le domaine de la localisation. En outre, il examine l'utilisation de l'intelligence artificielle pour assurer la tolérance aux défauts des capteurs. Ce faisant, il met en lumière le contexte de cette thèse.

2.2 Localisation des véhicules autonomes

Le problème fondamental de la localisation réside dans la nature même de l'environnement dans lequel les agents autonomes évoluent. L'environnement est souvent incertain, avec des facteurs tels que le bruit des capteurs, les erreurs de mesure, les obstacles imprévus et les changements dynamiques.

La localisation de robots mobiles, une sous-partie du problème fondamental de localisation fait référence à la capacité d'un robot à estimer de manière continue et fiable sa position et son orientation par rapport à un référentiel de l'environnement. Ce problème fondamental réside dans l'obtention d'une précision, d'une fiabilité et d'une robustesse élevées, quelles que soient les conditions environnementales et les situations de conduite. La localisation des robots mobile est souvent associée à :

Définition 1 *"La détermination de la position du robot par rapport à un cadre de référence global" selon* Leonard *et al., 1992.*

Définition 2 *"L'estimation de la position ou suivi de la position" selon* Thrun, Burgard *et al., 2005.*

Définition 3 *"La capacité de détecter et d'estimer avec précision l'emplacement d'une plateforme" selon* Durrant-Whyte, Rye *et al., 1996.*

L'importance de la localisation dans le domaine des véhicules autonomes ne peut pas être sous-estimée. Une localisation précise est cruciale pour garantir la sécurité des passagers, du véhicule lui-même et des autres usagers de la route, ainsi que pour assurer une navigation précise et efficace du véhicule.

La localisation, remontant à ses origines, trouve ses racines dans l'astro navigation ou la navigation astronomique, qui repose sur l'observation des étoiles et d'autres corps célestes pour déterminer la position. L'étude des astres a toujours suscité l'intérêt et la motivation de l'humanité, ce qui se reflète dans l'évolution des sciences. Les différentes étapes de cette évolution sont perceptibles, allant des lois de Kepler au XVIIe siècle qui ont jeté les bases de notre compréhension des orbites des objets célestes, à la formulation de la loi de la gravitation universelle par Newton en 1687. Plus tard, au début du XXe siècle,

la théorie de la relativité d'Einstein, développée en 1905, a ouvert de nouvelles perspectives et a conduit à la conceptualisation des premiers satellites.

Ces concepts ont été concrétisés par les premières avancées dans le domaine spatial, notamment avec le lancement de Spoutnik 1 et 2 par l'Union soviétique en 1957, et celui de l'Explorer 1 par les États-Unis peu après, pendant la période de la guerre froide marquée par une intense course à l'espace entre ces deux nations.

À la suite de recherches approfondies, le projet TRANSIT a été lancé dans les années 1960 par le Département de la Défense des États-Unis. Ce système de navigation par satellite était précurseur du GPS (Global Positioning System), qui a vu le jour en 1970. Initialement conçu à des fins militaires, le GPS est devenu pleinement opérationnel en 1995, offrant aux utilisateurs civils une précision de localisation de l'ordre de quelques mètres. Le GPS fonctionne en utilisant la méthode de triangulation. Chaque satellite diffuse en permanence un signal radio contenant des informations précises sur sa position et l'heure à laquelle le signal a été émis. En connaissant la distance entre le récepteur et chaque satellite (calculée à partir du temps de trajet du signal), le récepteur peut alors effectuer des calculs complexes pour déterminer sa propre position géographique en termes de latitude, de longitude et d'altitude.

En parallèle, l'évolution logique suivante des systèmes de localisation est apparue dans le domaine cellulaire, où l'exigence de localisation a été lancée par le mandat Emergency-911. Avant que le GPS ne soit largement disponible sur les téléphones mobiles, les opérateurs cellulaires ont adopté et déployé diverses technologies pour localiser les mobiles dans le rayon d'une tour de transmission. Les techniques de calcul basées sur le temps d'arrivée, qui proviennent des systèmes GPS, ont été adaptées afin d'obtenir des performances de localisation similaires pour le canal commun des téléphones cellulaires.

Cependant, la solution prometteuse du GPS avait par contre ses limitations comme la dépendance aux signaux satellitaires. En effet, dans les environnements dans lesquels la réception des signaux GPS est obstruée, la précision de la localisation peut être réduite (voire la localisation peut même être impossible), en raison des erreurs de propagation, de latence et de multi-chemin nocif à l'exactitude de la solution. Ces limitations du GPS ont poussé vers le recours à d'autres solutions, qui appartiennent à l'environnement observé. Au cours de la progression dans la conquête de l'espace dans les années 1960, l'humanité a abouti à l'invention et au développement de diverses technologies visant à optimiser l'atteinte de ses objectifs. Parmi ces avancées, les systèmes LiDAR (Light Detection and Ranging) ont été développés pour la mesure de distances, les gyroscopes ont été exploités pour les changements d'orientation, et les dispositifs de capture d'images, tels que les caméras, ont été mis en œuvre pour la capturer le contexte de l'environnement.

Initialement conçu pour mesurer avec précision la distance entre la Terre et la Lune, le système LiDAR, basé sur le principe de la télémétrie laser, a rapidement dépassé son application initiale. Il s'est avéré être une technologie polyvalente et a trouvé des applications dans divers domaines, notamment la cartographie topographique et la géologie. La capacité du LiDAR à générer

des cartes 3D détaillées de l'environnement a été exploitée pour la localisation précise d'objets et de structures, offrant ainsi des opportunités prometteuses pour l'odométrie télémétrique et la cartographie tridimensionnelle.

D'autre part, l'utilisation de gyroscopes, d'unités de mesure inertielle (IMU) et d'encodeurs incrémentaux a été d'une importance capitale pour la navigation spatiale et l'orientation des engins. Ces dispositifs ont permis de déterminer avec précision la position et l'orientation relatives des véhicules spatiaux, garantissant ainsi leur trajectoire avec une haute précision, même dans des environnements dynamiques et imprévisibles.

En parallèle, la vision par ordinateur, étayée par l'usage de caméras sophistiquées, a constitué une avancée significative dans la compréhension visuelle de l'environnement. En permettant aux machines d'interpréter les informations visuelles de manière analogue à la perception humaine, la vision par ordinateur a ouvert de nouvelles perspectives en matière de localisation et de navigation. Notamment, l'odométrie visuelle basée sur la stéréo-vision a permis aux machines de naviguer dans des environnements inconnus en se basant sur les caractéristiques visuelles capturées, leur permettant de suivre leur évolution relative et d'effectuer des mises en correspondance en temps réel avec des modèles 3D préexistants stockés en mémoire.

En associant ces technologies de manière synergique, l'homme a pu créer des systèmes de navigation et de cartographie avancés, exploitant les avantages de chaque composant pour atteindre un niveau de performance encore plus élevé. Ces développements novateurs ont ouvert la voie à des explorations spatiales plus poussées, ainsi qu'à une meilleure compréhension et exploitation de l'environnement terrestre. Dans un contexte de recherche scientifique et de développement technologique, l'intégration et la combinaison judicieuse de ces avancées ont été cruciales pour optimiser les capacités de localisation et de navigation, ouvrant ainsi de nouvelles perspectives pour les futurs défis spatiaux et terrestres.

Le recherche dans le domaine de localisation de véhicules a beaucoup progressé. C'était initialement en 1939 par des véhicules opérés à distance par des signaux radio, puis par l'usage des dispositifs électroniques intégrés dans la chaussée, arrivant à la fin au projet ALV (Autonomous Land driven Vehicle) financé par la DARPA, à la première démonstration de suivi de route utilisant le lidar, la vision par ordinateur et la commande robotique autonome pour diriger un véhicule robotisé. En 2006, l'équipe du Laboratoire d'intelligence artificielle de Stanford a développé *Stanley*. Ce véhicule comporte cinq télémètres laser SICK, une caméra couleur pour la perception de la route à longue distance, des capteurs RADAR couvrant la zone frontale jusqu'à 200 mètres, pour informer *Stanley* du terrain à venir. De plus, deux antennes du système de positionnement GPS sont montées avec une unité de mesure inertielle (IMU) Buehler et al., 2007. Initialement, les recherches ont débuté par la localisation des robots en intérieur : Durrant-Whyte, 1988 a commencé par étudier l'incertitude des robots dans des milieux internes, puis vint la localisation de véhicules de vitesse élevée Julier et al., 1998 qui a déclenché une série d'études dans le domaine. Mais c'est grâce à la collaboration entre *Sebastien Thrun*, *Dieter Fox*

et *Wolfram Burgard* débutant par l'article THRUN, BURGARD et al., 1998 que ce progrès dans le domaine de la localisation de véhicules et l'usage de ces algorithmes ont été permis. Cette collaboration a été concrétisée par le livre *Probabilistic Robotics* de THRUN, BURGARD et al., 2005, qui constitue un ouvrage fondamental dans le domaine de la robotique intelligente et des véhicules autonomes.

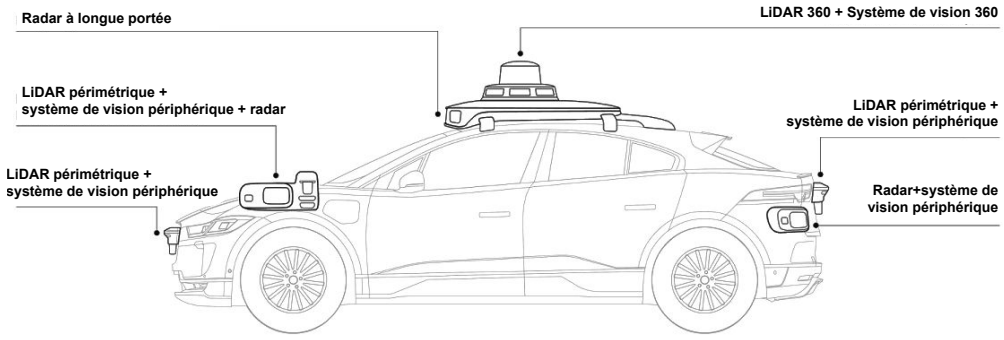


Figure 2.1 – Capteurs du véhicule Waymo de Google ©

De suite, la fusion entre localisation et cartographie a eu lieu. En effet, le fameux "Simultaneous Localization and Mapping - SLAM" (localisation et cartographie simultanées) a été initialement conceptualisé par DURRANT-WHYTE et BAILEY, 2006. Cet algorithme se base sur l'estimation simultanée de la position d'un robot (localisation) et sur la création d'une carte de son environnement (cartographie). Il s'agit d'une capacité cruciale pour les robots autonomes, car elle leur permet de naviguer et d'opérer dans des environnements inconnus ou partiellement cartographiés.

Thrun était à l'origine de la compagnie *Waymo* qui aujourd'hui rend des services de transport dans de nombreux États d'Amérique, et il est considéré comme le parrain de l'industrie de la voiture autonome.

La recherche dans le domaine de localisation a depuis lors continué à s'élargir, notamment les travaux de :

UPCROFT et al., 2006 présentent le développement et la démonstration d'une estimation d'état décentralisée non gaussienne à l'aide d'un réseau de capteurs extérieurs composé d'un véhicule aérien autonome, d'un véhicule terrestre manuel et de deux opérateurs humains. La localisation et l'apparence des repères ont été estimées en utilisant des observations de direction à partir de caméras monoculaires. Dans CAPPELLE et al., 2007, une méthode de géolocalisation a été proposée en utilisant le GPS, l'INS, une caméra monovision et une nouvelle source de géoinformation, qui est le modèle cartographique 3D. REINA et al., 2011 explore la perception des environnements extérieurs par des véhicules autonomes en utilisant un radar à ondes millimétriques. L'innovation réside dans l'utilisation du radar pour segmenter le sol et améliorer la précision de portée. BENDER et al., 2014 présente une carte détaillée pour la conduite autonome, introduisant le concept de "lanelets" pour représenter l'environnement praticable. Les "lanelets" sont des segments de route interconnectés avec des données supplémentaires. La spécification de la carte, le processus de création

et la bibliothèque d'accès sont décrits.

Bauer et al., 2016 met en avant l'utilisation d'algorithmes de fusion de données pour améliorer la précision et l'intégrité de la localisation, notamment en utilisant les cartes haute définition (HD). L'article présente un algorithme de localisation basé sur les filtres à particules qui intègre les GNSS, l'odométrie et les cartes HD pour démontrer les avantages des cartes HD à partir de mesures réelles.

Pour clore cette section, à mesure que les véhicules autonomes continuent de progresser vers une intégration plus étroite dans notre société, la recherche en localisation reste au cœur de leur fonctionnement sûr et efficace. La combinaison d'une compréhension historique, d'innovations technologiques et d'approches novatrices forme le socle sur lequel repose la poursuite du développement des véhicules autonomes.

2.3 Systèmes multi-véhicules

Dans le contexte actuel de l'intelligence artificielle et de l'automatisation, les systèmes multi-véhicules émergent comme des acteurs clés pour résoudre des problèmes variés, allant de la logistique à la surveillance environnementale. La localisation de ces systèmes, qui consiste à estimer avec précision leur position et leur orientation dans un environnement donné, constitue un pilier fondamental pour leur fonctionnement fluide et fiable.

Le problème de la localisation se pose naturellement pour les équipes de robots. À première vue, chaque robot peut se localiser individuellement et le problème de la localisation multi-robots peut donc être résolu par la localisation d'un seul robot. Si les robots sont capables de se détecter mutuellement, il est toutefois possible de faire mieux, car la croyance d'un robot peut servir de base à la croyance d'un autre robot si l'on connaît la position relative des deux robots. La localisation multi-robots soulève des questions intéressantes et non triviales sur la représentation des croyances et la nature de la communication entre elles. Nous définissions ce système :

Système multi-véhicules : "désigne un ensemble de véhicules interagissant de manière coordonnée pour atteindre des objectifs communs ou individuels."

Ces systèmes présentent un potentiel immense pour révolutionner divers secteurs en permettant des opérations plus efficaces, une meilleure utilisation des ressources et une réduction des risques humains. Qu'il s'agisse de flottes de véhicules de livraison coordonnés pour optimiser les itinéraires, de drones collaborant pour cartographier des zones difficiles d'accès ou de véhicules terrestres opérant dans des environnements hostiles, la coordination entre les véhicules repose sur une localisation précise et fiable.

La localisation au sein d'un système multi-véhicules diffère significativement de la localisation traditionnelle d'un véhicule isolé. Les exigences augmentent en raison des interactions complexes entre les véhicules, de la nécessité de maintenir une cohérence spatiale en temps réel et de la gestion des incertitudes inhérentes à tout environnement réel. En effet, dans un système multi-véhicules,

lors du partage des information, et leur intégration à plusieurs itérations, les erreurs de mesure peuvent s'amplifier à cause de l'accumulation d'erreurs et compromettre la fiabilité des estimations globales. De plus, la variabilité des capteurs, les obstructions environnementales et les erreurs de communication introduisent des défis considérables dans la tâche de localisation.

Quant aux apports, la coordination au sein d'un système multi-véhicules apporte des avantages considérables à la performance de l'ensemble du système. En reliant les actions individuelles des véhicules à une stratégie collective, la coordination permet d'atteindre des résultats qui dépassent les capacités de chaque véhicule pris isolément. De plus, cette coordination permet d'éviter les conflits, d'optimiser les trajectoires et de réaliser des tâches complexes de manière efficace.

Dans la suite de cette section, nous présentons les méthodes et les approches de traitement d'un système multi-véhicules, puis l'implémentation d'un système de localisation coopérative.

2.3.1 Méthodes et approches de coordination des systèmes multi-véhicules

Lors de traitement du sujet de système multi-véhicules, il faut distinguer entre deux termes spécifiant deux modes d'opération différents : coopération et collaboration.

Sous le concept de collaboration, nous trouvons le comportement *eusocial*. Ce comportement se retrouve chez de nombreuses espèces d'insectes (par exemple, les colonies de fourmis ou d'abeilles) et est le résultat d'un comportement individuel génétiquement déterminé. Dans les sociétés eusociales, les agents individuels ne sont pas très capables, mais un comportement apparemment intelligent naît de leurs interactions. Ce comportement "collaboratif" est nécessaire à la survie des individus dans les colonies.

À partir de ces informations, nous proposons la définition suivante :

Collaboration : "une tâche est menée en collaboration si un ensemble de machines travaillent ensemble de manière proactive, en partageant activement leurs connaissances, leurs compétences et leurs ressources pour atteindre un objectif commun".

D'autre part, le comportement coopératif est définie par McFarland, 1994 comme le comportement social observé chez les animaux supérieurs (vertébrés), c'est-à-dire que la coopération est le résultat d'interactions entre des agents égoïstes. Contrairement au comportement eusocial, le comportement coopératif n'est pas motivé par un comportement inné, mais par un désir intentionnel de coopérer afin de maximiser l'utilité individuelle.

À partir de ces connaissances, nous proposons la définition suivante :

Coopération : "une tâche est menée en coopératif si un ensemble de machines agissant indépendamment travaillent ensemble en apportant leur contribution respective sur l'ensemble pour atteindre des objectifs individuels".

Dans une vue historique du domaine, la première tentative de création de système multi-robots était dans les années 1940, où *William Grey Walter*

a étudié des robots équipés de capteurs de lumière et de toucher étudiant le "comportement social complexe" en réagissant aux mouvements de leurs congénères. Durant la première période de leur création et développement à la fin du XXième siècle, l'intérêt était de cadrer et définir ces systèmes, les théories à appliquer et la résolution de problématiques possible grâce à cette technique. Un intense intérêt était alors accordé à la coordination et l'interaction de multiples agents logiciels intelligents dans le cadre du concept d'intelligence artificielle distribuée (DAI) qui s'est porté sur la tâche, le mécanisme de coopération et la performance du système. Les domaines d'applications se sont rapidement élargis, et c'était à la période entre 1987 et 1995 où le domaine a prospéré, avec plus de 200 publications selon Cao et al., 1997 sous le sujet de multi-véhicules. Ces travaux constituent la base des recherches qui ont permis son utilisation étendue et avancée dans les applications actuelles.

Un premier formalisme a été présenté par ACTRESS (ACTor based Robots and Equipment Synthetic System) de Matsumoto et al., 1990 appliqué sur des robots décentralisés hétérogènes d'autonomie de haut-niveau en fonction indépendante ou en coopération par communication. Les intérêts dans ce domaine étaient variés. Certains chercheurs, tels que Le Pape, 1990, ont examiné les architectures en vue de coordonner les actions des robots dans un environnement dynamique imprévisible. D'autres, comme Brooks, 1991, ont travaillé sur la création de robots autonomes capables de fonctionner en parallèle sans supervision. Yuta et al., 1991 ont concentré leurs recherches sur l'évitement des collisions, tandis que Singh et al., 1993 se sont penchés sur la cartographie. Enfin, Yoshida et al., 1994 ont exploré l'impact du comportement du groupe sur les performances de communication.

Les différentes manifestations des systèmes multi-véhicules ont évolué pour revêtir diverses formes, dont l'une des plus notables est les systèmes multi-agents manipulant des objets passifs au sein d'un environnement partagé comme définit par Ferber, 1995 et les essaims (swarms), visant l'étude de la conception et l'expansion d'un grand nombre d'agents physiques afin de maintenir un comportement collectif grâce à aux interactions locales et avec l'environnement, telle que expliquée par Soysal et al., 2005.

Les applications dans ces domaines partaient de l'exploration collective, au mouvement coordonné, transport collectif, prise de décision collective, obtention d'un consensus et allocation de tâche, selon l'étude des systèmes multi-agents de Rossi et al., 2018 et celle des essaims de Brambilla et al., 2013. La principale différence entre les deux types réside dans le niveau d'autonomie et d'intelligence des individus et dans la nature de la coordination. Les systèmes multi-agents impliquent des entités intelligentes, souvent hétérogènes, capables de prendre des décisions autonomes. Les essaims, en revanche, reposent sur des agents simples dotés d'une intelligence individuelle limitée et fonctionnent sans communication explicite ni contrôle centralisé, leur comportement émergeant des interactions locales.

En conclusion, l'évolution des systèmes multi-robots a parcouru un chemin passionnant, de ses premiers balbutiements à des avancées révolutionnaires. Au fil du temps, ces systèmes ont évolué depuis les premières notions de co-

opération et de coordination jusqu'à des domaines d'application diversifiés. Aujourd'hui, les systèmes multi-robots sont essentiels dans des secteurs allant de la robotique autonome à la logistique, de la recherche en environnements inconnus à la surveillance de l'environnement, et bien plus encore (voir Fig.2.2). Alors que la recherche et le développement continuent de progresser, l'avenir des systèmes multi-robots promet de révolutionner encore davantage les industries, ouvrant la voie à des innovations extraordinaires et à des applications encore inexplorées.

(a) Entrepôt kiva GUIZZO, 2008 **(b)** Spectacle de Lumières WENG et al., 2022 **(c)** Fusionnable MATHEWS et al., 2017

(d) MiniCity BUCKMAN et al., 2022 **(e)** Réponse aux catastrophes ROUČEK et al., 2022 **(f)** Agriculture GONZALEZ-DE-SANTOS et al., 2017

Figure 2.2 – Exemples de domaines d'application de systèmes multi-véhicules

2.3.2 Localisation coopérative

Le terme "comportement collectif" désigne de manière générique tout comportement des agents dans un système comportant plus d'un agent. Le comportement coopératif est une sous-classe du comportement collectif caractérisé par la coopération.

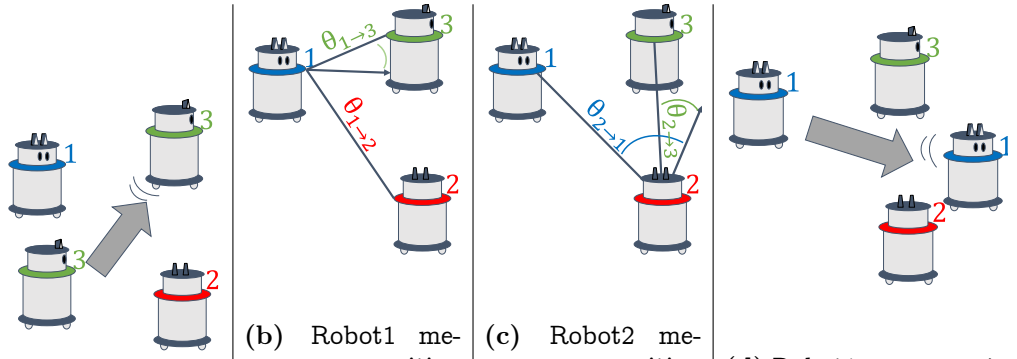
Définition 4 *coopérer : "Opérer conjointement avec quelqu'un ; concourir à une œuvre ou à une action commune ", selon* Dictionnaire de l'Académie Française *1992.*

Les définitions explicites de la coopération dans la littérature robotique à la fin du XXème siècle étaient rares, et non directement liées au domaine. La première définition explicite a été mise par CAO et al., 1997 :

Définition 5 *"Compte tenu d'une tâche spécifiée par un concepteur, un système robotique multiple présente un comportement coopératif si, en raison d'un mécanisme sous-jacent, c'est-à-dire le "mécanisme de coopération", il y a une augmentation de l'utilité totale du système".*

Dans ses premiers usages, la localisation coopérative avait comme but de cartographier et naviguer dans un espace inconnu intérieur, et bénéficier de la connaissance que le partage de données et la perception d'autres véhicules dans le même environnement peut apporter.

L'établissement de cette localisation passe par plusieurs étapes dans l'historique. Nous avons fait l'état de l'art de la localisation coopérative d'une façon historique, commençant par l'origine, les développement de la même ère, arrivant aux recherches récentes les plus répandus dans le domaine. Le "système de positionnement coopératif (CPS)" a été introduit pour la première fois par KURAZUME et al., 1994 pour résoudre le problème de l'accumulation d'erreur de l'approche à l'estime dans un système multi-robot, et la navigation dans un environnement inconnu. Leur concept de division du système en "points de repère portables" stipule que les véhicules sont divisés en deux groupes et qu'à chaque instant, une équipe est en mouvement, tandis que l'autre reste stationnaire et sert de point de repère, puis les rôles sont inversés plusieurs fois jusqu'à ce que les deux équipes atteignent leur cible. Cette méthode a été améliorée au cours des années 1994 et 2000, et était à l'inspiration de plusieurs recherches qui ont suivi dans cette période.



(a) Robot3 avance

(b) Robot1 mesure sa position par rapport aux Robots2 et 3

(c) Robot2 mesure sa position par rapport aux Robots1 et 3

(d) Robot1 avance après l'estimation de la position du Robot3

Figure 2.3 – Évolution de la méthode de points de repère portables

D'une façon similaire, GRABOWSKI et al., 2000 proposent une approche où un seul robot se déplace, tandis que le reste de l'équipe de robots de petite taille forme un triangle équilatéral de balises de localisation afin de mettre à jour leurs estimations de pose par usage du sonar. Par la suite, un algorithme de cartographie bayésienne de la grille d'occupation est utilisé pour combiner les données des capteurs de plusieurs robots.

En parallèle, FOX et al., 1999 ont développé la localisation collaborative de deux robots d'intérieur équipés d'un modèle de l'environnement en utilisant une version basée sur des échantillons de l'algorithme de localisation de Monte Carlo (MCL).

Les avancées continues dans ce domaine de recherche ont engendré de nombreuses opportunités et sous-domaines pour affiner la solution, demandant une analyse approfondie. Ces défis incluent la résolution de conflits, la gestion de l'incertitude* du système, les mécanismes de communication, la garantie de

l'intégrité*, la prévention de la consanguinité, et le maintien de la cohérence des données.

En ce qui concerne l'incertitude, SANDERSON, 1997 ont examiné l'impact de l'incertitude sur l'estimation de l'état en utilisant le filtre de Kalman. Ils ont négligé l'effet des mesures de la position relative, ce qui a conduit à la création d'un algorithme distribué simplifié. En utilisant ce même outil, S. ROUMELIOTIS et al., 2002 ont décomposé en un certain nombre de filtres communicants plus petits, un pour chaque robot, traitant les données des capteurs collectées par son robot hôte. En changeant l'estimateur par celui du maximum de vraisemblance, HOWARD, MATARK et al., 2002 ont traité la pose relative et les mesures odométriques enregistrées par les robots et une solution a été dérivée en faisant appel à l'optimisation numérique. Deux années plus tard, S. I. ROUMELIOTIS et al., 2004 ont étudié l'amélioration de la précision de la localisation par robot supplémentaire à mesure que la taille de l'équipe augmente, et le calcul de l'expression analytique pour la limite supérieure du taux d'augmentation de l'incertitude de positionnement pour une équipe de N robots en fonction de N.

Du point de vue de l'intégrité, REKLEITIS, IOANNIS M. et al., 2002 présentent le cas d'une application intérieure ou dans un environnement dépourvu de GPS, et se concentrent sur les questions liées à l'estimation de la confiance pour les groupes de plus de deux robots afin d'obtenir une estimation précise de la position.

Ces applications étaient dans des environnements intérieurs, ce qui n'est pas adapté aux milieux externes. En effet, dans ces environnements, les erreurs introduites en raison de la distance parcourue peuvent être importantes et imprévisibles. C'est une conséquence directe de la nature ondulatoire du terrain et des incertitudes introduites dans les données des capteurs. Afin d'étudier ce cas, MADHAVAN et al., 2002 ont présenté un cadre de localisation hétérogène multi-robots basé sur l'EKF sur un terrain extérieur inégal et non structuré, avec un modèle cinématique non linéaire et une capacité de positionnement absolu au moyen d'un GPS différentiel (DGPS). KARAM, N. et al., 2006 ont proposé une localisation collective du groupe basée sur l'échange d'états tout en tenant compte des interdépendances entre les distributions des estimations de positions et les différentes informations de détection et contraintes de communication. Afin de résoudre ce problème, une autre méthode présenté par LI, H. et al., 2013 traite ce problème par l'usage du filtre d'intersection de covariances CIF, avec des véhicules qui partagent la position absolue, la position relative et les mesures de l'état du mouvement.

De nos jours, les recherches ont progressivement évolué d'une façon multi-discipline, soulignant que le travail ne repose pas exclusivement sur un seul domaine comme la communication, l'architecture, la perception, ou l'estimation d'état individuellement, mais plutôt sur une synthèse de ces domaines. C'est le cas des travaux de LASSOUED et al., 2016, où la localisation coopérative devient un problème d'inversion d'ensembles distribués, en proposant des véhicules qui coopèrent et échangent des informations afin d'améliorer l'estimation absolue et relative en fusionnant les corrections de pseudo-range partagées entre eux,

communiquant dans un réseau de véhicule à véhicule (V2V) de manière instantanée (époque par époque). De plus, PIERRE et al., 2018 présentent une localisation basée sur des mesures de portée uniquement en bande ultra-large (UWB), l'algorithme de fusion est basé sur un filtre de covariance à intersection divisée S-CIF, sous une architecture décentralisée. D. WANG et al., 2023 proposent un algorithme de localisation coopérative robuste et décentralisée, avec adaptation du processus de mise à jour des mesures de l'algorithme conventionnel de localisation coopérative décentralisée pour surveiller les corrélations entre les robots et garantir l'autonomie du processus de mise à jour des mesures pour les filtres individuels.

En conclusion, la localisation coopérative a connu une évolution remarquable, passant d'une idée théorique à une réalité pratique avec des applications variées et prometteuses. Elle a trouvé sa place dans des domaines aussi divers que la robotique autonome, la navigation GPS, la gestion de flottes, et même la réalité augmentée. À mesure que la technologie continue de progresser, nous pouvons anticiper des avancées encore plus significatives dans ce domaine, ouvrant la voie à de nouvelles opportunités passionnantes pour résoudre des problèmes de localisation complexes de manière coordonnée.

Dans la suite de cette section, nous développons les approches de coopération, en terme d'architecture de fusion de données, communication et perception entre véhicules.

2.3.2.1 Architectures de localisation coopérative

Selon l'application, l'architecture de fusion change pour répondre à certaines exigences de performance, temps de calcul, tolérance aux fautes et évolutivité, pour avoir une optimisation des résultats obtenus. Selon DURRANT-WHYTE et HENDERSON, 2008, les architectures de fusion multi-capteurs peuvent être classées en fonction du choix effectué selon quatre dimensions de conception indépendantes : *centralisé-décentralisé*, interaction *locale-globale* des composants, *modulaire-monolithique*, et *hétéroclite-hiérarchique*. Dans la suite nous présentons les architectures (du point de vue de la dimension *centralisée-décentralisée)* les plus connues dans le domaine de fusion multi-capteurs.

1. **Architecture centralisée** : Les informations des capteurs sont communiquées à une unité centrale qui les traite. Selon la définition de MADHAVAN et al., 2002, dans une telle architecture, toutes les tâches de planification, d'exécution, de contrôle et de surveillance sont effectuées par une seule unité de contrôle. Il est très difficile d'avoir un système entièrement centralisé pour le contrôle de plusieurs robots, car l'augmentation de la charge de calcul est proportionnelle au nombre de robots dans l'équipe. Cette architecture est gourmande en bande passante car les données brutes des capteurs doivent être envoyées. Une puissance calculatoire importante est donc requise, mais elle profite de l'information globale du système.
2. **Architecture hiérarchisée** : L'augmentation de l'intelligence des

nœuds de capteurs locaux se traduit naturellement par une structure hiérarchique de l'architecture de fusion. Les éléments de niveau de traitement bas transmettent des informations vers le niveau haut, à travers les niveaux intermédiaires (voir Fig.2.4).

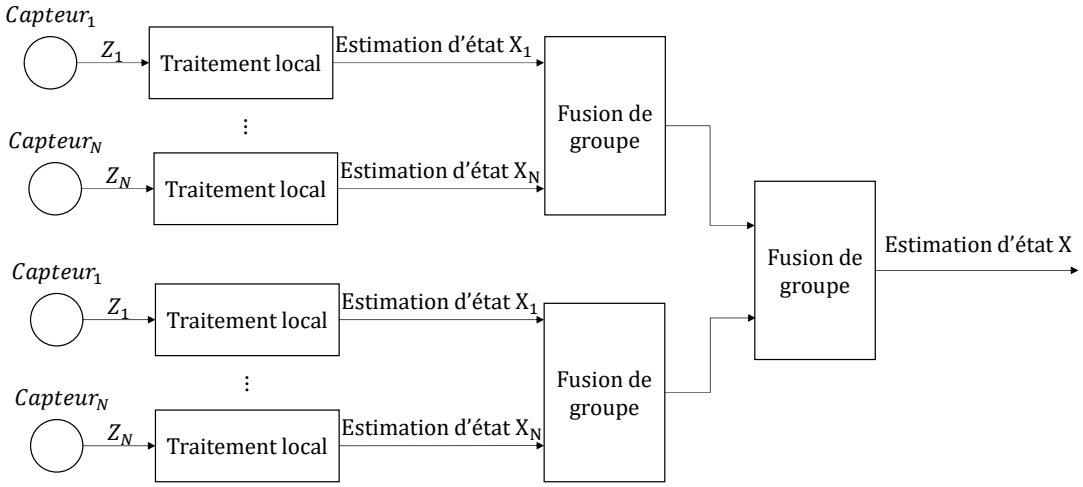


Figure 2.4 – Architecture hiérarchisée

3. **Architecture distribuée** : Dans une architecture distribuée, chaque capteur possède un processeur local qui permet d'extraire et traiter l'information utile. L'intérêt est à la fois de transmettre moins d'informations et de réduire la charge de calculs de l'unité centrale.

   En outre, l'ajout ou l'exclusion d'un élément n'impose aucun changement sur l'architecture globale du système, mais nécessite un certain type de contrôle central. La migration vers les organisations de systèmes distribués est la plus évidente dans les domaines d'application de l'Intelligence Artificielle (IA), où l'IA distribuée est devenue un domaine de recherche à part entière. Ces systèmes utilisent une plateforme partagée (Blackboard) pour communiquer les informations.

4. **Architecture décentralisée** : Un système décentralisé se compose d'un ensemble de capteurs dont chacun possède son propre processeur local. La fusion se produit au niveau de chaque nœud sur la base de ses observations locales et de l'information communiquée par les nœuds voisins. Une organisation décentralisée se distingue d'un système de traitement distribué par l'absence d'installation de centrale de traitement ou de communication. Chaque nœud de capteur dans une organisation décentralisée est entièrement autonome et peut fonctionner de manière totalement indépendante de tout autre composant du système. La communication entre les nœuds est strictement biunivoque. Cette architecture est présentée dans la Fig.2.5.

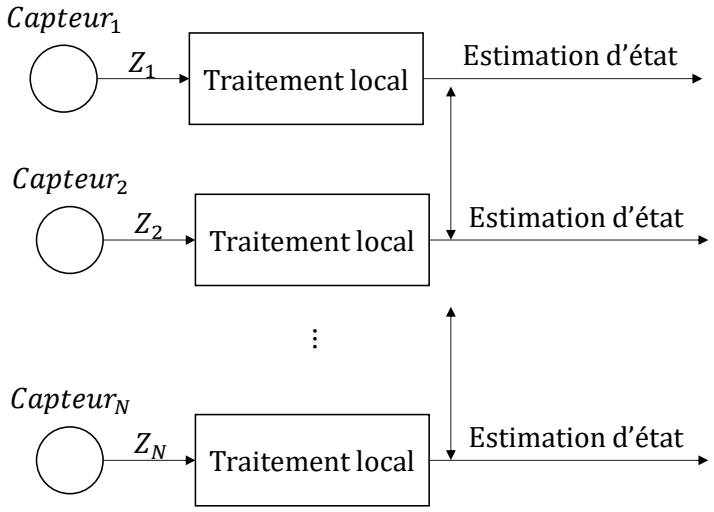


Figure 2.5 – Architecture décentralisée

2.3.2.2 Communication dans un système multi-véhicules

La réussite d'un système multi-véhicules repose en grande partie sur sa capacité à établir une communication efficace entre les véhicules eux-mêmes ainsi qu'avec l'infrastructure. Cette communication joue un rôle fondamental dans la coordination des mouvements et la prise de décisions collectives au sein du système. La structure de communication d'un groupe de véhicules détermine les différents modes d'interaction possibles entre les agents. Parmi les premières méthodes de communication, on compte l'utilisation des clignotants et même la transmission verbale via la radio, illustrant ainsi les débuts de l'interconnexion entre les véhicules sur la route. En terme de classification d'usage de communication, les réseaux de véhicules connectés définissent différents modes de communication pour automatiser la propagation des messages : Véhicule à Véhicule (V2V), Véhicule à Infrastructure (V2I), Véhicule au réseau *Network* (V2N), Véhicule à Piéton (V2P) et Véhicule à tout (V2X) (voir Fig.2.6).

Dans le contexte de coopération de véhicules, l'intérêt initial repose sur le V2V, une communication qui va continuer à augmenter en raison de la croissance exponentielle du nombre de véhicules sur les routes. Aujourd'hui, le nombre total de véhicules (voitures, camions et bus) dans le monde a déjà atteint 1,32 milliard en 2016 selon Petit, 2017, soit près du double du volume enregistré 20 ans auparavant, lorsque les véhicules en circulation totalisait 670 millions d'unités en 1996. Selon certaines prévisions, ce nombre pourrait atteindre 2,8 milliards de véhicules d'ici à 2036.

Dans le domaine de la communication dans les systèmes multi-véhicules, il est essentiel de comprendre les différentes technologies et normes qui sont déployées pour faciliter l'échange d'informations entre les véhicules autonomes.

À l'heure actuelle, il n'existe pas de normes de données sans fil universelles pour l'échange d'informations entre les véhicules autonomes. Cependant, le

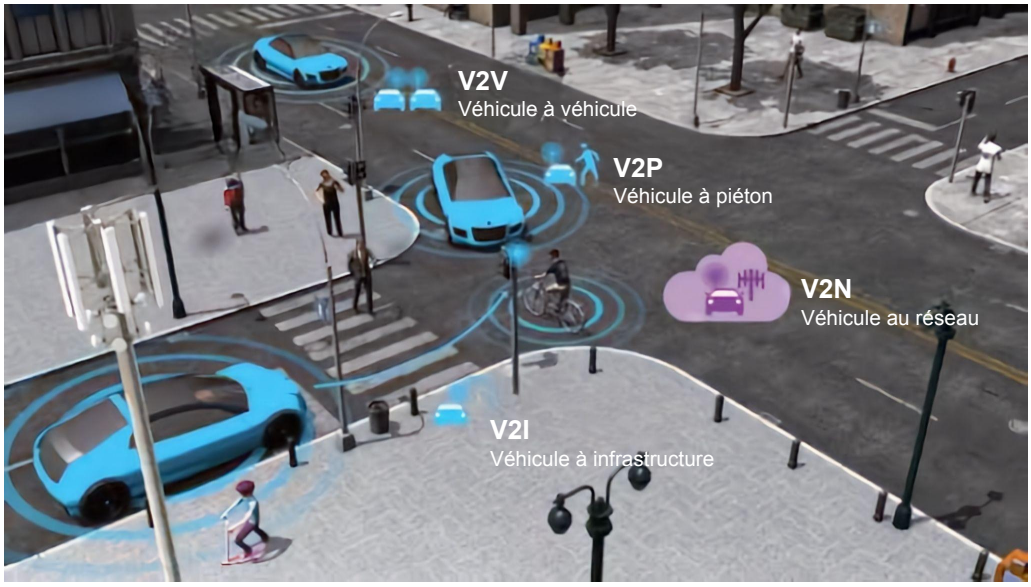


Figure 2.6 – Types de communication de véhicules

National Highway Traffic Safety Administration (NHSTA) recommande l'utilisation du *Dedicated Short Range Communications (DSRC) Performance Requirements for V2V Safety Awareness* 2018 conformément à la norme J2945/2-201810. Cette norme vise à établir des exigences de performance pour la communication véhicule-à-véhicule (V2V) dans le but d'accroître la sécurité routière.

En ce qui concerne les technologies de communication, différentes méthodes sont utilisées, en fonction de la portée de communication requise :

- Courte portée : Ces communications s'étendent sur une portée de moins de 25 mètres. Elles peuvent utiliser des technologies telles que le Bluetooth, Zigbee, et les Ultra-Wideband, la RFID (Radio-Frequency Identification), même les ondes acoustiques pour les applications sous-marines.
- Moyenne portée : Les communications de moyenne portée couvrent une plage allant de 25 mètres à 100 mètres. Les technologies couramment utilisées à cette portée incluent le Wireless Fidelity (Wi-Fi) IEEE 802.11 et le DSRC.
- longue portée : Les communications à longue portée dépassent les 100 mètres et peuvent atteindre des distances considérables. Les technologies telles que les réseaux cellulaires de la troisième génération (3GPP), la 4G-LTE, la 5G-NR, ainsi que LoRAWAN (802.15.4) qui offre une distance de transmission pouvant atteindre jusqu'à 16 km.

En outre, il convient de mentionner les technologies de communication à très courte distance tel que la NFC (Near Field Communication), même visuels comme la vision par ordinateur avec des marqueurs fiduciaires. Ces technologies permettent un échange d'informations extrêmement proche, généralement limité à quelques centimètres. Elles sont souvent utilisées dans des applications de contrôle d'accès, et la gestion des véhicules dans des zones de stationnement restreintes.

L'architecture mise en place pour permettre cette communication est similaire à celle des réseaux de communication classiques. De nombreuses questions standard dans le domaine des réseaux se posent, notamment la conception des topologies de réseau, les protocoles de communication, et leur dépendance par rapport à l'architecture déployée dans le système. Dans la littérature, une terminologie couramment utilisée est la nomenclature "VANET" (Vehicular Ad-Hoc Networks) pour désigner les réseaux ad hoc véhiculaires, une sous-catégorie émergente de réseaux mobiles ad hoc capable de créer spontanément un réseau de dispositifs mobiles/véhicules. Cette nomenclature est largement reconnue pour décrire les réseaux de communication entre véhicules. D'autres exemples incluent celles de diffusion d'information sur le réseau avec contrôle du maître classique en centralisé. D'autres étudient des systèmes distribués sous le concept de "sign-board" de Jing W., 1994 agissant comme un black-board du véhicule lui même, ou même des communications en "Dual-hop" proposant un modèle probabiliste permettant de saisir l'impact des objets statiques sur l'environnement en terme de ligne de vue (LOS) de Abbas et al., 2015. Une autre alternative des méthodes hand-crafted ou "bespoke" sont les méthodes génériques qui repose sur l'usage des méthodes d'apprentissage tel que le Graph Neural Networks (GNNs), permettant l'apprentissage de stratégies de communication explicites et une coordination multi-agents complexe, comme les travaux de Paulos et al., 2019.

En terme d'informations, les véhicules peuvent recevoir des alertes de collision, vitesse dans les virages et freinage d'urgence, des informations sur les feux de signalisation et conditions routières, assistance aux mouvements à l'intersection, ainsi que des alertes sur le contexte environnemental comme les dangers, embouteillages, météorologiques et les zones de travaux.

La mise en place d'une communication entre les véhicules offre de nombreux avantages significatifs, notamment la réduction des embouteillages, l'amélioration de l'expérience de conduite, et l'accroissement de la sécurité tant pour les passagers que pour les conducteurs. En outre, cette communication joue un rôle essentiel dans le renforcement de la sécurité routière, la diminution des congestions de trafic, et la gestion efficace du flux de véhicules sur les routes. Par exemple, en situations périlleuses telles que les accidents ou les embouteillages, les véhicules peuvent interagir les uns avec les autres et avec le réseau pour partager des informations vitales.

Cependant, le V2V présente également plusieurs limites, car à mesure que les véhicules deviennent des systèmes dépendants des technologies de l'information, ils deviennent vulnérables aux attaques ainsi qu'aux coûts supplémentaires engendrés par l'installation des systèmes V2V dans les véhicules. Pour protéger les véhicules, le domaine de la cyber-sécurité est appliqué. Ce domaine vise à garantir la sécurité, l'intégrité et la disponibilité des données et des fonctions essentielles au bon fonctionnement des véhicules autonomes. Ses applications sont souvent utilisées pour la détection d'attaques et la protection de données tel que l'usage du Machine learning ML de Alsaade et al., 2023. Une autre alternative à l'usage de ce domaine repose dans la réutilisation du matériel disponible sur le véhicule tels que les travaux de Lim et al., 2019 qui

exploitent les capteurs de perception embarqués disponibles pour authentifier les véhicules, réutilisant ainsi les composants sans aucun ajout au véhicule.

2.3.2.3 Perception d'un système multi-véhicules

Dans le contexte complexe de la circulation routière moderne, un environnement où les véhicules collaborent de manière intelligente, réagissent instantanément aux situations d'urgence, préviennent les collisions et optimisent la fluidité du trafic devient de plus en plus réalisable grâce aux avancées technologiques en matière de communication et de capteurs. Cela est faisable si on accorde aux véhicules l'aptitude à percevoir l'environnement qui les entoure afin de le comprendre et anticiper les actions des autres véhicules pour agir de manière coordonnée.

La mise en œuvre de la perception englobe une gamme diversifiée de technologies, dont certaines ont été spécifiquement conçues à cet effet, telles que les caméras et les LiDARs, tandis que d'autres ont été développées en tirant parti de la réflexion des ondes et de la propagation des signaux. Cette variété de technologies permet de couvrir un large éventail de méthodes de perception, allant de la vision par ordinateur classique à l'utilisation de capteurs basés sur la détection des ondes réfléchies, offrant ainsi des perspectives riches et complémentaires pour résoudre les défis de la perception.

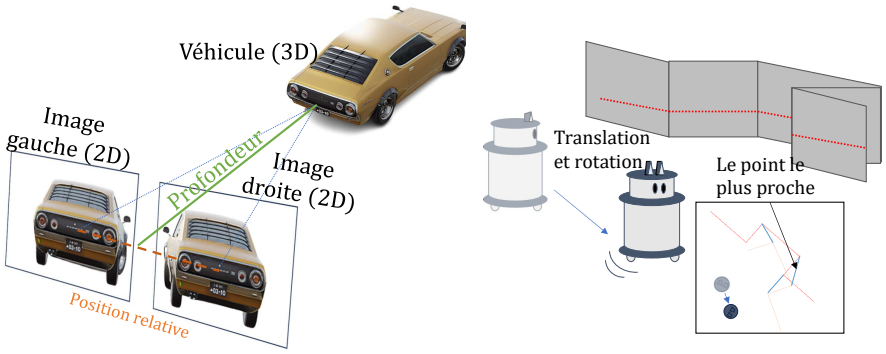
Afin d'observer l'environnement ainsi que les autres véhicules, plusieurs technologies peuvent être déployés :

1. Perception par observation d'environnement :
   — Caméra : Les caméras des véhicules peuvent être classées comme des caméras à lumière visible ou à infrarouge, en fonction de la longueur d'onde de l'appareil. La portée maximale de la caméra est d'environ 250 mètres en fonction de la qualité de l'objectif.

     Pour les caméras à lumière visible, le concept le plus utilisé est la stéréovision pour la localisation où deux caméras sont combinées avec une focale et une inter-distance connues pour générer un nouveau canal contenant l'information de profondeur (voir Fig.2.7a).
   — LiDAR : Il fournit une liste de points dans un espace tridimensionnel. Ces points sont extraits de l'angle formé par le faisceau laser et de la distance entre le capteur et l'impact. Un LiDAR 2D utilise un seul faisceau laser diffusé sur le miroir qui tourne à grande vitesse. Un LiDAR 3D peut obtenir une image 3D de l'environnement en plaçant plusieurs lasers sur la nacelle.

   Les applications des deux techniques dans le cas du mono et multi-véhicules sont vastes allant de la cartographie de l'environnement, à la localisation et cartographie simultanées (SLAM), la détection d'autres véhicules, l'odométrie télémétrique/visuelle par l'application de l'algorithme ICP (Iterative closest point), et la localisation relative. Thrun, 2001 présente une méthode de cartographie d'une équipe de robots mobiles distribuée à l'aide d'un télémètre laser avec un algorithme probabiliste permettant à une équipe de robots de générer en coopération une

carte unique de leur environnement en tenant compte des erreurs d'odométrie. MARTINELLI et al., 2005 proposent d'utiliser une webcam comme capteur extéroceptif pour extraire des informations sur la position relative à partir d'images. Ils comparent les performances de différentes observations relatives, notamment la position relative, la distance relative et l'orientation relative, et concluent que l'intégration du relèvement relatif à l'odométrie donne les meilleurs résultats en termes de réduction des erreurs d'estimation de la pose. HÉRY, E., XU, P. AND BONNIFAIT, P., 2021 présentent une méthode de localisation relative cohérente basée sur les observations LiDAR afin d'estimer la position relative entre les véhicules avec une métrique point à ligne dans le cadre d'une approche itérative du point le plus proche (ICP) (voir Fig.2.7b). Les mesures LiDAR sont utilisées pour calculer les poses relatives entre les véhicules, qui sont ensuite incorporées dans les cartes dynamiques locales échangées entre les véhicules pour la localisation coopérative. S. CHEN et FREMONT, 2021 présentent une approche de fusion de capteurs à couplage lâche capable de produire la solution odométrique en appliquant le filtrage d'intersection de covariance pour coupler la vision stéréo et l'odométrie LiDAR, en tenant compte de leurs incertitudes respectives.



(a) Stéréo vision **(b)** Algorithme ICP

Figure 2.7 – Usage de la perception par observation de l'environnement pour la localisation

2. Perception par propagation des signaux : Elle repose sur diverses métriques temporelles et de force du signal, chacune apportant une perspective unique à la perception et à la localisation, notamment le temps d'arrivée (TOA) qui mesure précisément le moment où un signal est reçu par un capteur, la force du signal reçu (RSS) qui mesure l'intensité du signal lorsqu'il atteint le capteur, et le temps de vol (ToF) qui évalue la durée nécessaire pour qu'un signal parcourt la distance entre l'émetteur et le récepteur. Ces métriques permettent de déterminer la distance entre l'émetteur et le récepteur en utilisant la vitesse de propagation du signal et l'affaiblissement du signal avec la distance, permettant ainsi la localisation. De plus, elles sont exploitées dans une variété de capteurs et

de technologies de communication sans fil, notamment Bluetooth, Wifi, RFID, UWB, Ultrasonic et RADAR. D. Jin et al., 2016 proposent un algorithme de localisation coopératif basé sur des mesures de l'intensité du signal reçu (RSS) quantifiées dans les réseaux de capteurs sans fil. Ils adoptent le cadre bien connu de l'"algorithme de la somme des produits sur un réseau sans fil" (SPAWN) pour relever le défi posé par les mesures fortement quantifiées. Ponte Müller et al., 2016 utilisent le radar comme capteur de distance embarqué pour obtenir la position relative des autres véhicules par rapport à l'*ego*-véhicule. Le capteur radar fournit des informations précises dans des situations de visibilité directe, mais ne peut pas fournir d'informations sur la position relative lorsque le faisceau radar est bloqué. Pour surmonter cette limitation, un cadre de fusion est proposé qui combine les données du capteur radar avec les informations de positionnement coopératif obtenues par le biais de la communication de véhicule à véhicule. Pierre et al., 2018 abordent le problème de la localisation des robots mobiles à l'aide de mesures de portée provenant uniquement de capteurs à bande ultra-large (UWB) peu coûteux. La solution proposée considère chaque objet statique ou mobile comme une balise dotée d'informations contextuelles. Une mesure de balise à balise est effectuée à l'aide de capteurs UWB, et l'estimation de la position est calculée par la balise cible.

3. Perception par réutilisation de connaissances : C'est une approche qui exploite des connaissances préalablement acquises pour améliorer la perception et la compréhension de l'environnement. Un exemple intéressant de cette approche est l'utilisation de la méthode Differential Global Positioning System (DGPS). Le DGPS va au-delà du système classique du GPS en utilisant des stations terrestres fixes qui connaissent leur emplacement avec une grande précision ou d'autres véhicules plus proches qui ont été corrigés. Une mise en œuvre très précoce de cette technique a été faite par Kato et al., 2002, où le DGPS est utilisé pour la localisation des véhicules dans un système de conduite coopérative. Une étude plus récente est celle de Rohani et al., 2014, où la méthode DGPS est rendue plus dynamique par l'utilisation de stations de référence mobiles au lieu de stations fixes pour générer les corrections de pseudo-distance (voir Fig.2.8).

Dans la section précédente, nous avons présenté l'historique et les usage des systèmes multi-véhicules, puis élargi la thématique de la localisation coopérative, en la différenciant de la localisation collaborative, et en définissant les approches utilisées, les manières d'établir la communication et la perception entre les véhicules d'un tel système. Dans la suite, nous présentons le problème provenant du partage de données entre les véhicules, ses conséquences et les solutions qui ont été proposées pour remédier à ce problème.

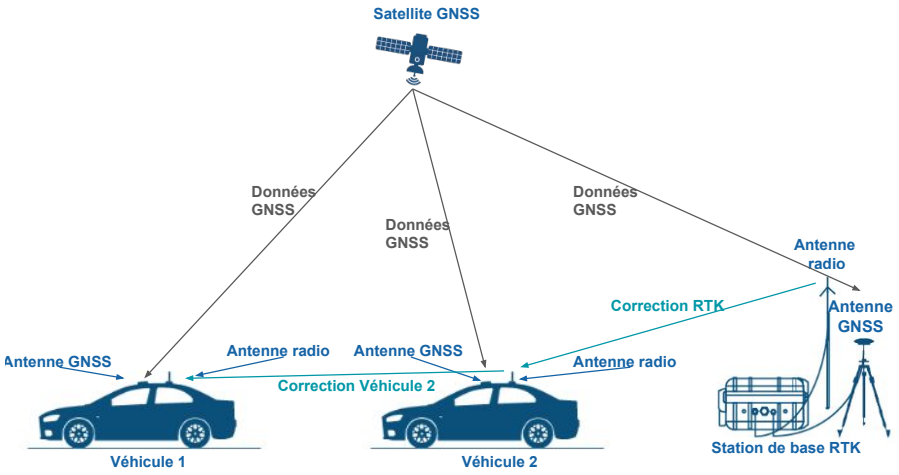


Figure 2.8 – Fonctionnement du DGPS en multi-véhicules

2.3.3 Problème de consanguinité de données

Dans un système multi-véhicules en communication, le partage de données et l'interaction des véhicules produit une intégration de mêmes données plusieurs fois, induisant des "échos" dans le système. Le problème de la consanguinité de données ou *Data incest* peut être définie comme suit :

Définition 6 *"une situation où les données utilisées présentent des redondances excessives, des corrélations fortes ou des biais inhérents".*

Selon McLaughlin et al., 2003, le problème de l'inceste des données découle du conditionnement multiple d'une estimation sur le même élément d'information (mesure). La fusion répétée de la densité optimale avec la deuxième mesure conduit à un excès de confiance dans une estimation sous-optimale qui est biaisée par rapport aux informations répétées.

2.3.3.1 Origines et conséquences

Dans le domaine de la localisation coopérative, où plusieurs appareils ou capteurs travaillent ensemble pour estimer la position d'un objet ou d'une personne, la consanguinité des données peut survenir de différentes manières :

- Partage de données entre les appareils : Lorsque les appareils coopératifs partagent leurs données de mesure, il peut y avoir une redondance dans les informations fournies. Par exemple, si deux appareils mesurent la même grandeur physique et partagent leurs données, cela peut entraîner une corrélation élevée entre les ensembles de données. Ce cas est visualisé par la Fig.2.9.
- Capteurs similaires : Si tous les appareils coopératifs utilisent des capteurs similaires pour collecter des données, il peut y avoir une similitude dans les caractéristiques des données capturées. Cela peut conduire à une consanguinité des données, car les appareils fourniront essentiellement les mêmes informations.

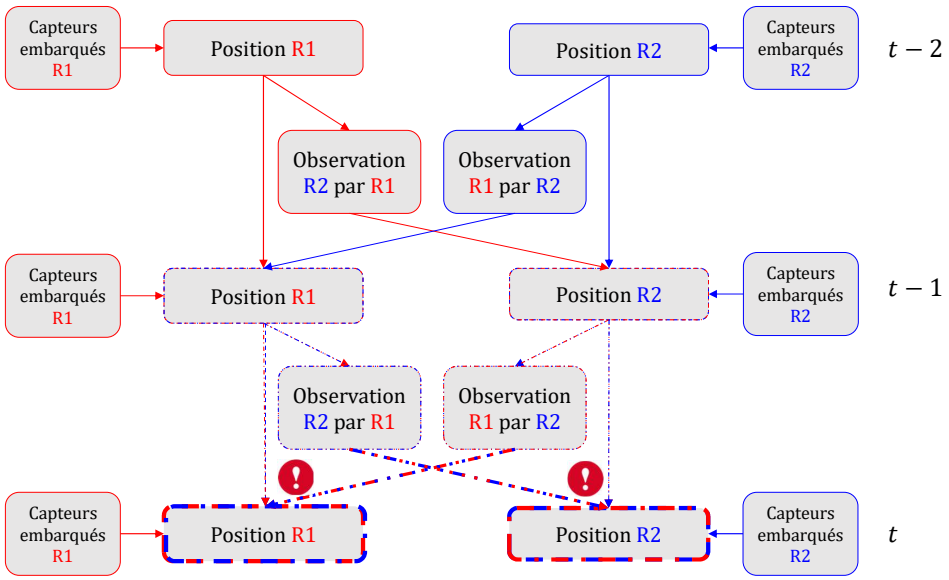


Figure 2.9 – Cas de partage de données, inspiré de KARAM, 2009

— Environnement homogène : Si les appareils coopératifs évoluent dans un environnement homogène, où les conditions et les caractéristiques de l'environnement sont similaires, cela peut entraîner une corrélation élevée entre les ensembles de données collectées.

En conséquent, comme le montre BAHR et al., 2009, même si le partage d'informations entre véhicules peut être très bénéfique, l'échange de mesures et d'estimations de l'état peut également être dangereux en raison du risque que les mesures soient utilisées par un véhicule plus d'une fois ; une telle réutilisation des données conduit à des estimations incohérentes (trop confiantes), ce qui rend l'association des données et le rejet des valeurs aberrantes plus difficiles et les divergences plus probables.

Les effets de la consanguinité des données dans le domaine de la localisation coopérative peuvent inclure :

— une diminution de la précision et de la fiabilité des estimations de position,
— une vulnérabilité accrue aux erreurs systématiques,
— une dégradation des performances dans des situations nouvelles ou imprévues,
— une intégration de la même information plusieurs fois résulte par une sur-convergence vers une solution incorrecte.

Ces problèmes diminuent la performance du système et produisent une divergence dans son comportement, et doivent être étudiés lors de la conception de la modélisation du système. Dans la suite, nous présenterons les solutions proposées dans la littérature.

2.3.3.2 Solutions proposées

Afin de résoudre le problème de la consanguinité des données, plusieurs aspects peuvent être étudiés et traités.

Initialement, HOWARD, MATARIC et al., 2003 examine le problème de la surconvergence, en disposant un arbre de dépendance pour mettre à jour l'historique de la dépendance des distributions.

En termes d'architecture et de communication, KARAM, 2009 propose une approche par échange d'états. Elle vise à combiner les données de plusieurs robots mobiles communicants pour mettre à jour et maintenir une carte optimale de l'ensemble de la flotte et améliorer l'estimation de la pose. L'approche implique l'échange de l'estimation de l'état de chaque robot avec les autres membres du groupe. La fusion de ces estimations est effectuée dans chaque robot, en veillant à ce qu'il n'y ait pas d'incohérence dans les données. L'approche tient compte des contraintes liées aux capteurs et à la communication et est conçue pour fonctionner en temps réel dans des environnements ouverts.

En terme d'estimation d'état, LI, H. et al., 2013 proposent une méthode pour la localisation qui utilise le filtre d'intersection à covariance divisée, qui aborde la question de la corrélation entre les estimations. Elle y parvient en maintenant une estimation d'un état de groupe décomposé pour chaque véhicule et en partageant cette estimation avec les véhicules voisins. L'estimation est mise à jour en utilisant à la fois les données des capteurs de l'*ego*-véhicule et les estimations envoyées par les autres véhicules, sur la base du filtre d'intersection à covariance divisée. Cette méthode permet d'incorporer et de maintenir des informations indépendantes connues dans les estimations, évitant ainsi le problème de surconvergence causé par la corrélation entre les estimations. Cette méthode a été développée par la création de plusieurs filtres basés sur ce filtre afin d'obtenir une meilleure estimation. Parmi ces variantes on trouve le filtre d'intersection de covariance divisé, le filtre inverse d'intersection de covariance, le filtre d'intersection de covariances par lots, et bien d'autres. Le détails de ces filtres peuvent être trouvés à la section 2.

En termes de données, LASSOUED, 2016, propose une solution au problème de l'incohérence des données en utilisant une méthode appelée Set Inversion Via Interval Analysis (SIVIA). SIVIA est une méthode d'estimation de l'appartenance à un ensemble qui fournit une solution fiable au problème de l'incohérence des données en donnant des ensembles qui contiennent toujours la véritable position des véhicules sans surconvergence.

Une autre approche efficace pour résoudre cette problématique consiste à mettre en place un diagnostic des informations échangées entre les véhicules avant leur intégration complète dans le système. Cette démarche repose sur une comparaison minutieuse des données transmises par les véhicules avec les corrections fournies par d'autres capteurs ou sources d'information. En effectuant cette comparaison, il devient possible de détecter rapidement les incohérences ou les anomalies dans les données provenant des véhicules. De plus, en utilisant des méthodes avancées d'analyse de données et de fusion sensorielle, il est possible d'identifier et de corriger ces incohérences de manière proactive, contribuant ainsi à la fiabilité et à la précision du système global.

Dans le cadre de cette thèse, nous adoptons deux méthodes, la méthode à base de filtre d'intersection de covariances CIF et celle du diagnostic.

En résumé, cette section explore les aspects relatifs aux systèmes multi-véhicules, notamment leur localisation, leur mode de coopération, ainsi que les répercussions de la mutualisation des données. Comme démontré dans la section actuelle, le processus de diagnostic revêt une importance cruciale au sein des systèmes multi-véhicules. Son objectif premier est de prévenir l'apparition de défauts au sein de l'un des véhicules et, de manière tout aussi importante, d'empêcher la propagation de ces défauts vers les autres véhicules du système.

Dans le paragraphe suivant, nous aborderons le traitement de défauts, par le biais de la surveillance de la localisation, en mettant en avant ses impératifs et la terminologie associée. En outre, nous examinerons en détail les diverses méthodes de diagnostic répertoriées dans la littérature, en accordant une attention particulière aux approches basées sur les modèles et à celles guidées par les données.

2.4 Supervision de la localisation

L'augmentation de l'usage des robots et des véhicules autonomes dans diverses industries a modifié la perception de l'automatisation et de la mobilité. Ces technologies de pointe présentent un immense potentiel pour améliorer l'efficacité et la productivité. Toutefois, leur bon fonctionnement repose fortement sur leur sécurité et la précision de leur positionnement dans leur environnement. Cette préoccupation découle des tragédies survenues sur les routes. Selon une récente étude de l'administration nationale de la sécurité routière (NHTSA), en 2020, on a recensé 38 824 décès et 2,282,015 blessures résultant d'accidents de la route signalés aux États-Unis, comme l'indique le rapport National Highway Traffic Safety Administration, U.S. Department of Transportation, 2020.

Cela justifie la nécessité de surveiller attentivement le comportement des véhicules autonomes afin d'améliorer leur performance par comparaison du comportement humain. En effectuant une surveillance constante des décisions et des actions des véhicules autonomes, il est possible de détecter rapidement d'éventuelles erreurs et d'intervenir si nécessaire pour prévenir des situations dangereuses. Cette approche peut contribuer significativement à renforcer la sécurité routière en réduisant les risques de collisions et d'incidents associés à la conduite, tout en encourageant une adoption plus étendue de la technologie autonome prometteuse. La faisabilité de cet objectif est soutenue par les performances des véhicules autonomes récemment déployés à San Francisco en août de cette année. Cette implémentation avait un accident tous les 60,000 miles (environ 96,560.64 kilomètres), soit environ cinq ans de conduite pour un automobiliste humain typique, selon un article récent de T. B. Lee, 2023, renforcent l'optimisme quant à l'avenir de cette technologie.

Dans cette section, nous mettons en lumière les différentes terminologies de supervision, avec un accent particulier sur les méthodes

de diagnostic. Notre objectif est de développer des solutions qui non seulement assurent la surveillance des systèmes, mais qui vont au-delà en visant la création de dispositifs tolérants aux défaillances. Cette démarche est cruciale dans le contexte de l'ingénierie, car elle vise à garantir la fiabilité et la continuité des opérations, même en présence de problèmes ou de défaillances potentielles.

2.4.1 Vocabulaire et terminologie

Afin de viser les bonnes applications, nous fournissons les définitions admises lors de nos recherches qui sont les suivantes :

- **Surveillance** : Surveillance d'un système physique et prise de mesures appropriées pour maintenir le fonctionnement en cas de défaillance, selon IFAC Technical Committee on Control Education, 2006. Pour les processus techniques, elle vise à montrer l'état actuel, à indiquer les états indésirables ou non autorisés et à prendre les mesures appropriées pour éviter les dommages ou les accidents. Les écarts par rapport au comportement normal du processus résultent de *défauts* et d'erreurs, qui peuvent être attribués à de nombreuses causes. Ils peuvent entraîner des périodes plus ou moins longues de *dysfonctionnement* ou de *défaillance* si aucune mesure n'est prise pour y remédier. L'une des raisons de la supervision est d'éviter ces dysfonctionnements ou défaillances, selon Isermann, 2006.
  - **Défaut** : Un défaut est une déviation non autorisée d'au moins une propriété caractéristique du système par rapport à l'état acceptable, habituel ou standard, Isermann, 2006.
  - **Dysfonctionnement** : Un dysfonctionnement est une irrégularité intermittente dans l'accomplissement de la fonction souhaitée d'un système, Isermann, 2006.
  - **Défaillance** : Une défaillance est une interruption permanente de la capacité d'un système à remplir une fonction requise dans des conditions de fonctionnement spécifiques, Isermann, 2006.
  - **Erreur** : Écart entre une valeur mesurée ou calculée (d'une variable de sortie) et la valeur réelle, spécifiée ou théoriquement correcte, comme définie par IFAC Technical Committee on Control Education, 2006
  - **Détection des défauts** : Détermination des défauts présents dans un système et du moment de leur détection ;
  - **Isolation des défauts** : Détermination du type, de l'emplacement et du moment de la détection d'un défaut par l'évaluation des symptômes. Suit la détection des défauts ;
  - **Identification des défauts** : Détermination de la taille et du comportement variable dans le temps d'un défaut.
  - **Diagnostic des défauts** : Le terme "diagnostic" est d'origine grecque et signifie "la détection et la détermination d'une maladie". Pour les processus technologiques, la "maladie" correspond aux

"perturbations" qui affectent le processus de manière négative, selon SZAFARCZYK, 1994. Le diagnostic des défauts consiste à déterminer le type, la taille et l'emplacement du défaut le plus probable, ainsi que son moment de détection ISERMANN, 2006.

- — **Pronostic des défauts** : Le terme pronostic est basé sur le mot grec "progignôskein" qui signifie "connaître à l'avance", qui comporte l'estimation fiable de l'état de performance futur d'un système, basée sur une évaluation continue de la santé, basée sur des observations directes ou indirectes de la dégradation du système avant défaillance.
- — **Tolérance aux fautes** : La tolérance décrit la notion d'essayer de contenir les conséquences des défauts et des défaillances de manière à ce que les composants restent fonctionnels. Cela signifie que les défauts sont compensés de telle sorte qu'ils n'entraînent pas de défaillance du système.

— **Disponibilité** : Probabilité qu'un système ou un équipement fonctionne de manière satisfaisante et efficace à tout moment ISERMANN, 2006.

— **Intégrité** : L'intégrité d'un système est sa capacité à détecter des erreurs dans son propre fonctionnement et à en informer un opérateur humain ISERMANN, 2006.

— **Sécurité** : Capacité d'un système à ne pas causer de danger pour les personnes, les équipements ou l'environnement ISERMANN, 2006.

2.4.2 Méthodes de diagnostic

Les systèmes de détection des défauts sont la forme la plus simple des systèmes de diagnostic des défauts qui déclenchent des signaux d'alarme pour indiquer l'apparition des défauts. Selon BASSEVILLE et al., 1993, la question du diagnostic ou de l'isolation concerne la détermination de l'origine du changement, une fois que le changement du paramètre multidimensionnel a été détecté.

Ces systèmes de diagnostic peuvent être classés en plusieurs catégories, en fonction de leurs méthodes de détection et de leurs types. Ces méthodes sont groupées dans la Fig.2.10, et discutées dans la suite de cette section.

1. À base de modèle : L'exploration du diagnostic des défaillances basé sur des modèles a pris son essor au début des années 1970. Ce mouvement a été fortement influencé par l'émergence de la théorie de l'observateur à cette époque. La toute première méthode de détection des défaillances basée sur un modèle, connue sous le nom de filtre de détection des défaillances, a été conceptualisée par Beard et Jones. Depuis cette époque, la théorie et la pratique du diagnostic des défaillances basé sur un modèle ont connu une croissance dynamique et rapide. Actuellement, elles se sont épanouies pour devenir un domaine crucial au sein de la théorie et de l'ingénierie du contrôle automatique.

   Dans son article de synthèse de renommée, FRANK, 1990 a classé les résultats majeurs obtenus au cours des quinze premières années dans le domaine de la technique de détection et isolation de défauts FDI

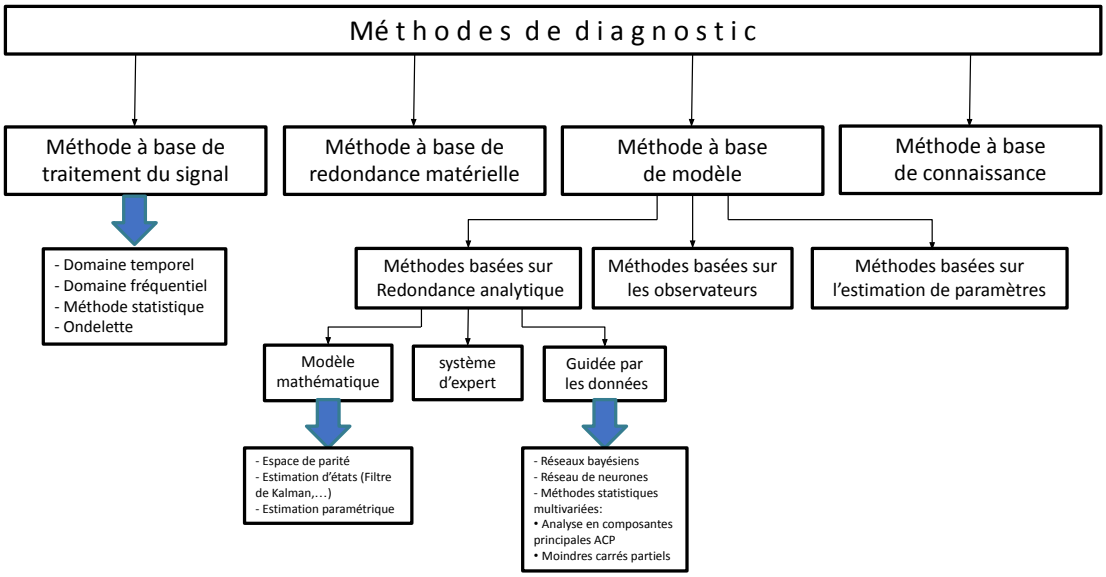


Figure 2.10 – Méthodes de diagnostic inspiré de DING, 2013

basée sur un modèle en méthodes basées sur l'observation - les méthodes basées sur l'espace de parité et - les méthodes basées sur l'identification des paramètres.

— **Les observateurs** : Un système permettant d'obtenir les estimations de variables inconnues (telles que les variables d'état) est appelé un *observateur* :

Définition 7 *un observateur est un système dynamique prenant pour entrée les signaux connus du système sur lequel il est implanté et dont les sorties convergent vers une estimation des variables d'état (ou d'une partie des variables d'état) selon* CHRISTOPHE, *2001.*

Ces observateurs sont l'origine du discours concernant le Problème Fondamental de la Génération des Résidus (PFGR).

Les observateurs sont utilisés dans de nombreuses applications robotiques comme pour le remplacement de capteurs physiques, la prédiction des états internes, le contrôle robuste et la génération de résidus pour des fins de diagnostic. Wünnenberg et Frank ont proposé le premier schéma de génération de résidus à entrée inconnue en 1987. Ces résidus sont obtenus par la comparaison entre la valeur estimée et la valeur correspondante mesurée grâce aux capteurs (voir Fig.2.11).

Chaque observateur de la banque est synthétisé pour être sensible à un sous-ensemble de fautes et insensible aux autres. La détection et la localisation des pannes, nécessitent de pouvoir découpler les vecteurs d'entrée/sortie en correspondance avec le défaut.

Deux formes d'architecture existent dans la littérature :

(a) Dedicated Observer Scheme (DOS) : structure d'observateurs dé-

diés, où chaque entrée/sortie possède son observateur dédié, intégrant uniquement son information.

(b) Generalized Observer Scheme (GOS) : structure d'observateurs généralisés, où toutes les entrées/sorties sont intégrées dans un observateur sauf une à la fois, puis deux et ainsi de suite.

La différence entre DOS et GOS est que DOS est une banque d'observateurs sensibles à un défaut alors que le GOS est composé d'observateurs sensibles à tous les défauts sauf un.

Les figures 2.12 et 2.13 présentent la forme de la banque d'observateurs à entrée inconnue dédiée et généralisée pour les deux types de défauts capteurs et actionneurs.

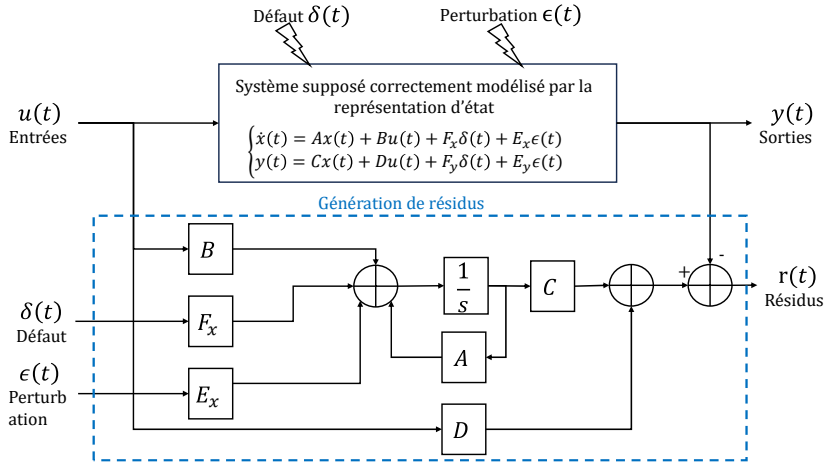


Figure 2.11 – Forme de l'observateur à entrée inconnue

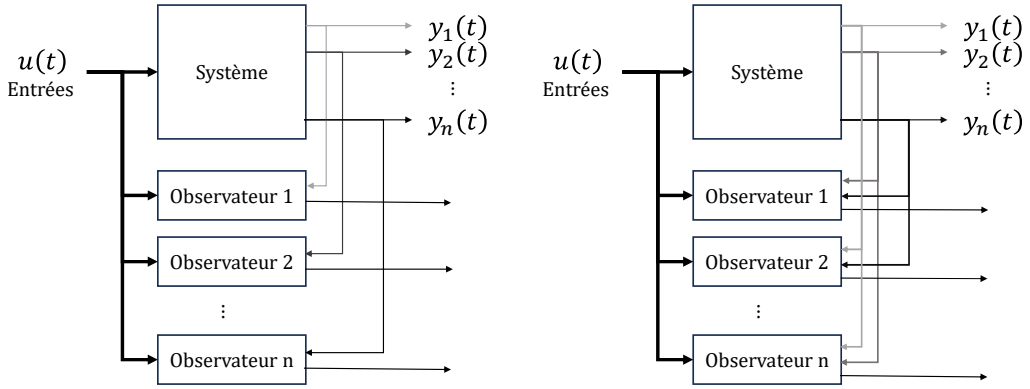


Figure 2.12 – DOS/GOS pour l'isolation des défauts capteurs

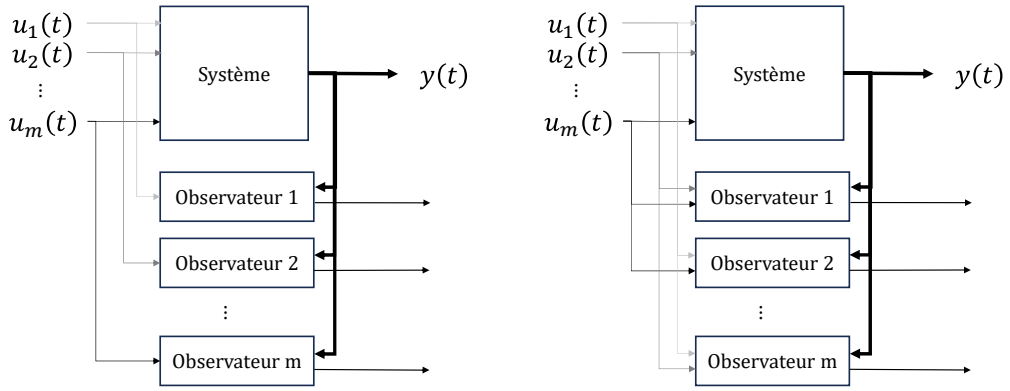


Figure 2.13 – DOS/GOS pour l'isolation des défauts actionneurs

Comme l'explique Ding, 2013, le rôle le plus important de l'observateur dans un système FDI est de rendre les signaux résiduels générés indépendants des signaux d'entrée du processus et des conditions initiales du processus.

— **Espace de parité et relations de redondance analytique :** Le diagnostic basé sur l'analyse de relations de parité est une méthode naturelle de diagnostic qui consiste à vérifier la consistance entre les différentes variables d'un système. Cela conduit à une modélisation du système, réalisée soit d'une façon quantitative (modèle mathématique connue *a priori*) soit qualitative (exploiter la connaissance sur le processus de systèmes experts) ou guidées par les données qui reposent sur l'historique sans nécessité d'une connaissance approfondie du système physique. Elle est fondée sur la construction de variables à partir de deux sources a minima (modèles, capteurs, entrées et sorties). Une comparaison de ces variables conduit à la génération de résidus. Cette approche nécessite peu de puissance de calcul. Cependant, un post-traitement est obligatoire pour obtenir des résultats satisfaisants. Le seuillage simple est sensible au bruit. Un schéma montrant la génération des relations de redondance analytiques est présenté dans la Fig.2.14.

— **Estimation de paramètres :** Cette méthode se fonde sur l'idée que les anomalies se manifestent par des altérations dans les caractéristiques physiques du système. Afin de détecter ces anomalies, les paramètres du système sont continuellement estimés en utilisant des techniques bien établies d'estimation des paramètres. Les résidus dans cette approche correspondent essentiellement à la disparité entre les estimations en temps réel des paramètres du système et leurs valeurs attendues en l'absence d'anomalies. Initialement déve-

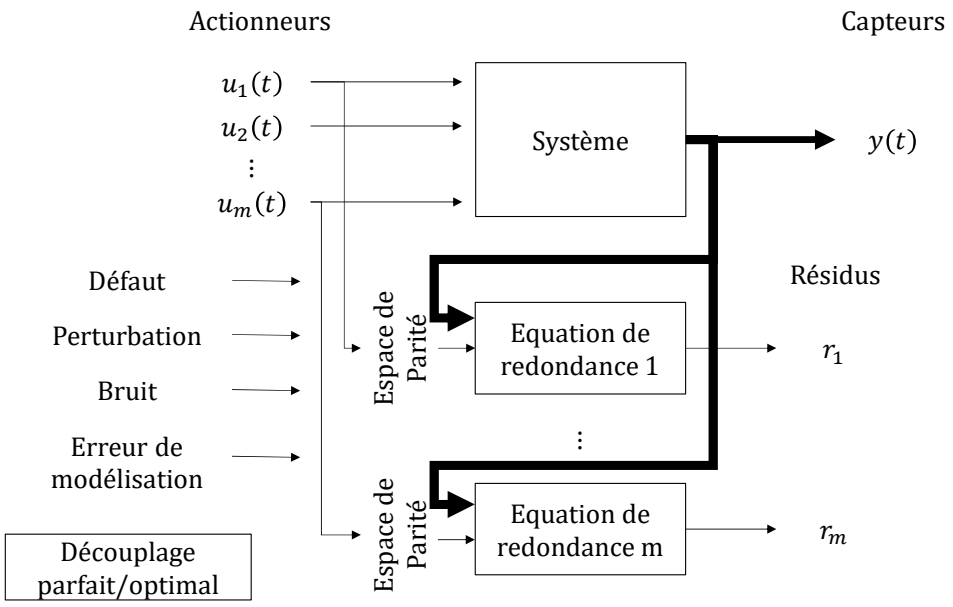


Figure 2.14 – Génération des relations de redondance

loppée pour les systèmes linéaires en raison de la disponibilité de méthodes d'estimation des paramètres linéaires bien connues, cette approche a évolué pour prendre en compte les systèmes dynamiques non linéaires grâce aux progrès qui a eu lieu dans les méthodes d'estimation des paramètres non linéaires, telles que l'utilisation de filtres de Kalman non linéaires UKF et de réseaux de neurones adaptatifs.

2. À base de redondance matérielle :

   L'idée la plus basique pour surveiller un capteur est la duplication de ce capteur ou l'ajout d'autres capteurs pour mesurer la même variable pour pouvoir détecter et isoler le capteur défectueux en vérifiant la cohérence des données. Pour utiliser cette méthode, le système doit pouvoir supporter les capteurs, triplés ou quadruplés. La redondance matérielle apparaît ainsi comme un cas particulier de la redondance analytique statique.

   L'avantage de cette méthode apparaît dans sa fiabilité et sa simplicité. Par contre, le coût élevé résultant de l'utilisation de capteurs redondants limite le domaine d'application de cette méthode, et peut rapidement surcharger le système.

3. À base de la théorie des graphes :

   Comme définie par Balakrishnan et al., 2012, un graphe est un triplet ordonné d'un ensemble non vide d'arêtes, un autre ensemble disjoint de sommets et de relations d'incidence qui associent à chaque arête une paire de sommets. Dans le contexte du diagnostic des processus, les

graphes permettent de représenter visuellement les relations entre les éléments d'un système, ce qui peut faciliter l'identification des anomalies et des problèmes potentiels. L'un des avantages clés de l'utilisation de la théorie des graphes dans le diagnostic des processus est sa capacité à représenter de manière claire et concise des relations complexes entre les entités. Les anomalies ou les problèmes dans un processus se traduisent souvent par des comportements anormaux ou des ruptures dans les schémas d'interactions.

4. À base de traitement de signal :

   Les méthodes à base de traitement de signaux supposent que certains signaux comportent des informations sur les défauts sous forme de symptômes. Un traitement approprié du signal dans le domaine temporel ou dans le domaine fréquentiel permet d'atteindre le diagnostic.

5. Guidé par les données :

   Pour les processus à grande échelle, tels que les usines chimiques, le développement de méthodes de détection des défauts basées sur des modèles nécessite un effort considérable et finalement trop élevé. Les méthodes d'analyse basées sur les données offrent alors une alternative, selon ISERMANN, 2006.

   Ces méthodes sont intéressantes lorsque les mesures de processus disponibles sont fortement corrélées mais que seul un petit nombre d'événements (défauts) produit des modèles inhabituels. Lorsque les données du processus sont fortement corrélées, les données originales du processus peuvent être projetées sur un plus petit nombre de composantes principales (ou variables latentes), réduisant ainsi la dimension des variables.

2.4.3 Diagnostic à base de modèle et théorie de l'information

Lorsque l'on examine un système dynamique non linéaire qui est influencé par un environnement non modélisable, il peut être difficile de mesurer avec précision les variations de l'entrée et des perturbations qu'il subit. Par conséquent, l'observation de cet environnement ne reflète pas toujours fidèlement la réalité, et les mesures ne sont pas déterministes, mais plutôt sujettes à une certaine incertitude. En conséquence, le système est abordé dans un cadre stochastique non déterministe, ce qui conduit à des distributions de probabilités plutôt qu'à des valeurs déterministes. L'analyse de ces distributions de probabilités peut être réalisée efficacement à l'aide de la théorie de l'information.

Dans un contexte de système de localisation, comme expliqué précédemment, l'objectif principal réside dans l'estimation de l'état, une étape essentielle pour anticiper et planifier le comportement futur du véhicule. Toutefois, cette estimation est vulnérable aux perturbations provenant de l'environnement, aux défauts matériels potentiels, ou encore aux interactions complexes entre les capteurs au sein du même système. Dans une perspective d'analyse statistique, cela se traduit par un passage d'un comportement de mesure du capteur à un

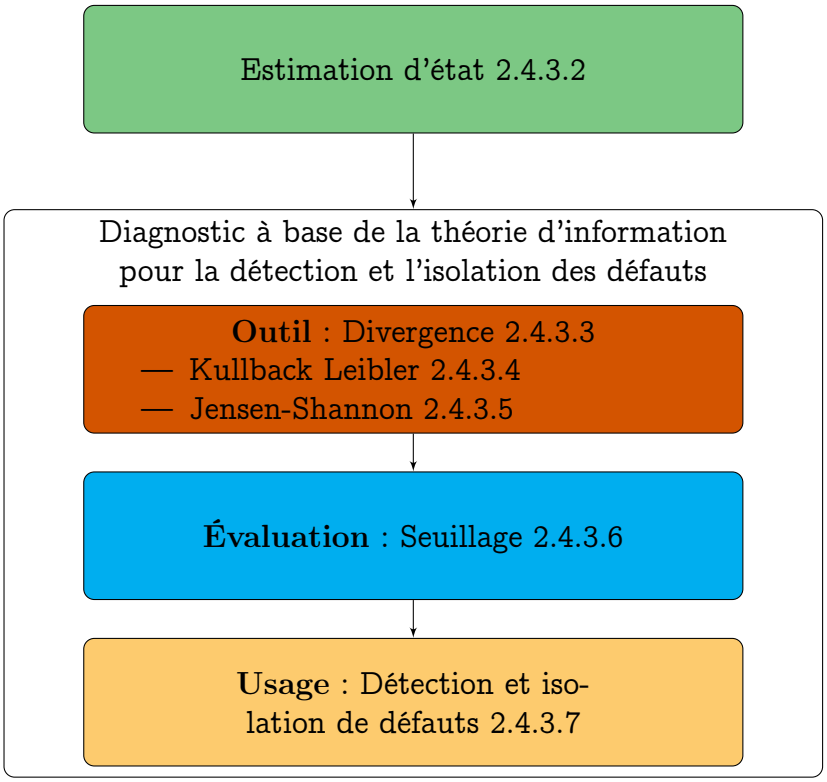


Figure 2.15 – Étapes de la méthode basée sur la théorie d'information

comportement qui ne s'aligne pas de manière homogène avec l'ensemble du système. Ces événements sont généralement rares. L'association entre la rareté et les défauts repose sur l'hypothèse que la plupart des défauts ou anomalies dans un système ne sont pas des occurrences courantes. Au contraire, ils représentent des déviations par rapport au comportement anticipé, les classant ainsi parmi les événements relativement peu fréquents dans l'historique opérationnel du système. L'identification et la caractérisation de ces événements rares sont d'une importance capitale, car ils servent souvent de signaux précurseurs de problèmes potentiels qui, s'ils ne sont pas pris en considération, peuvent conduire à des problèmes plus substantiels, à des défaillances du système ou à des risques pour la sécurité.

C'est à ce stade que la pertinence du diagnostic se révèle cruciale, car il consiste à détecter de manière appropriée l'apparition de défauts.

Pour évaluer la similitude entre différentes distributions de probabilité, on fait appel à des mesures de divergence, qui reposent sur les entropies et leurs dérivés, notamment la divergence de Kullback-Leibler et sa forme symétrique, la divergence de Jensen-Shannon. Ces métriques sont utilisées pour quantifier la dissimilarité entre deux distributions de probabilité et exigent la définition des plages de fonctionnement normales ainsi que des seuils au-delà desquels un défaut est susceptible de se produire. En de telles circonstances, afin d'assurer la continuité du fonctionnement du système, une décision doit être prise concernant l'entité responsable de cette erreur.

Dans la suite de cette section, nous abordons les diverses étapes de cette

méthode selon la structuration proposée dans Fig.2.15.

2.4.3.1 Historique de la méthode

La synthèse et le développement de la méthodologie mise en œuvre dans cette thèse font suite à de nombreux travaux en théorie de l'information.

Un bref historique de l'évolution de la thématique dans l'équipe ToSyMA est ici relaté afin de positionner ce travail par rapport à l'historique de la thématique dans l'équipe.

Aït Tmazirte et al., 2012 s'est intéressé au couplage serré entre des mesures GNSS et l'odométrie à l'aide d'un filtre d'informations. Ensuite, il a développé une approche basée sur un formalisme purement informationnel : filtre informationnel d'une part pour la fusion, et outils de la théorie de l'information d'autre part pour le diagnostic. Al Hage, 2016 a ainsi développé des résidus basés sur la divergence de Kullback-Leibler permettant la détection et l'exclusion des défauts capteurs, via des méthodes optimisées de seuillage.

Ces méthodes de diagnostic sont essentielles pour la localisation mais aussi pour la commande. Ainsi, dans sa thèse, Boussad, 2019 a développé de nouveaux algorithmes qui contribuent à rendre le système de navigation robuste et tolérant aux défauts capteurs et également aux défauts actionneurs. Ceci permet alors un retour de la localisation sur la commande et donc de travailler en boucle fermée.

Plusieurs divergences présentes dans la littérature ont ensuite été étudiées pour trouver un formalisme permettant d'être adaptatif au contexte de navigation et aux changements d'exigences opérationnelles. Ceci a été réalisé par le biais d'une métrique informationnelle : l'α-*Rényi divergence*, étudiée largement par Makkawi, 2020 dans sa thèse et qui permet de généraliser les métriques habituellement utilisées (en faisant varier le paramètre α). Cette divergence permet la conception de résidus paramétriques qui tiennent compte du changement d'environnement et donc du changement de la probabilité *a priori* de faire face ou non à un défaut de mesures GNSS. Ce paramètre α a été étudié et optimisé par Harbaoui, 2022 par l'usage de l'apprentissage profond pour un couplage serré GNSS - odométrie.

Cet héritage de connaissance sur le formalisme informationnel pour une localisation tolérante aux défauts a abouti à l'extension de ces travaux antérieurs afin de prendre en compte également l'aspect multi-véhicules, avec l'incorporation de l'intelligence artificielle dans la méthode.

2.4.3.2 Outils fondamentaux pour l'estimation d'état

La tâche consistant à extraire des valeurs d'état à partir de mesures imprécises, incertaines et bruitées est appelée estimation d'état. L'objectif principal est de minimiser l'erreur d'estimation lorsqu'elle est projetée dans l'espace de sortie.

1. Les réseaux bayésiens

   Le réseau bayésien (RB) est un modèle graphique probabiliste qui permet de traiter efficacement divers problèmes d'incertitude basés sur la représentation et l'inférence d'informations probabilistes. Selon CAI, L. HUANG et al., 2017, l'intérêt du réseau bayésien réside en trois raisons. Premièrement, il est cohérent et représente et définit complètement une distribution de probabilités unique sur les variables du réseau. Deuxièmement, le réseau est modulaire ; sa cohérence et son exhaustivité sont assurées par des tests localisés, qui ne s'appliquent qu'aux variables et à leurs causes directes. Troisièmement, RB est une représentation compacte, car il permet de spécifier une distribution de probabilité de taille exponentielle à l'aide d'un polynôme de probabilités.

   Thomas Bayes a introduit la célèbre règle de Bayes pour l'inférence statistique, qui fournit la formule de base pour les méthodes d'estimation bayésienne.

   Les réseaux bayésiens statiques, sont des modèles graphiques acycliques dirigés probabilistes. Ils utilisent des nœuds pour représenter les variables, des arcs pour signifier les dépendances directes entre les nœuds liés et des probabilités conditionnelles pour quantifier les dépendances. Les réseaux bayésiens statiques sont largement utilisés dans l'évaluation de la fiabilité, et de nombreuses monographies ont présenté les réseaux bayésiens en détail.

   Ils sont définis mathématiquement selon CAI, LIU et al., 2020 comme suit.

   Pour n variables aléatoires X_1, X_2,...,X_n et un graphe acyclique dirigé à n nœuds, parmi lesquels le nœud j $(1 \leq j \leq n)$ est associé à la variable X_j, le graphe est le RB représentant les variables X_1, X_2,...,X_n dans l'équation suivante :

$$P(X_1, X_2, \ldots, X_n) = \prod_{j=1}^{n} P(X_j | parent(X_j)) \tag{2.1}$$

   où les parents (X_j) représentent l'ensemble de toutes les variables X_i et où un arc relie le nœud i au nœud j dans le graphe.

2. Filtres gaussiens :

   Les filtres gaussiens sont une classe de filtres bayésiens utilisés pour l'estimation d'état dans les systèmes dynamiques stochastiques. Ils reposent sur l'hypothèse que le bruit et les incertitudes du système peuvent être modélisés par des distributions gaussiennes. Historiquement, les filtres gaussiens constituent les premières implémentations réalisables du filtre de Bayes pour les espaces continus. L'objectif principal des filtres gaussiens est d'extraire des valeurs précises de l'état et des paramètres à partir de mesures partielles et bruitées tout en maintenant la robustesse vis-à-vis des incertitudes de modélisation.

   Selon AFSHARI et al., 2017, les filtres gaussiens peuvent être classés en plusieurs groupes en fonction de leurs applications pour l'estimation

de l'état. Ces groupes comprennent le filtrage optimal linéaire, le filtrage non linéaire, le filtrage adaptatif et le filtrage robuste. Chaque groupe utilise des techniques et des algorithmes différents pour estimer l'état du système.

Le concept principal des filtres gaussiens consiste à calculer la fonction de Densité de probabilité postérieure de l'état sur la base des informations disponibles. Cela implique des calculs récursifs de la moyenne et de la covariance de la fonction de densité de probabilité conditionnelle de l'état.

— **Filtre de kalman classique** :

Le filtre de Kalman KF développé par Rudolf KÀLMÀN dans R. KALMAN, 1960 et R. E. KALMAN et al., 1961 est un filtre bayésien sous l'hypothèse gaussienne et modèles linéaires. C'est un estimateur récursif qui estime les états du système à partir des mesures bruitées tout en minimisant l'erreur quadratique moyenne. Ses étapes sont :

(a) Initier la moyenne et la matrice de covariance des états.
Puis, pour chaque étape pas du temps :

(b) Prédire l'état par Eq(2.4) et sa matrice de covariance par Eq(2.5).

(c) Corriger les valeurs prédites par les mesures des capteurs Z_k par Eq(2.6) pour l'état et Eq(2.7) pour la covariance.
Soit un système linéaire qui a la représentation d'état suivante :

$$\begin{cases} X_{k+1} = F_k X_k + B_k u_k + \omega_k \text{ avec } k > 0 \\ Z_k = H_k X_k + \nu_k \end{cases} \tag{2.2}$$

tel que :

$$E\left[\begin{pmatrix} \omega_k \\ \nu_k \end{pmatrix} \begin{pmatrix} \omega_k^T & \nu_k^T \end{pmatrix}\right] = \begin{bmatrix} Q_k & 0 \\ 0 & R_k \end{bmatrix} \tag{2.3}$$

— X_k est le vecteur d'état
— F_k est la matrice d'état
— B_k est la matrice d'entrée
— u_k est le vecteur d'entrée
— ω_k est le bruit du modèle considéré comme étant un bruit blanc gaussien de valeur moyenne nulle et de matrice de covariances Q_k
— Z_k est le vecteur d'observations
— H_k est la matrice d'observations
— ν_k est le bruit associé à l'observation Z_k, considéré comme étant un bruit blanc gaussien de moyenne nulle et de covariances R_k. Les bruits ω_k et ν_k sont indépendants entre eux.

(a) **Étape de Prédiction** L'estimation de l'état $X_{k|k-1}$ et de la matrice de covariances $P_{k|k-1}$ du système se réalisent de la façon suivante :

$$X_{k|k-1} = F_{k-1} X_{k-1/k-1} + B_{k-1} u_{k-1} \tag{2.4}$$

$$P_{k|k-1} = F_{k-1} P_{k-1/k-1} F_{k-1}^T + Q_{k-1} \tag{2.5}$$

(b) **Étape de Correction**

$$X_{k|k} = X_{k|k-1} + K_k(Z_k - H_k X_{k|k-1}) \tag{2.6}$$

$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^T = (1 - K_k H_k) P_{k|k-1} \tag{2.7}$$

avec :

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{2.8}$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{2.9}$$

Où : $[Z_k - H_k X_{k|k-1}]$ est l'innovation, S_k est la matrice de covariances des innovations, K_k est le gain de Kalman.

— **Filtre de kalman non linéaire** :

Le filtre de Kalman standard est adapté à la fusion de données des capteurs, mais limité aux systèmes linéaires. Ce qui nécessite une extension du filtre pour outrepasser cette limitation. Ce nouveau filtre, le filtre de kalman étendu (Extended Kalman Filter (EKF)), possède les mêmes étapes de prédiction et correction que le filtre standard. En complément, une étape de linéarisation y est introduite. Les matrices d'observation et de transition sont remplacées dans l'algorithme par leurs Jacobiennes. Soit le système non linéaire de la forme :

$$X_{k+1} = f(X_k, u_k) + \omega_k \tag{2.10}$$

$$Z_k = h(X_k) + \nu_k \tag{2.11}$$

(a) **Étape de Prédiction**

$$X_{k|k-1} = f(X_{k-1/k}, u_{k-1}) = X_{k-1|k-1} + B_{k-1} u_{k-1} \tag{2.12}$$

$$P_{k|k-1} = F_{k-1} P_{k|k-1} F_{k-1}^T + Q_{k-1} \tag{2.13}$$

Où F_k représente la matrice Jacobienne de f(.) :

$$F_{k-1} = \frac{\partial f}{\partial X}|_{X = X_{k-1/k-1}} \tag{2.14}$$

Et B_k est la matrice Jacobienne de f(.) par rapport à l'entrée u_k :

$$B_{k-1} = \frac{\partial f}{\partial u}|_{u = u_{k-1}} \tag{2.15}$$

(b) **Étape de Correction**

$$X_{k|k} = X_{k|k-1} + W_k(Z_k - h(X_{k|k-1})) \tag{2.16}$$

$$P_{k|k} = P_{k|k-1} - W_k[H_k P_{k|k-1} H_k^T + R_k] W_k^T \tag{2.17}$$

$$W_k = P_{k|k-1} H_k^T [H_k P_{k|k-1} H_k^T + R_k]^{-1} \tag{2.18}$$

Avec H_k la matrice Jacobienne de la fonction non linéaire h(.) definie comme suit :

$$H_k = \frac{\partial h}{\partial X}|_{X = X_{k|k-1}} \tag{2.19}$$

(+) L'extension du KF présente des avantages significatifs, notamment son adaptation aux systèmes non-linéaires, permettant ainsi une flexibilité accrue dans la modélisation des systèmes complexes et une capacité à traiter des non-linéarités de différents degrés.
(-) Son inconvénient réside dans l'augmentation de la complexité de calcul par rapport au KF, par l'ajout du calcul des matrices Jacobiennes.

— **Filtre de kalman sans parfum** :

Le filtre de Kalman sans parfum (Unscented Kalman Filter (UKF)) résout les problèmes d'approximation (de linéarisation) de l'EKF, comme évoqué par Wan et al., 2000. La distribution de l'état est à nouveau représentée par une variable aléatoire gaussienne, mais elle est maintenant spécifiée à l'aide d'un ensemble minimal de points d'échantillonnage soigneusement choisis. Ces points d'échantillonnage capturent complètement la moyenne et la covariance réelles de la variable aléatoire gaussienne et, lorsqu'ils sont propagés dans le véritable système non linéaire, ils capturent la moyenne et la covariance postérieures avec précision jusqu'au troisième ordre (expansion de la série de Taylor) pour toute non-linéarité.

Le principe de base de l'UKF est de créer un ensemble de points appelé "sigma points", qui représentent la distribution de probabilité des états à prédire. Ces points sont ensuite propagés à travers les fonctions de prédiction et de mesure pour calculer la moyenne et la covariance de la prédiction et de la mesure.

Contrairement au filtre de Kalman standard, l'UKF peut gérer des modèles de mouvement non linéaires, ce qui le rend plus précis et plus fiable dans les applications de localisation et de navigation.

L'UKF se base sur la transformation non parfumée qui est une méthode de calcul des statistiques d'une variable aléatoire qui subit une transformation non linéaire. Considérons la propagation d'une variable aléatoire x (de dimension L) à travers une fonction non linéaire, $y = g(x)$. Supposons que x ait une moyenne $\hat{x}$ et une covariance P_x. Pour calculer les statistiques de y, on forme une matrice χ_i avec des poids correspondants W_i comme suit :

$$\begin{cases} \chi_0 = \hat{x} \\ \chi_i = \hat{x} + \sqrt{(L+\lambda)P_x} \;\; i = 1, .., L \\ \chi_i = \hat{x} - \sqrt{(L+\lambda)P_x} \;\; i = L+1, .., 2L \\ W_0^{(m)} = \lambda/(L+\lambda) \\ W_0^{(c)} = \lambda/(L+\lambda) + (1 - \alpha^2 + \beta) \\ W_i^{(m)} = W_i^{(c)} = 1/2(L+\lambda) \;\; i = 1, .., 2L \end{cases} \tag{2.20}$$

où $\lambda = \alpha^2(L + k) - L$ est un paramètre d'échelle. α détermine la dispersion des points sigma χ_i autour de $\hat{x}$. k est un paramètre d'échelle secondaire qui est généralement fixé à 0, et β est utilisé

pour incorporer la connaissance préalable de la distribution de x. Ces vecteurs sigma se propagent selon la fonction non linéaire g : $\Upsilon_i = g(\chi_i) \quad i = 0, .., 2L$, et la moyenne et covariance de y sont approximées en utilisant une moyenne d'échantillon pondérée et une covariance des points sigma *a posteriori* :

$$\begin{cases} \hat{y} = \sum_{i=0}^{2L} W_i^{(m)} \Upsilon_i \\ P_y = \sum_{i=0}^{2L} W_i^{(c)} (\Upsilon_i - \hat{y})^T \end{cases} \tag{2.21}$$

Ce filtre peut être pris en forme informationnelle, et étendu à l'estimation multi-capteurs afin d'accroître la fiabilité de l'estimation.

(+) L'UKF présente plusieurs avantages par rapport au filtre de Kalman standard, notamment une meilleure prise en compte des non-linéarités du modèle de mouvement, une meilleure précision de l'estimation, ainsi qu'une plus grande robustesse aux erreurs de mesure.

(-) Cependant, l'UKF présente également des inconvénients, tels que sa complexité de mise en œuvre et de calcul, ainsi que sa sensibilité aux paramètres de la fonction de mesure. Malgré ces limitations, l'UKF reste une méthode de filtrage de données couramment utilisée dans les applications de localisation et de navigation, en particulier dans les cas où les modèles de mouvement sont non linéaires.

— **Filtre informationnel** :

Ce filtre reste toujours moins connu que le filtre de Kalman, bien qu'il admette plusieurs avantages. En effet, avec le filtre informationnel (IF), il est plus simple de représenter une incertitude globale en fixant la matrice informationnelle à 0, tandis qu'avec le filtre de Kalman, une telle incertitude est équivalente à une covariance de l'ordre de l'infini. De plus, le filtre informationnel tend à être calculatoirement plus stable que le filtre de Kalman et permet une mise en œuvre plus facile de la fusion de données multi-capteurs, surtout quand le nombre de capteurs augmente.

Le filtre informationnel utilise la forme informationnelle du vecteur d'état et de la matrice de covariances nommés respectivement vecteur informationnel (y) et matrice informationnelle (Y) comme évoqué par Grime et al., 1994. La compacité de la distribution de probabilités (ou la certitude sur les estimations d'état) peut être fournie par la matrice informationnelle.

La forme informationnelle du filtre de Kalman est la suivante :

$$\begin{cases} y_k = P_k^{-1} X_k \\ Y_k = P_k^{-1} \end{cases} \tag{2.22}$$

Ce filtre comme le KF est constitué de deux étapes :

(a) **Étape de prédiction** : peut être déduite à partir du KF et en

utilisant les équations (2.4) et (2.5) :

$$Y_{k|k-1} = [F_{k-1} Y^{-1}_{k|k-1} F^T_{k-1} + Q_{k-1}]^{-1} \tag{2.23}$$

$$y_{k|k-1} = Y_{k|k-1}[F_{k-1} Y^{-1}_{k|k-1} y_{k-1/k-1} + B_{k-1} u_{k-1}] \tag{2.24}$$

(b) **Étape de correction** : obtenu par l'usage de l'équation (2.7) :

$$I_{3x3} - W_k H_k = P_{k|k} P^{-1}_{k|k-1} \tag{2.25}$$

— H_k est la matrice d'observations
— K_k est le gain de Kalman $= P_{k|k} H^T_k R^{-1}_k$
A partir de l'équation (2.16), et en utilisant l'équation (2.25), on obtient :

$$X_{k|k} = [I_{3x3} - W_k H_k] X_{k|k-1} + W_k Z_k \tag{2.26}$$

$$P^{-1}_{k|k-1} X_{k|k-1} + H^T_k R^{-1}_k Z_k \tag{2.27}$$

Donc

$$y_{k|k} = y_{k-1|k} + H^T_k R^{-1}_k Z_k \tag{2.28}$$

(+) Ce filtre présente des avantages notamment dans sa flexibilité, la réduction de la complexité de calcul en évitant l'inversion de la matrice de covariance des innovations S_k de l'étape de correction du KF, ainsi que sa capacité à fusionner plusieurs corrections en une seule étape.
(-) Sa limitation, comme tout filtre de Kalman, est sa sensibilité aux erreurs de modélisation.

— **Filtre informationnel étendu** :

L'extension du filtre informationnel comme le filtre de Kalman a pour objectif principal de traiter la non-linéarité, et ce filtre Extended Informational Filter (EIF) se déduit de l'IF.

Rappelons de nouveau les équations du système non-linéaire :

$$X_{k+1} = f(X_k, u_k) + \omega_k \tag{2.29}$$

$$Z_k = h(X_k) + \nu_k \tag{2.30}$$

(a) **Étape de Prédiction** En se basant sur les équations de Jacobien (2.14) et (2.15), la matrice informationnelle et le vecteur informationnel peuvent s'écrire sous la forme suivante :

$$Y_{k|k-1} = [F_{k-1} Y^{-1}_{k-1/k-1} F^T_{k-1} + Q_{k-1}]^{-1} \tag{2.31}$$

$$y_{k|k-1} = Y_{k|k-1} f(X_{k-1/k-1}) \tag{2.32}$$

(b) **Étape de Correction**

$$Y_{k|k} = Y_{k|k-1} + H^T_k R^{-1}_k H_k \tag{2.33}$$

$$y_{k|k} = y_{k|k-1} + H^T_k R^{-1}_k [Z_k - h(X_{k|k-1} + H_k X_{k|k-1}] \tag{2.34}$$

— **Fusion de données multi-capteurs** :

L'approche de base de fusion de données considérée est fondée sur le fait de fusionner les contributions de plusieurs capteurs pour aboutir à une solution de localisation plus robuste, étant donné l'usage de technologies variées pour percevoir l'environnement D. Lee, 2008. Considérons N capteurs dont chacun possède le modèle d'observation suivant :

$$z_K^i = H_k^i X_k + \nu_k^i \tag{2.35}$$

Dans le cas du filtre de Kalman standard, l'estimation peut être modélisée comme étant une simple combinaison linéaire des innovations et de la prédiction. Par contre, l'état est :

$$X_{k|k} \neq X_{k|k-1} + \sum_{i=1}^{N} W_k^i (z_k^i - H_k^i X_{k|k-1}) \tag{2.36}$$

La raison pour laquelle l'équation (2.36) ne peut se limiter à une égalité est que, à un instant donné, les innovations des différents capteurs partagent entre elles la prédiction. Ce qui conduit à une corrélation des innovations apparaissant au niveau des éléments "hors diagonal" de la matrice S_k. Donc, pour N capteurs, le vecteur d'observations, la matrice d'observations et le vecteur bruit lié aux capteurs sont obtenus comme suit :

$$z_k = \begin{pmatrix} z_k^1 \\ \vdots \\ z_k^N \end{pmatrix} \qquad H_k = \begin{pmatrix} H_k^1 \\ \vdots \\ H_k^N \end{pmatrix} \qquad \nu_k = \begin{pmatrix} \nu_k^1 \\ \vdots \\ \nu_k^N \end{pmatrix} \tag{2.37}$$

$$R_k = E(\nu_k \nu_k^T) = \begin{pmatrix} R_k^1 & 0 & \dots & 0 \\ 0 & R_k^2 \dots & 0 & \\ \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & R_k^N \end{pmatrix} \tag{2.38}$$

Les contributions informationnelles I_k^l et i_k^l sur $Y_{k|k}$ et $y_{k|k}$ peuvent donc être développées comme suit :

$$H_k^T R_K^{-1} H_k = \begin{pmatrix} (H_k^1)^T \dots (H_k^1)^T \end{pmatrix} \begin{pmatrix} (R_k^1)^{-1} & 0 & \dots & 0 \\ 0 & (R_k^2)^{-1} \dots & 0 & \\ \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & (R_k^N)^{-1} \end{pmatrix} \begin{pmatrix} H_k^1 \\ H_k^2 \\ \vdots \\ H_k^N \end{pmatrix} \tag{2.39}$$

$$= \sum_{i=1}^{N} (H_k^i)^T (R_k^i)^{-1} H_k^i \tag{2.40}$$

De même :

$$H_k^T R_K^{-1} z_k = \begin{pmatrix} (H_k^1)^T \dots (H_k^1)^T \end{pmatrix} \begin{pmatrix} (R_k^1)^{-1} & 0 & \dots & 0 \\ 0 & (R_k^2)^{-1} \dots & 0 & \\ \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & (R_k^N)^{-1} \end{pmatrix} \begin{pmatrix} z_k^1 \\ z_k^2 \\ \vdots \\ z_k^N \end{pmatrix} \tag{2.41}$$

$$= \sum_{i=1}^{N} (H_k^i)^T (R_k^i)^{-1} z_k^i \tag{2.42}$$

Par conséquent, l'ajout d'un capteur supplémentaire nécessite le calcul des termes $(H_k^i)^T (R_k^i)^{-1} H_k^i$ et $(H_k^i)^T (R_k^i)^{-1} z_k^i$.

Le filtre informationnel modélise l'étape de mise à jour comme étant une somme des informations provenant des observations.

Le filtre informationnel est plus rapide si la dimension du vecteur d'observations est supérieur à $1.65n$ pour un système variable dans le temps et s'il est supérieur à $0.75n$ pour un système invariable dans le temps, avec n la dimension du vecteur d'état.

— **Filtre d'intersection de covariances** :

Le filtre d'intersection de covariances CIF est une technique de fusion de données couramment utilisée dans les systèmes de localisation et de navigation. Il s'agit d'un filtre à plusieurs modèles qui utilise plusieurs filtres de Kalman étendus EKF.

Le CIF utilise une méthode d'intersection de covariances pour fusionner les sorties de chaque EKF en une seule estimation. Cette méthode consiste à calculer l'intersection des ellipses d'incertitude de chaque EKF pour obtenir une ellipse d'incertitude globale. L'estimation de position finale est obtenue en pondérant les sorties de chaque EKF en fonction de leur contribution respective à l'intersection des ellipses.

Pour 2 corrections $\{X_a, P_{aa}\}$ et $\{X_b, P_{bb}\}$, la correction $\{X_c, P_{cc}\}$ obtenu en appliquant le filtre CIF est représenté par les équations suivantes selon JULIER, S. et al., 2009 :

$$P_{cc}^{-1} = \omega \times P_{aa}^{-1} + (1 - \omega) \times P_{bb}^{-1} \tag{2.43}$$

Et concernant le vecteur d'état :

$$P_{cc}^{-1} \times X_c = \omega \times P_{aa}^{-1} \times X_a + (1 - \omega) \times P_{bb}^{-1} \times X_c \tag{2.44}$$

(+) Le CIF présente plusieurs avantages par rapport aux filtres de Kalman classiques. Tout d'abord, il est plus robuste face aux perturbations et aux incertitudes dans les données de capteurs. En outre, il peut gérer des situations où il y a plusieurs hypothèses de mouvement possibles. Enfin, il est capable de fournir une estimation de l'état plus précise et plus fiable en utilisant les informations de plusieurs sources.

(-) Cependant, le CIF présente également certains inconvénients, tels que sa complexité de mise en œuvre et de calcul.

Malgré ces limitations, le CIF reste une technique de fusion de données couramment utilisée dans les applications de localisation et de navigation, en particulier dans les cas où les données de capteurs sont bruitées ou incomplètes.

— **Filtre d'intersection de covariance par lots** :

Le CIF peut être généralisé vers un nombre arbitraire n > 2 de mises à jour. Ce filtre est appelé le filtre d'intersection de covariances par lot ou la *batch covariance intersection filter* B-CIF. Il permet d'obtenir une correction $\{X_c, P_{cc}\}$ depuis un lot de corrections pondérés $\{X_{a_i}, P_{a_i a_i}\}$. Il s'applique comme suit :

$$P_{cc}^{-1} = \omega_1 P_{a_1 a_1}^{-1} + \cdots + \omega_n P_{a_n a_n}^{-1} \tag{2.45}$$

$$P_{cc}^{-1} \times X_c = \omega_1 P_{a_1 a_1}^{-1} \times X_{a_1} + \cdots + \omega_n P_{a_n a_n}^{-1} \times X_{a_n} \tag{2.46}$$

où $\sum_{i=1}^{n} \omega_i = 1$. Les poids sont optimisés au cours des deux premières itérations de la connexion, afin d'éviter que l'optimisation du poids ne compense le défaut s'il se produit. Ceci est réalisé en résolvant l'équation suivante :

$$min_{\omega_i} trace(P_{a_i a_i}) \tag{2.47}$$

— **Filtre d'intersection à covariance divisée** :

Le filtre de covariance à intersection divisée (S-CIF) est une alternative entre EKF et CIF. Selon Pierre et al., 2018 et Julier, S. et al., 2009 dans S-CIF, la matrice de covariance $P_{k|k}$ est divisée en deux parties, partie indépendante $P_{i,k|k}$ correspondant à la covariance de l'état sans corrélation d'autres informations et dépendante $P_{d,k|k}$ correspondant à celle avec ces corrélations, tel que :

$$P_{i,k|k} = P_{i,k|k} + P_{d,k|k} \tag{2.48}$$

Cette représentation influence toute la modélisation du système. Les équations de l'EKF sont modifiées telles que pour l'équation d'évolution, la matrice d'erreur de modèle est elle aussi subdivisée en partie dépendante et indépendante.

(a) **Étape de Prédiction**

Pour une entrée $(u_k, Q_k, Q_{i,k})$, l'évolution du système est définie par :

$$X_{k|k-1} = f(X_{k-1/k}, u_{k-1}) \tag{2.49}$$

$$\begin{cases} P_{k|k-1} = F_{k-1} P_{k|k-1} F_{k-1}^T + Q_{k-1} \\ P_{i,k|k-1} = F_{k-1} P_{i,k|k-1} F_{k-1}^T + Q_{i,k-1} \\ P_{d,k|k-1} = P_{k|k-1} - P_{i,k|k-1} \end{cases} \tag{2.50}$$

(b) **Étape de Correction**

Pour une fonction d'observation h et une mesure définie par le triplet $(z_k, R_{i,k}, R_{d,k})$, la correction est calculée comme suit :

$$\begin{cases} P_1 = \omega^{-1} P_{d,k|k-1} + P_{i,k|k-1} \\ P_2 = (1-\omega)^{-1} R_{d,k} + R_{i,k} \end{cases} \tag{2.51}$$

$$\begin{cases} W_k = P_1 H_k^T (H_k P_1 H_k^T + V_k P_2 V_k^T)^{-1} \\ P_{k|k} = (I - W_k H_k) P1 \\ P_{i,k|k} = (I - W_k H_k) P_{i,k|k-1} (I - W_k H_k)^T + (I - W_k V_k) R_{i,k} (I - W_k V_k)^T \\ P_{d,k|k} = P_{k|k} - P_{i,k|k} \end{cases} \tag{2.52}$$

où V_k est le Jacobien de la l'observation par rapport au vecteur d'entrée ν_k, $V_k = \frac{\partial h}{\partial u}(X_{k|k-1}, 0)$

Le vecteur d'état évolue de la même manière que celle présentée pour KF.

(+) Ce filtre est conçu pour gérer efficacement les systèmes non linéaires. Il peut traiter des systèmes avec des modèles non linéaires sans nécessiter une linéarisation préalable, réduit la perte d'information et retire les dépendances des covariances.

(-) Le S-CIF présente un inconvénient majeur lié à la nécessité d'une initialisation précise des matrices de covariances, ainsi qu'à la dépendance à une connaissance préalable adéquate du système et de ses paramètres.

3. Filtre particulaire :

Les probabilistes ont commencé à développer une théorie générale du filtrage non linéaire en temps continu au milieu des années soixante. En statistique, les modèles d'espace d'état et les techniques de filtrage ont mis plus de temps à s'implanter. Dans les années 70 et 80, la relation entre l'espace d'état linéaire et les modèles ARMA a été étudiée et utilisée.

L'article de N. Gordon et al., 1993 qui a développé des méthodes de Monte Carlo récursives connues sous le nom de filtres à particules, a constitué une percée. Il est intéressant de noter que Handschin et al., 1969 avaient proposé l'utilisation de méthodes de Monte Carlo bien plus tôt, mais l'idée de rééchantillonnage était absente.

Le filtre particulaire est connu sous le nom de méthode de Monte-Carlo séquentielle. Il conduit à une approximation du filtre bayésien en utilisant des méthodes numériques. L'hypothèse gaussienne n'est pas requise contrairement au filtre de Kalman étendu. En outre, aucune hypothèse n'est prise ni sur la linéarité ni sur le type du bruit. Ce filtre traite le système comme un ensemble de variables discrètes ou modes (normal, défaut1, défaut2...), et de variables continues observables ou non (vitesses, accélérations, orientations, ...). Donc, théoriquement, il peut être appliqué sur n'importe quel système.

La distribution de probabilités est représentée par un ensemble de particules $(X_k^1, .., X_k^N)$ caractérisées par les poids $(\omega_k^1, .., \omega_k^N)$. Pour l'initialisation on prend : $\omega^{1:N} = \frac{1}{N}$ et $X^{1:N} \sim p(X_0)$.

(a) **Étape de Prédiction**

$$X_k^i \sim p(X_k^i | X_{k-1}^i) \tag{2.53}$$

(b) **Étape de Correction**

$$\tilde{\omega}_k^i = \tilde{\omega}_{k-1}^i p(Z_k|X_k^i) \tag{2.54}$$

La normalisation des poids se fait par (2.55) :

$$\omega_k^i = \frac{\tilde{\omega}_k^i}{\sum_{i=1}^{N} \tilde{\omega}_k^i} \tag{2.55}$$

(-) Cette relation présente un risque de dégénérescence lorsque le poids des particules tend vers zéro si k tend vers l'infini. Il faut donc passer par un ré-échantillonnage du jeu des particules en dupliquant celles de plus grand poids et en éliminant celles de faible poids, correspondant à un filtre particulaire avec interaction SIR *(Sampling Importance Resampling)*.

Lorsqu'il s'agit de localiser un véhicule, il existe différentes approches pour estimer sa position et son orientation en fonction de l'application et des capteurs embarqués. Cependant, l'environnement dans lequel évolue le véhicule est souvent dynamique et non linéaire, ce qui rend parfois difficile une modélisation exhaustive de son comportement. Les capteurs observant cet environnement sont, quant à eux, sujets à des bruits et à des incertitudes. Dans cette perspective, une modélisation adéquate des mesures repose sur une approche stochastique non déterministe, considérant ces mesures comme des distributions de probabilités ayant une incertitude. Dans ce contexte, la comparaison de ces distributions nécessite un outil capable de prendre en compte leurs caractéristiques. C'est là que les divergences interviennent, elles servent à quantifier la dissimilarité entre deux distributions. Dans la suite de cet exposé, nous examinerons en détail les outils de la théorie de l'information, notamment les notions de divergences.

2.4.3.3 Notion de divergence

Afin de vérifier la cohérence entre la prédiction et la correction considérés comme des distributions de probabilités, nous utilisons la notion de divergence. L'objectif est de détecter une éventuelle incohérence entre ces deux estimations qui peut impliquer la présence d'un défaut.

L'entropie de Shannon constitue le concept le plus fondamental de la théorie de l'information. C'est une mesure de l'incertitude associée à l'information provenant d'une source.

$$H(X) = -\sum p(x) \log(p(x)) \tag{2.56}$$

— (x) représente la probabilité associée à la variable aléatoire X
— $H(X)$ est l'entropie de Shannon
— $log(.)$ est comme le logarithme népérien

Dans notre cas, nous considérons que les mesures de capteurs sont considérées comme étant une distribution de probabilité. La cause de cette hypothèse est la nature incertaine des données fournies par les capteurs, qui ne sont pas finies et assimilées au point de mesure, mais considérées comme une variable aléatoire qui est assimilée dans notre cas à une loi gaussienne. En conséquence, ce point de mesure constitue la moyenne de la distribution de probabilité, avec une variance définie selon l'incertitude du capteur. Cette incertitude est lié à la technologie du capteur et à son comportement dans l'environnement. Pour clarifier l'idée, prenons l'exemple d'un magnétomètre. Ce capteur possède selon sa fiche technique une certaine incertitude lié à la qualité de ses composants et à la technologie. De plus, avec le changement de la magnétisation de l'environnement, et sa température, le comportement du capteur et par suite la valeur qu'il fournit diffère.

Plusieurs divergences ont été dérivées de l'entropie de Shannon. Dans la suite, nous présentons celle de Kullback-Leibler et celle de Jensen-Shannon.

2.4.3.4 Divergence de Kullback-Leibler

La divergence de Kullback-Leibler (ou entropie relative) est une mesure de dissimilarité entre deux probabilités $p_1(x)$ et $p_2(x)$ considérés comme étant celle *à priori* et *à posteriori* respectivement.

Cette mesure initialement fondée sur l'entropie de Shannon est une mesure de l'incertitude associée à l'information provenant d'une source, qui constitue le concept le plus fondamental de la théorie de l'information.

Elle est définie comme suit :

$$KL(p||q) = \int_x p(x) \log\left(\frac{p(x)}{q(x)}\right) \quad continue \tag{2.57}$$

$$KL(p||q) = \sum_x p(x) \log\left(\frac{p(x)}{q(x)}\right) \quad discret \tag{2.58}$$

Cette métrique informationnelle peut être considérée comme étant l'espérance d'un rapport de vraisemblance logarithmique (*Logarithmic Likelihood Ratio* (LLR)) :

$$KL(p||q) = E_p\left(\log\left(\frac{p(x)}{q(x)}\right)\right) \tag{2.59}$$

La divergence de Kullback-Leibler entre deux distributions gaussiennes $f(x)$ et $g(x)$, de dimension M, de moyennes μ_1 et μ_2, et de matrices de covariances P_1 et P_2 respectivement peut être définie dans l'équation Eq(2.60) :

$$KL(f(x)||g(x)) = \frac{1}{2}\left[trace(P_2^{-1}P_1) + \log|\frac{P_2}{P_1}| - M + (\mu_1 - \mu_2)^T P_2^{-1}(\mu_1 - \mu_2)\right] \tag{2.60}$$

Avec :

— $(\mu_1 - \mu_2)^T P_2^{-1}(\mu_1 - \mu_2)$ est la distance de Mahalanobis qui mesure la distance entre les moyennes des deux distributions, représentant le signal.

— $trace(P_2^{-1}P_1)$ est l'orientation d'une distribution par rapport à l'autre.
— $\log|\frac{P_2}{P_1}| - M$ est la compacité des distributions de données, représentant la dispersion.

Lorsque $\mu_1 = \mu_2$, cette divergence est identique à une divergence de Bregman matricielle.

En dressant deux distributions de probabilités gaussienne dans la Fig.2.16, nous pouvons voir ces différentes parties de l'équation (2.60) de Kullback-Leibler. La distance de Mahalanobis entre la distribution 1 et la distribution 2 au cas où ces deux distributions sont concentrique ne suffit pas pour mesurer la dissimilarité entre ces distributions (voir les cas dans la Fig.2.17), puisque ce terme ne peut pas seul décrire la compacité et l'orientation des deux distributions. Cette interprétation pour le cas de capteurs est :

— Cas de différentes orientations (Fig.2.17a) : La différence dans l'orientation implique que les capteurs sont relativement indépendants dans la capture des variations de la valeur mesuré le long des différents axes, ayant des corrélations différentes entre leurs deux variables aléatoires.
— Cas de différentes compacités (Fig.2.17b) : La différence dans la variance indique un différent niveau de confiance dans la mesure obtenu : plus la variance est petite, plus la confiance dans la mesure obtenue augmente.

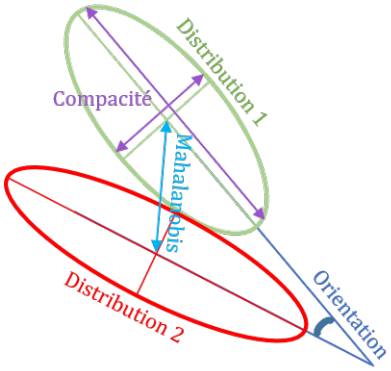


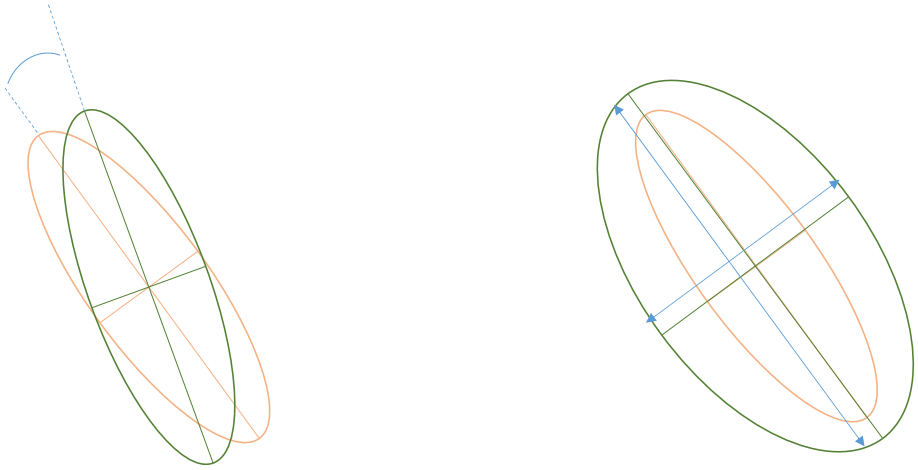
Figure 2.16 – Divergence de kullback-leibler entre deux distributions

Dans la littérature, sous certaines hypothèses (taille d'échantillon importante, matrice de covariances diagonale, indépendance des composantes de la distribution de dimension M...), la divergence de Kullback-leibler est associée à une loi de χ^2 sous la forme de l'équation (2.61) :

$$\chi^2(p,q) = \sum_x \frac{(p(x) - q(x))^2}{q(x)} \tag{2.61}$$

La loi de distribution de la divergence de Kullback-leibler, dans le cas "sans défaut", est reliée à celle de Fisher et du χ^2, et elle peut être obtenue dans Eq(2.62) :

$$KL(p||q) \sim \frac{1}{2}\left[\frac{M(n-1)}{(n-M)n}F_{M,n-M} + \frac{1}{n-1} \times \frac{1}{1 - \frac{1}{6(n-1)-1}(2M+1-\frac{2}{M+1}}\chi^2_{\frac{1}{2}(M(M+1))}\right] \tag{2.62}$$



(a) Cas concentrique avec différentes orientations

(b) Cas concentriques avec différentes compacités

Figure 2.17 – Cas de dissimilarités entre deux distributions divergence de kullback-leibler entre deux distributions

Il faut noter que lorsque la taille de l'échantillon passe à l'infini, la distribution de Fisher se rapproche d'une loi du χ^2 : $F_{M,n-M} \sim \frac{\chi^2_M}{M}$.

Étude de consistance : Dans notre cas, M qui est la dimension du vecteur d'état est 3, et n qui est le nombre d'échantillons est plus grand que 100. Donc dans notre cas, la divergence de Kullback-Leibler suit une loi de $\frac{\chi^2_3}{3}$. Les limites d'une telle distribution de probabilité sont données par un tableau. Dans notre cas, les marges sont présentées dans le Tableau 2.1.

k	0.995	0.990	0.975	0.950	0.900	0.500	0.100	0.050	0.025	0.010	0.005
3	0.07	0.11	0.22	0.35	0.58	2.37	6.25	7.81	9.35	11.34	12.84

Table 2.1 – Loi du χ^2 avec k =3 degrés de liberté

2.4.3.5 Divergence de Jensen-Shannon

En théorie des probabilités et en statistique, la divergence de Jensen-Shannon est également une méthode de mesure de la similarité entre deux distributions de probabilité. Cette divergence de Jensen-Shannon est symétrique, contrairement à la divergence de kullback-leibler qui est asymétrique. Cela veut dire que la valeur de cette divergence ne varie pas en fonction de l'ordre dans lequel les distributions sont implémentés.

Il est complexe de déterminer de manière universelle la supériorité d'une divergence par rapport à l'autre, car leur sélection dépend étroitement du contexte d'application particulier. Lorsque nous cherchons à mesurer la dissimilitude entre deux distributions de probabilité sans avoir une distinction claire entre l'une servant de "référence" et l'autre de "rapprochement", l'emploi d'une divergence symétrique est pertinent. En revanche, si cette nuance

est connue, alors l'utilisation d'une divergence asymétrique est appropriée.
Elle est définie comme suit :

$$D_{JS}(p(x)||q(x)) = H\left(\frac{1}{2}p(x) + \frac{1}{2}q(x)\right) - \frac{1}{2}H\left(p(x)\right) - \frac{1}{2}H\left(q(x)\right) \tag{2.63}$$

et sa relation avec la divergence de Kullback Leibler est la suivante :

$$\begin{aligned} D_{JS}(p(x)||q(x)) &= H\left(\frac{1}{2}p(x) + \frac{1}{2}q(x)\right) - \frac{1}{2}H(p(x)) - \frac{1}{2}H(q(x)) \\ = -\sum\left(\frac{1}{2}(p(x)+q(x))log(\frac{1}{2}(p(x)+q(x)))\right) &+ \frac{1}{2}\sum p(x)log(p(x)) + \frac{1}{2}\sum q(x)log(q(x)) \\ &= \sum\left(\frac{1}{2}p(x)log\left(\frac{p(x)}{\frac{1}{2}(p(x)+q(x))}\right)\right) + \sum\left(\frac{1}{2}q(x)log\left(\frac{q(x)}{\frac{1}{2}(p(x)+q(x))}\right)\right) \\ &= \frac{1}{2}D_{KL}\left(p(x)||\frac{1}{2}(p(x)+q(x))\right) + \frac{1}{2}D_{KL}\left(q(x)||\frac{1}{2}(p(x)+q(x))\right) \end{aligned} \tag{2.64}$$

Dans notre étude, ces outils sont utilisés pour comparer la prédiction à la correction afin de vérifier leur cohérence et détecter en cas de leur incohérence la présence d'un fonctionnement anormal. Ces divergences par la suite jouent le rôle d'indicateurs de défaillance dans le système : lorsqu'une correction d'un capteur est dissimilaire à sa prédiction, il se peut que ce capteur est défaillant. Afin de confirmer l'hypothèse de défaut, l'évaluation de la quantité de dissimilarité est nécessaire. D'où, cette divergence est considéré dans notre cas comme résidu pour la détection de défauts.

2.4.3.6 Évaluation des résidus et seuillage

Dans la section précédente, nous avons présenté la notion de divergence. C'est une mesure de dissimilarité entre deux distributions de probabilités. Dans notre cas, elle est utilisée pour générer des résidus comparant la prédiction et la correction du capteur afin vérifier sa cohérence. La décision de la présence de défaut dépend de la valeur obtenue de ce résidu. Elle est prise sur la base de deux hypothèses : H_0 représente l'hypothèse nulle (absence de défaut) et H_1 est l'hypothèse alternative (présence d'un défaut). On définit :

— **La probabilité de détection** : c'est la probabilité de choisir H_1 sachant que H_1 est vraie :

$$P_D = P(u_1/H_1) = \int_{th}^{\infty} p(x/H_1)dx \tag{2.65}$$

— **La probabilité de fausse alarme** : c'est la probabilité de choisir H_1 sachant que H_0 est vraie :

$$P_F = p(u_1/H_0) = \int_{th}^{\infty} p(x/H_0)dx \tag{2.66}$$

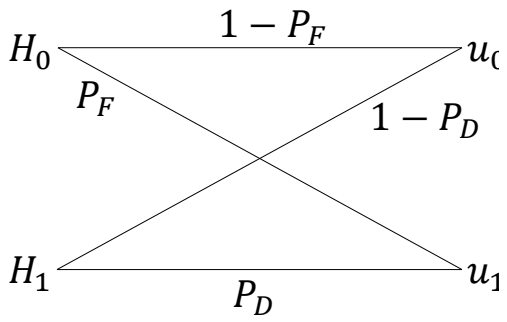


Figure 2.18 – Relation entre la décision et l'hypothèse

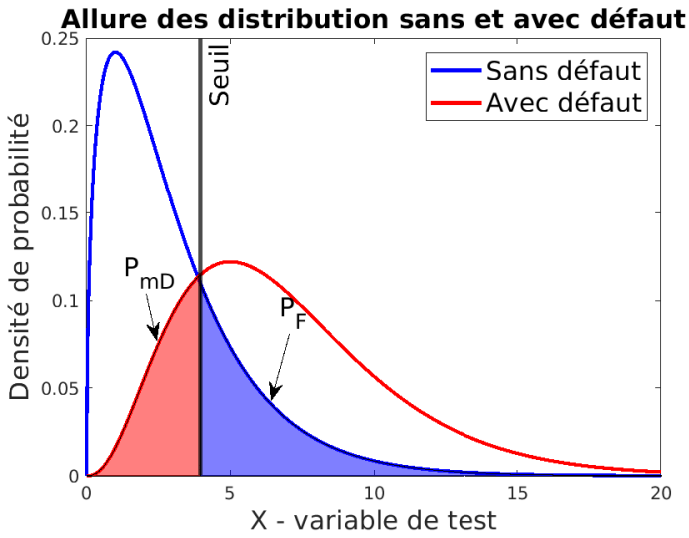


Figure 2.19 – Modélisation des probabilités de fausse alarme et de détection manquée

— **La probabilité de détection manquée** : c'est la probabilité de choisir H_0 sachant que H_1 est vraie :

$$P_{mD} = p(u_0/H_1) \tag{2.67}$$

où $p(x = H_0)$ et $p(x = H_1)$ sont les densités de probabilités de la variable x suivant respectivement les hypothèses H_0 et H_1, et th est le seuil de détection, ces probabilités sont visualisées dans Fig(2.19).

Divers critères pour le calcul des seuils sont disponibles dans la littérature. Les paragraphes suivants fournissent un résumé de ces critères.

— **Critère de Bayes :**

Étant donné l'équation (2.68) qui classifie l'observation x selon les hypothèses H_0 et H_1 :

$$p(H_1/x) \underset{<}{\geq} p(H_0/x) \tag{2.68}$$

L'équation (2.68) peut être interprétée comme suit :

— Si $p(H_1 = x) > p(H_0 = x)$, alors H_1 est choisie.
— Si $p(H_1 = x) < p(H_0 = x)$, alors H_0 est choisie.
En utilisant la règle de Bayes, on obtient :

$$\frac{p(x/H_1)p(H_1)}{p(x)} \underset{<}{\overset{H}{>}} \frac{p(x/H_0)p(H_0)}{p(x)} \tag{2.69}$$

Afin de choisir une hypothèse et en utilisant la règle de Bayes, on obtient un test similaire à celui du rapport de vraisemblance présenté par l'équation (2.69). Le critère de Bayes attribue un coût C_{ij} à chacune des situations : « on décide H_i alors que H_j est correct ». Le risque de Bayes, qui représente le coût moyen, peut être donné par la formule suivante :

$$R_{bayes} = \sum_{i=0}^{1}\sum_{j=0}^{1} C_{ij}p(x \in R_i/H_j)p(H_j) = \sum_{i=0}^{1}\sum_{j=0}^{1} C_{ij}p(H_j)\int_{R_i} p(x/H_j)dx$$
$$= [C_{11}p_1 + C_{10}p_1] + \int_{R_0}[p_1C_{01} - C_{11})p(x/H_1) - p_0(C_{10} - C_{00}p(x/H_0))]dx \tag{2.70}$$

R_i représente le domaine de choix de l'hypothèse H_i. En général, le coût attribué à une décision correcte doit être plus faible que celui attribué à une décision erronée. Donc $C_{10} > C_{00}$ et $C_{01} > C_{11}$. En ignorant le premier membre constant de l'équation (2.70), on constate que minimiser le risque de Bayes consiste à chercher une région de décision R_0 qui minimise l'intégrale. Ce qui conduit au test final introduisant le seuil de détection selon Jutten, 2007 :

$$\Lambda = \frac{p(x/h_1)}{p(x/h_0)} \underset{<}{\overset{H}{>}} \frac{P_0}{1 - P_0}\frac{C_{10} - C_{00}}{C_{01} - C_{11}} \tag{2.71}$$

— **Courbe ROC :**
La courbe Receiver Operating Characteristic (ROC) est un graphique qui trace les taux de vrais positifs (probabilité de détection) en fonction des faux positifs (probabilité de fausse alarme). Cette courbe permet la comparaison de tests de diagnostic effectués sur un même jeu de données. Elle permet également de fixer un seuil de détection optimisé.

Différentes valeurs du seuil ont été introduits, toutes liées à la courbe ROC :
— L'indice de Youden connu sous le nom « Youden's Index ». C'est le point de la courbe ROC le plus loin verticalement de la ligne de chance (la diagonale) :

$$J = max(P_D - P_F) \tag{2.72}$$

— $(P_F = 0 ; P_D = 1)$ représente le point optimal qui est difficile à atteindre. Le critère (0,1) consiste à chercher le point de la courbe ROC à distance minimale du point (0,1) :

$$min(\sqrt{(1 - P_D)^2 + P_F^2}) \tag{2.73}$$

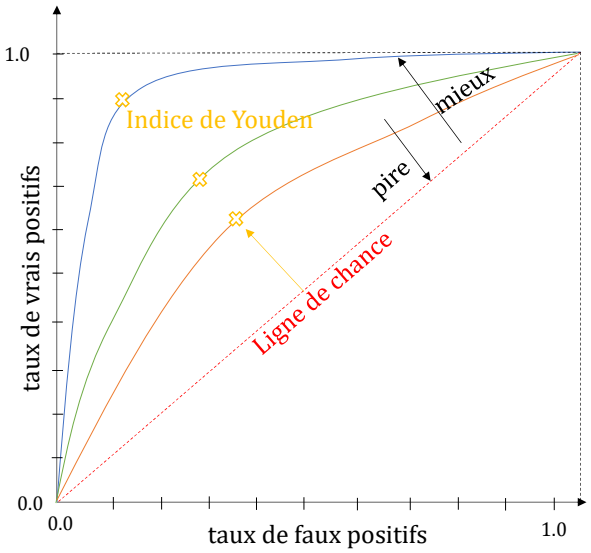


Figure 2.20 – Visualisation de la courbe ROC

— **Information Mutuelle** :

L'objectif est d'obtenir les règles de décision pour minimiser le coût moyen, ce qui est équivalent au problème de maximisation de l'information mutuelle :

$$I(H,u) = h(H) - h(H/u) = \sum_{H}\sum_{u} \log \frac{p(u/H)}{p(u)} \tag{2.74}$$

tel que h(.) est l'entropie de Shannon présentée dans l'équation (2.56). En développant cette équation selon les valeurs correspondantes, on obtient :

$$\begin{aligned} I(H,u) = P_0(1-P_F)\log \frac{1-P_F}{P_0(1-P_F)+(1-P_0)(1-P_D)} \\ +P_0P_F \log \frac{P_F}{P_0P_F+(1-P_0)P_D} \\ +(1-P_0)(1-P_D)\log \frac{1-P_D}{P_0(1-P_F)+(1-P_0)(1-P_D)} \\ +(1-P_0)(1-P_D)\log \frac{P_D}{P_0P_F+(1-P_0)P_D} \end{aligned} \tag{2.75}$$

Selon la valeur de P_0, le maximum de l'information mutuelle est obtenu en fixant à la fois P_{mD} et P_F soit 0 ou 1. Prenons la dérivée partielle de P_D par rapport à P_F qui correspond au seuil. En réarrangeant l'information mutuelle et la dérivant partiellement par rapport à P_F, on obtient :

$$Seuil = \frac{-P_0[\log(\alpha_0/\alpha_1) - log((1-P_F)/P_F)]}{(1-P_0)[\log(\alpha_0/\alpha_1) - \log((1-P_D)/P_D)]} \tag{2.76}$$

— **Critères basés sur les divergences :**

En utilisant une analogie similaire à celle de l'information mutuelle, on cherche à construire des critères de choix de seuil à partir des mesures de divergences. À partir du théorème de Bayes, le problème d'optimisation du seuil utilise le gain informationnel entre la distribution à priori (p(H)) et la distribution à posteriori (P(H/u)) sachant qu'une décision a été prise et considérée.

La somme des gains informationnels (*Kullback-Leibler Summation* (KLS)) associés aux décisions u_0 et u_1 est donnée par Eq(2.77) :

$$KLS = D_{KL}(p(H/u_0)||p(H)) + D_{KL}(p(H/u_1)||p(H)) \tag{2.77}$$

Cette mesure peut s'écrire sous la forme d'une double somme en considérant les deux hypothèses :

$$KLS = \sum_{i=0,1}\sum_{j=0,1} p(H_i/u_j)\frac{p(H_i/u_j)}{p(H_i)} \tag{2.78}$$

En utilisant une analogie similaire à celle de l'information mutuelle, on cherche à construire des critères de choix du seuil à partir des mesures de divergence.

$$\begin{aligned}\frac{\partial KLS}{\partial \nu} = \sum_{i=0,1}\Bigg\{ &\frac{\partial \alpha_i}{\partial \nu}\left(\frac{\beta_i}{(\alpha_i+\beta_i)^2}\log\left(\frac{1-P_0}{P_0}\frac{\alpha_i}{\beta_i}\right)\right) \\ &+\frac{\partial \beta_i}{\partial \nu}\left(\frac{\alpha_i}{(\alpha_i+\beta_i)^2}\log\left(\frac{P_0}{1-P_0}\frac{\beta_i}{\alpha_i}\right)\right)\Bigg\}\end{aligned} \tag{2.79}$$

avec :

$$\begin{aligned}\alpha_0 &= P_0(1-P_F) & \alpha_1 &= P_0 P_F \\ \beta_0 &= (1-P_0)(1-P_D) & \beta_1 &= (1-P_0)P_D\end{aligned}$$

De la même manière que le critère de Kullback-Leibler, on peut calculer celui de Jensen-Shannon :

$$JSS = JS(p(H/u_0)||p(H)) + JS(p(H/u_1)||p(H)) \tag{2.80}$$

$$JSS = \frac{1}{2}\sum_{i=0,1}\sum_{j=0,1}\left(D_{KL}(p(H_i/u_j)||M_{ij}) + D_{KL}(p(H_i)||M_{ij})\right) \tag{2.81}$$

avec

$$M_{ij} = \frac{1}{2}\left(p(H_i/u_j) + p(H_i)\right) \tag{2.82}$$

En développant et replaçant les membres dans l'équation (2.81), on obtient :

$$\begin{aligned}\frac{\partial JSS}{\partial \nu} = \frac{1}{2}\sum_{i=0,1}\frac{\frac{\partial \alpha_i}{\partial \nu}\beta_i - \frac{\partial \beta_i}{\partial \nu}\alpha_i}{\alpha_i+\beta_i}\Bigg\{ &\left(\frac{1}{(\alpha_i+\beta_i)}\log\left(\frac{\alpha_i(\beta_i+P_1(\alpha_i+\beta_i))}{\beta_i(\alpha_i+P_0(\alpha_i+\beta_i))}\right)\right) \\ &-\left(\frac{\alpha_i+\beta_i-P_0}{\alpha_i+P_0(\alpha_i+\beta_i)}\right) - \left(\frac{\alpha_i+\beta_i-P_1}{\beta_i+P_1(\alpha_i+\beta_i)}\right)\Bigg\}\end{aligned} \tag{2.83}$$

2.4.3.7 Détection et isolation de défauts

La vérification de la cohérence de la correction apportée par la mesure d'un capteur avec la prédiction est réalisée par l'usage de divergences. La décision de la présence d'un défaut est basée sur la valeur de cette divergence, et par sa comparaison à un seuil.

Une première étape est d'appliquer une fusion de données de tous les capteurs et de la comparer à la prédiction. Cette comparaison est faite avec un résidu de détection Rd_{JS} qui quantifie la dissimilarité entre les distributions de probabilités de la correction obtenue par la fusion de données multi-capteurs à la prédiction. Puis, ce résidu de détection est évalué grâce à un seuil calculé par l'une des méthodes présentées dans la section précédente. Cette étape constitue l'étape de détection de défaut, où la décision de la présence d'un défaut est prise, qui par la suite déclenche l'étape d'isolation de ce défaut détecté. L'étape d'isolation de défaut sert à localiser le capteur qui a causé le défaut détecté.

Cette localisation du capteur défectueux se fait grâce à une table de signature. Sur cette table, les lignes représentent les capteurs du système, et les colonnes les résidus générés. Cette table est booléenne, c'est à dire qu'un élément de cette table possède la valeur nulle si le résidu a une valeur inférieure au seuil, et une valeur unitaire s'il le dépasse. Un défaut sur un capteur est évalué en comparant le comportement des résidus à la ligne de la table de signature qui correspond à ce capteur. Cette ligne constitue le vecteur de signature du capteur, dont la forme diffère selon le schéma d'observateur utilisé. Dans le cas du schéma d'observateur dédié DOS, seul un capteur est utilisé pour calculer la correction. Ceci génère un vecteur de signature nul sauf pour le résidu correspondant à la divergence entre la prédiction et la correction qui intègre le capteur dont ce vecteur est dédié. Le cas inverse est observé dans le cas du schéma d'observateur généralisé GOS. Dans un tel schéma, tous les capteurs sont utilisés pour calculer la correction sauf un à la fois. Ceci génère un vecteur de signature complémentaire à celui du cas de DOS. En effet, ce vecteur n'est nul que lorsque le capteur à l'origine de défaut n'est pas utilisé dans le résidu.

Le nombre d'indicateurs et l'information qu'ils reflètent doit être synthétisée de manière à ce que tout capteur du système soit observable et diagnosticable, et que le système est bien ou sur-observé. Cela signifie qu'il existe au moins un résidu par capteur. En effet, dès lors qu'un défaut est détecté, une procédure d'isolation doit être intégrée afin de déterminer l'origine de celui-ci. Contrairement à l'étape de détection qui nécessite en général un seul résidu, l'isolation en met en jeu plusieurs.

Pour les résidus structurés présentés d'une manière booléenne, les (r_j) sont conçus de telle façon que chacun réagisse à un ensemble différent de défauts (f_i) et reste insensible vis-à-vis des autres. Le seuillage est appliqué sur chaque élément du vecteur de résidus dans le but d'obtenir un vecteur binaire constitué de 0 et de 1.

Par conséquent, une table de signatures ou une table d'incidences est nécessaire afin de lier chaque défaut à l'ensemble des résidus correspondants.

Cette table est de dimensions $n \times m$ avec n **le nombre de résidu** et m **le nombre de défauts distincts à détecter**. Une valeur 1 du résidu r_i au

niveau du défaut f_j indique que ce résidu est sensible à ce défaut.

- **table non localisante :** plusieurs défauts influencent les mêmes résidus, ce qui signifie que des colonnes sont identiques dans la table de signatures.
- **table faiblement localisante ou localisante d'ordre 1 :** les colonnes sont différentes mais si on change un 1 en 0, on retrouve deux colonnes identiques.
- **table fortement localisante :** les différentes colonnes sont distinctes et aucune d'entre elles ne s'obtiennent à partir d'une autre en changeant un 1 en 0.

	c_1	c_2	c_3
r_1	1	1	0
r_2	1	1	0
r_3	1	1	0

(a) non localisante

	c_1	c_2	c_3
r_1	1	0	0
r_2	1	1	0
r_3	0	1	1

(b) faiblement localisante

	c_1	c_2	c_3
r_1	1	0	1
r_2	0	1	1
r_3	1	1	0

(c) fortement localisante

Figure 2.21 – Exemples de tables de signature

Dans cette section, nous avons présenté les méthodes de diagnostic, avec focalisation sur les méthodes à base de modèle et d'observateurs à entrées inconnues. Ensuite, nous avons présenté la raison pour laquelle nous utilisons la notion de divergence de la théorie de l'information. En effet, les capteurs sont sujets à des erreurs, de façon que la mesure n'est pas ponctuelle, mais plutôt une variable aléatoire qui appartient à un nuage de points, délimité par une incertitude, définie grâce à la technologie du capteur. Nous avons ensuite détaillé la prise de décision concernant la présence (ou absence) d'un défaut, et comment l'étape d'isolation de défaut est finalement appliquée.

Dans la section suivante, nous présentons les méthodes guidées par les données. Tout d'abord, nous introduisons les notions d'intelligence artificielle, en particulier l'apprentissage machine, puis nous présentons un état de l'art de l'apport de cette technique en terme de localisation et de diagnostic.

2.4.4 Intelligence artificielle pour la tolérance aux fautes des défauts capteurs

Avant d'entrer dans les détails techniques, dressons un bref historique de l'intelligence artificielle. Dans la première moitié du XXe siècle, la science-fiction a familiarisé le monde avec le concept de robots artificiellement intelligents. Dans les années 1950, une génération de scientifiques, de mathématiciens et de philosophes avait assimilé culturellement le concept d'intelligence artificielle. C'est le cas de TURING, 1950, qui a suggéré que les humains utilisent les

informations disponibles ainsi que la raison pour résoudre des problèmes et prendre des décisions, une capacité qui peut être accordée à la machine. Cinq ans plus tard, la preuve de concept a été initiée par le biais du programme "Logic Theorist" conçu pour imiter les capacités de résolution de problèmes d'un être humain.

Dans la période qui a suivi, l'intelligence artificielle était en pleine expansion, donnant ainsi naissance aux réseaux de neurones, un domaine qui a catalysé le développement de l'apprentissage machine et a persuadé les agences gouvernementales de soutenir financièrement la recherche en intelligence artificielle. Avec le financement et le développement, la loi de Moore [1], l'IA a finalement rattrapé et, dans de nombreux cas, dépassé les besoins de ce temps. Par la suite, la naissance de l'"apprentissage profond" a eu lieu dans les années 1980, constituant des techniques qui permettaient aux ordinateurs d'apprendre en s'appuyant sur l'expérience. Les évènements majeurs dans l'historique sont présentés dans Fig.2.22.

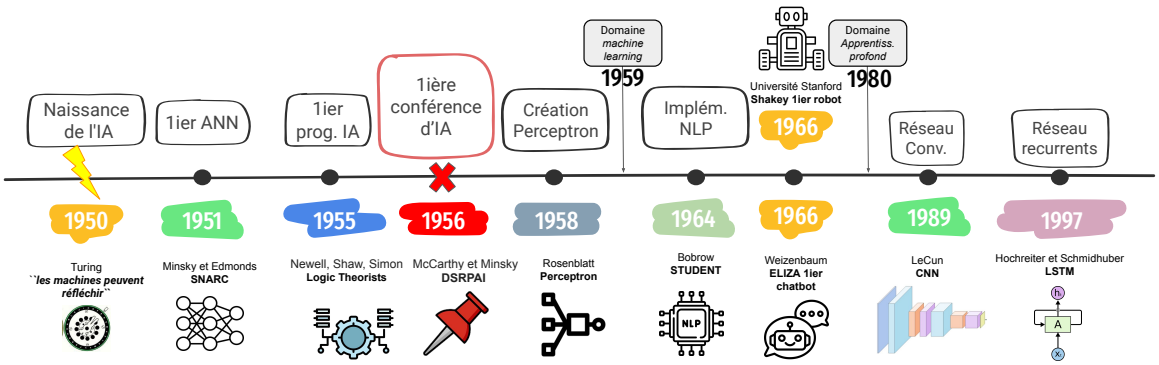


Figure 2.22 – Chronologie de l'évolution de l'apprentissage machine

En résumant cette évolution historique, deux sous-domaines de l'intelligence artificielle ont émergé : l'apprentissage machine (Machine Learning - ML) et l'apprentissage profond (Deep Learning - DL).

La différence entre chaque domaine est présentée par la Fig.2.23. L'intelligence artificielle correspond au développement de systèmes et de machines intelligents qui peuvent accomplir des tâches qui nécessitent souvent l'intelligence humaine.

L'apprentissage machine est un sous-domaine de l'intelligence artificielle qui permet automatiquement à une machine ou à un système d'apprendre et de s'améliorer à partir de l'expérience. Au lieu d'une programmation explicite, l'apprentissage machine utilise des algorithmes pour analyser de grandes quantités de données, tirer des comportements et prendre ensuite des décisions. Ce sous-domaine nécessite l'intervention humaine au cas où la décision prise était incorrecte. Plus particulièrement, parmi les types de l'apprentissage machine, on trouve les réseaux de neurones profonds, qui consiste essentiellement en un réseau neuronal à trois couches ou plus tentant de simuler le comportement

1. Loi de Moore : la mémoire et la vitesse des ordinateurs doublent chaque année

du cerveau humain. Ce type d'apprentissage profond cherche à atteindre des conclusions précises sans intervention de l'être humain.

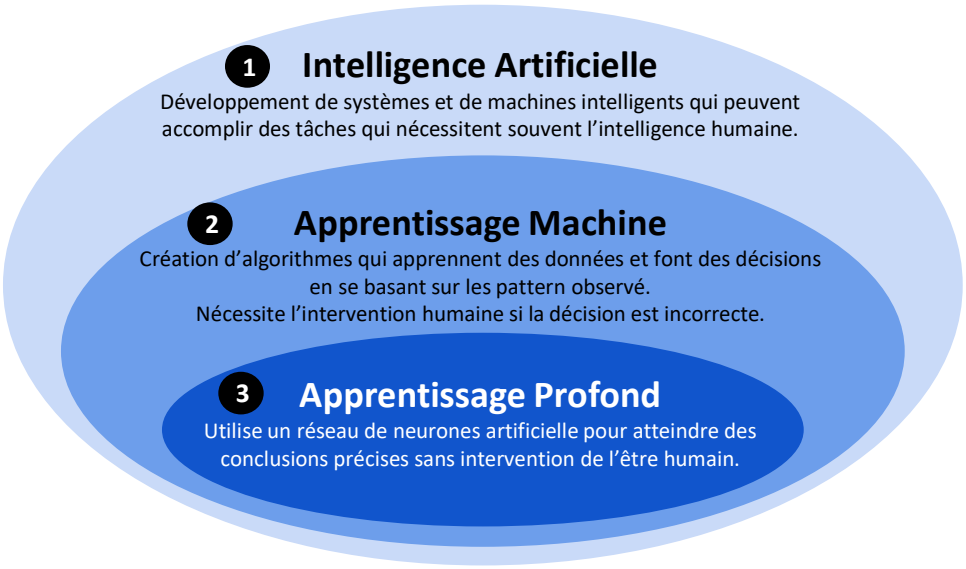


Figure 2.23 – Différents sous-domaines de l'intelligence artificielle et leurs fonctionnalités

Les applications de l'intelligence artificielle sont extrêmement variées, englobant de nombreux domaines tels que la biomédecine, la robotique, le commerce, la cyber-sécurité, l'assistance, le multimédia, et bien d'autres encore. Cette polyvalence de l'IA a ouvert la porte à un large éventail de possibilités et d'innovations dans de multiples secteurs.

Dans le domaine de la robotique, l'intelligence artificielle est employée pour des applications variées, notamment dans le diagnostic, la commande, la navigation, la planification de trajectoire ou la reconnaissance d'objets. L'intégration de l'IA permet aux robots de prendre des décisions en temps réel, d'interagir de manière plus adaptative avec leur environnement et de réaliser des tâches complexes de manière autonome, ce qui ouvre de nouvelles perspectives passionnantes dans le domaine de la robotique.

Les robots possèdent déjà un certain niveau d'intelligence, avec lequel, l'usage du ML est plus adapté que celui de l'IA. En effet, l'IA se concentre sur la création de systèmes qui imitent l'intelligence humaine d'une manière générale, tandis que le ML se concentre sur l'apprentissage machine à partir de l'expérience et de l'amélioration des données.

Dans les sections à venir, nous explorerons plus en détail les différentes facettes du ML dans le contexte de la tolérance aux défauts des capteurs. Nous commencerons par examiner les types de ML pertinents pour cette application spécifique, en mettant en lumière leurs avantages et leurs limites. Ensuite, nous nous plongerons dans l'utilisation du ML pour la localisation, en explorant

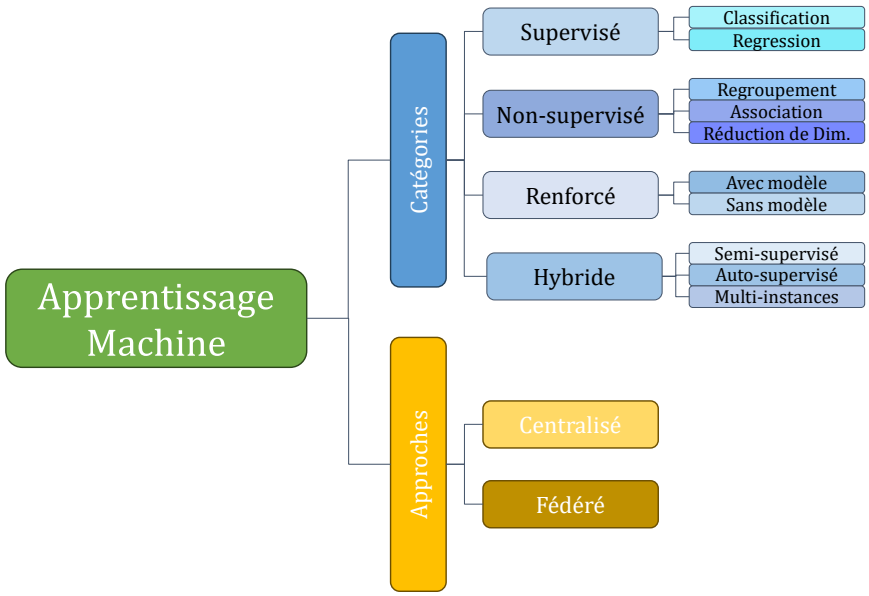


Figure 2.24 – Techniques d'apprentissage machine

comment les algorithmes du ML peuvent améliorer la précision des systèmes de positionnement, même en présence de capteurs défectueux.

Nous poursuivrons en examinant le rôle crucial du ML dans le diagnostic des capteurs, montrant comment il peut aider à identifier et à corriger les défauts de manière proactive. Enfin, nous aborderons l'intégration des méthodes guidées par les données et à base de modèle, démontrant comment la combinaison de ces approches peut aboutir à des systèmes de tolérance aux fautes des capteurs plus robustes et efficaces.

2.4.4.1 Types d'apprentissage machine

"*L'apprentissage est une recherche, dans l'espace des hypothèses possibles, d'une hypothèse qui donnera de bons résultats, même sur de nouveaux exemples au-delà de l'ensemble de formation. Pour mesurer la précision d'une hypothèse, nous lui donnons un ensemble d'exemples distincts de l'ensemble d'apprentissage*". Cette définition fondamentale de l'apprentissage comme formulée par Russell, 2010, met en lumière l'objectif central de l'apprentissage automatique : la capacité à généraliser à partir de données d'apprentissage pour résoudre des problèmes dans un contexte plus vaste.

Selon le problème d'apprentissage et les caractéristiques des données, l'apprentissage machine peut être classé selon des catégories, et des approches. Concernant les catégories, ce sont :

1. **Apprentissage supervisé** : selon Bishop, 2006, ce sont les applications dans lesquelles les données d'apprentissage comprennent des exemples de vecteurs d'entrée ainsi que les vecteurs cibles correspondants. Dans ce cas, les modèles sont formés à l'aide de données d'apprentissage contenant à la fois des entrées et des sorties. Ensuite, ces modèles sont utilisés

pour effectuer des prédictions sur des ensembles de test, où seules les entrées sont fournies. Les sorties générées par le modèle sont ensuite comparées aux valeurs cibles attendues, ce qui permet d'évaluer la performance et la compétence du modèle.

Les labels, qui sont les valeurs des sorties dans la base de données d'entraînement dont l'algorithme d'apprentissage cherche à apprendre, peuvent être classés en deux catégories principales : *discrètes ou continues*. Lorsqu'il s'agit de labels discrets et finis, cela correspond à ce que l'on appelle une classification. En revanche, lorsque les labels sont continus, on parle de régression. Les problèmes de classification et de régression peuvent avoir une ou plusieurs variables d'entrée et les variables d'entrée peuvent être n'importe quel type de données, telles que des données numériques ou catégorielles.

— Classification : La classification est une tâche fondamentale en apprentissage automatique où l'objectif est de regrouper des données en catégories ou classes distinctes en fonction de certaines caractéristiques ou attributs. Il existe différents types de classifications, dont deux des plus courants sont "One-Against-One (OAO)" et "One-Against-All (OAA)".

Dans l'approche OAO, un classifieur binaire est entraîné pour chaque paire de classes distinctes. Chaque classifieur binaire décide si une instance de données appartient à l'une des deux classes dans la paire ou non. Ensuite, lors de la phase de test, les décisions de tous les classifieurs binaires sont combinées pour attribuer une classe finale à l'instance de données. D'autre part, dans l'approche OAA, un classifieur binaire est formé pour chaque classe individuelle par rapport à toutes les autres classes. Chaque classifieur OAA décide si une instance de données appartient à la classe pour laquelle il a été formé ou non. Lors de la phase de test, les scores ou les décisions de chaque classifieur OAA sont évalués, et la classe avec le score le plus élevé est attribuée à l'instance de données.

Parmi les modèles utilisés pour la classification, on trouve :

■ Régression logistique (Logistic Regression) : Contrairement à ce que son nom pourrait laisser penser, la régression logistique fournit un mécanisme permettant d'appliquer les techniques de régression linéaire aux problèmes de classification, mais en changeant l'espace dans lequel ces valeurs sont calculées. Les modèles de régression logistique ou *logit* sont un cas particulier du modèle linéaire généralisé (GLM), pour les tâches de classification binaire et multi-classe. Ils fournissent un cadre probabiliste pour estimer la probabilité d'appartenance à une classe en fonction des caractéristiques d'entrée. Bien que cet outil ait été présenté pour la première fois par Cox, 1958 au milieu du XXe siècle, sa mise en œuvre a été retardée en raison des limitations des ressources informatiques, de la disponibilité d'autres algorithmes et de la reconnaissance progressive de son efficacité. Plus de détails peuvent

être trouvés dans l'Annexe A.

- Naïf bayes (naive bayes) : Comme l'indique son nom, les naive bayes sont des classifieurs probabilistes basés sur le théorème de Bayes, "naïf" puisqu'ils supposent que les classes sont indépendantes les unes des autres, même si ce n'est pas le cas dans la réalité. Ils font partie de la famille des algorithmes d'apprentissage génératif, et appliquent la classification en calculant la probabilité qu'un élément appartienne à une certaine catégorie ou classe en fonction de ses caractéristiques.
- Machines à vecteur de support (Support Vector Machine) : également appelées machines à noyau, elles fonctionnent en trouvant la meilleure ligne ou frontière possible (appelée hyperplan) qui sépare différentes classes ou groupes de points de données. L'idée principale est de maximiser la marge, qui est la distance entre l'hyperplan et les points de données les plus proches de chaque classe.

— Régression : Les modèles de régression dans l'apprentissage automatique sont utilisés pour modéliser et analyser la relation entre une variable dépendante (également appelée cible) et une ou plusieurs variables indépendantes (également appelées prédicteurs ou caractéristiques). L'objectif est l'interpolation pour trouver la fonction qui passe par tous les points, et l'extrapolation pour prédire la sortie en apprenant le comportement de l'entrée. Les *kernels* (noyaux) utilisés pour accomplir cette tâche varient selon le comportement des données.

— Classification et Régression : Certains types de modèles peuvent être trouvés dans les deux domaines d'applications :

- Arbres de décision (Decision Tree) : Comme défini par SAMMUT et al., 2010, un arbre de décision est un modèle de classification structuré en arbre, qui peut être compris par des utilisateurs non experts et peut être induit efficacement à partir de données. Cette technique a été développée dans les communautés statistiques et d'apprentissage automatique.

  Ces arbres sont appris de manière descendante, à l'aide d'un algorithme connu sous le nom de Top-Down Induction of Decision Trees (TDIDT), de partitionnement récursif ou d'apprentissage par division et conquête. L'algorithme sélectionne le meilleur attribut pour la racine de l'arbre, divise l'ensemble d'exemples en ensembles disjoints et ajoute les nœuds et branches correspondants à l'arbre. Les détails sur la sélection des attributs peuvent être trouvés dans l'Annexe B.

  Ce type d'apprentissage est largement déployé grâce à leur flexibilité, facilité d'interprétation et qu'il ne nécessite pas de préparation des données. Par contre, il est sujet à l'overfitting, coûteux, et de petites variations dans les données peuvent produire un arbre de décision très différent. Son usage figure dans

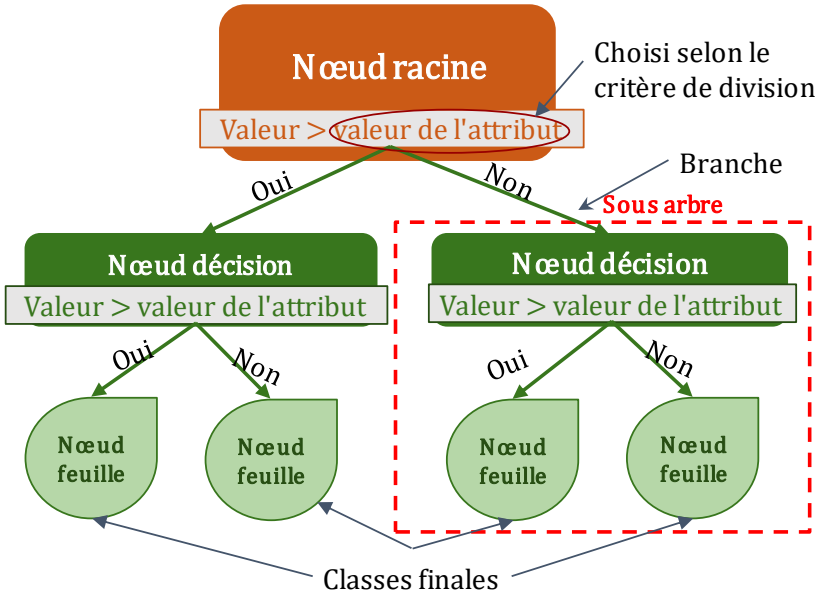


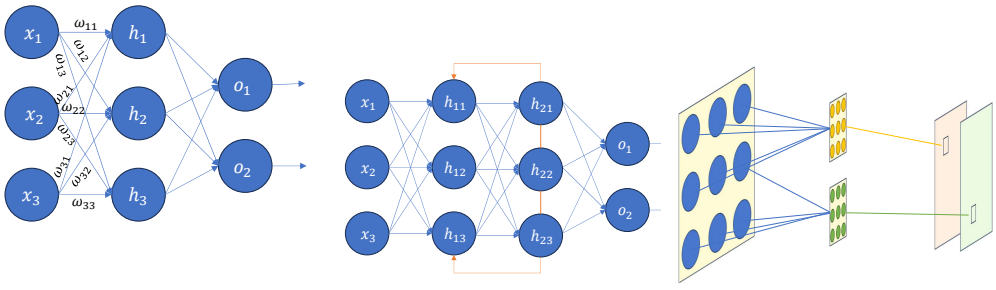
Figure 2.25 – Parties de l'arbre de décision

les domaines du diagnostic, de la détection d'anomalie ainsi que la gestion d'énergie et de réseaux.

- Forêts aléatoires (Random Forest) : Une forêt aléatoire se compose d'un certain nombre d'arbres de décision, où chaque arbre vote pour l'une des classes auxquelles appartient l'entrée désignée.
- Réseaux de neurones (NN) : Par la définition de D'Addona, 2014, un réseau neuronal artificiel (ANN), généralement appelé réseau neuronal (NN), est un modèle mathématique ou informatique qui s'inspire de la structure et/ou des aspects fonctionnels des réseaux de neurones biologiques. Un réseau neuronal se compose d'un groupe interconnecté de neurones artificiels et traite les informations en utilisant une approche connexionniste de l'informatique. Plus de détails peuvent être trouvés dans l'Annexe C. Une progression naturelle des NN est les réseaux de neurones profonds (DNN), qui étendent le concept en introduisant plusieurs couches cachées, ce qui leur permet de capturer des modèles et des relations complexes dans les données. Selon l'application, les connexions des couches et les fonctions d'activation, on peut distinguer 3 types de réseaux de neurones profonds :
  + *Perceptron multicouche (MLP) :* Goodfellow et al., 2016 définit le MLP comme un réseau neuronal artificiel qui traite les cas où la correspondance entre les entrées et les sorties n'est pas linéaire. La principale spécificité du MLP est sa capacité à traiter des données tabulaires et à résoudre un large

éventail de problèmes, notamment des tâches de régression et de classification. Chaque neurone d'une couche MLP est connecté à tous les neurones de la couche suivante, ce qui en fait une architecture polyvalente.

+ *Réseaux de neurones récurrents (RNN) :* ils sont tirés des graphes de calcul ayant un calcul récursif ou récurrent. Ces connexions récurrentes leur permettent de garder en mémoire les étapes précédentes du temps. L'unité de base d'un RNN est la "cellule". Les cellules traitent les données d'entrée à chaque pas de temps et mettent à jour leur état caché interne. Cet état caché sert de mémoire aux observations passées du réseau. L'architecture permet aux RNN de maintenir et de mettre à jour cette mémoire au fur et à mesure que de nouvelles données arrivent dans une séquence. Les modèles les plus renommés incluent les architectures de Long Short-Term Memory (LSTM), les unités récurrentes à portes (GRU), les Transformers et son extension Bidirectional Encoder Representations from Transformers (BERT) de Google.
+ *Réseau de Neurones Convolutifs (CNN) :* ils s'avèrent particulièrement efficaces pour les missions qui nécessitent des données organisées en grille, comme les images et les vidéos. Ils se démarquent par leur capacité à saisir les structures spatiales des données en utilisant des couches de convolution qui identifient les caractéristiques à différentes échelles. Les CNN jouent un rôle central dans des domaines liés à la vision par ordinateur, comme la catégorisation d'images, la détection d'objets et la segmentation d'images. En effet, les avancées dans ce domaine ont été à l'origine de la mise en place du défi d'apprentissage visuel à grande échelle d'ImageNet, qui a conduit à la notoriété de plusieurs modèles célèbres et à la disponibilité de modèles pré-entraînés tels que VGG, Inception, ResNet et MobileNet.



(a) Perceptron multi-couche **(b)** Réseaux récurrents **(c)** Réseaux convolutifs

Figure 2.26 – Types de réseaux de neurones

2. **Apprentissage non-supervisé** : Contrairement à l'apprentissage supervisé, dans un apprentissage non-supervisé les algorithmes sont utilisés pour extraire des structures et des relations inhérentes à un ensemble de données sans la présence d'étiquettes ou de réponses préalables. Selon l'objectif, il existe plusieurs types :
   — Regroupement (Clustering) : Dans cette approche, l'objectif est de regrouper les données en ensembles (clusters) où les éléments partagent des similitudes intrinsèques. Par exemple, le DBSCAN (Density-Based Spatial Clustering of Applications with Noise) Algorithm, le k-means, l'isolation forest, et les Autoencoders sont des techniques couramment utilisées pour cette tâche. Le DBSCAN identifie des groupes de densité dans les données, le k-means partitionne les données en clusters en minimisant la distance entre les points et les centres de cluster, l'isolation forest détecte les anomalies en isolant les observations atypiques, et les Autoencoders apprennent des représentations latentes des données.
   — Analyse d'association : Cette approche se concentre sur la découverte de règles d'association entre des éléments dans un ensemble de données. Par exemple, l'algorithme d'Apriori est utilisé pour extraire des règles d'association à partir de données transactionnelles, souvent utilisées dans le domaine du marketing pour identifier des tendances d'achat.
   — Réduction de dimension : L'objectif ici est de réduire la complexité des données en conservant les informations essentielles tout en réduisant le nombre de dimensions. L'Analyse en Composantes Principales (ACP) est un exemple classique de technique de réduction de dimension qui projette les données dans un nouvel espace en préservant leur variance maximale, permettant ainsi de simplifier l'analyse tout en minimisant la perte d'information.

   Certaines méthodes sont polyvalentes et sont utilisées à la fois dans des contextes supervisés et non supervisés, à l'exemple des "*k plus proches voisins*" (k-nearest neighbor - kNN). Cette technique peut être appliquée à des tâches de classification, de régression, voire de regroupement. Elle exploite la notion de proximité pour effectuer des classifications ou des prédictions concernant l'appartenance à un groupe d'un point de données individuel. Bien que kNN puisse être déployé pour des problèmes de régression ou de classification, il est généralement employé comme algorithme de classification, partant de l'idée que des points similaires se trouvent souvent à proximité les uns des autres.

3. **Apprentissage par renforcement** : L'apprentissage par renforcement (RL), selon SUTTON et al., 1998, est l'étude des programmes qui améliorent leurs performances dans une tâche donnée en recevant des récompenses et des punitions de l'environnement. Il est composé d'un agent qui apprend la décision dans un environnement, où sa situation actuelle est décrit par son état, ses décisions affectant l'environnement par des actions cadrés par des polices, et évalués par une rémunération/punition

accumulés et comparés à une fonction de valeur que l'agent peut atteindre à partir d'un état donné. Son usage est trouvé dans les domaines de l'automatisation industrielle, la robotique, les véhicules autonomes, l'agriculture et bien d'autres. Par exemple, D. GORDON et al., 2019 combine l'apprentissage par renforcement et les stratégies de planification afin d'obtenir des tendances à l'exploration et une résolution des problèmes orientée vers les objectifs.

Compte tenu de l'apprentissage, on peut distinguer deux sous-classes :

— À base de modèle : L'idée centrale du RL basé sur un modèle est de tirer parti d'un modèle appris ou pré-spécifié de l'environnement pour optimiser la prise de décision. Ce modèle saisit la dynamique de la réaction de l'environnement aux actions de l'agent, ce qui permet à ce dernier de simuler différents scénarios et de planifier ses actions en conséquence.

— Sans modèle : La limitation figurant dans la nécessité de représenter explicitement et d'apprendre des modèles d'action est une des raisons pour lesquelles les méthodes sans modèle ont été conçues. Les méthodes sans modèle telles que l'apprentissage Q permettent d'éviter complètement ce problème en apprenant des fonctions de valeur sur des paires état-action.

On peut trouver des méthodes hybrides tel que l'apprentissage semi-supervisé lorsque les données d'apprentissage contiennent très peu d'exemples étiquetés, l'apprentissage auto-supervisé qui se se réfère à un problème d'apprentissage non supervisé qui est formulé comme un problème d'apprentissage supervisé et l'apprentissage multi-instances qui est un problème d'apprentissage supervisé dans lequel des lots ou des groupes d'échantillons sont étiquetés, et non individuellement.

Outre les catégories pré-citées, les approches d'apprentissage peuvent également être classées selon la façon avec laquelle les modèles sont appris et utilisés. On distingue deux classes d'approches :

1. Apprentissage centralisé : c'est l'approche traditionnelle de l'apprentissage automatique dans lequel un serveur central ou une machine traite et analyse toutes les données d'apprentissage. Dans cette approche, toutes les données sont collectées et envoyées à un site central pour l'apprentissage du modèle. Le serveur central calcule les paramètres du modèle et distribue ensuite le modèle formé aux appareils ou clients participants.

2. Apprentissage décentralisé ou fédéré : c'est une approche d'apprentissage automatique plus récente conçue pour répondre aux préoccupations en matière de confidentialité et de décentralisation des données. Dans l'apprentissage fédéré, l'apprentissage du modèle se fait localement sur des appareils ou des clients décentralisés qui ont accès à leurs propres données. Au lieu d'envoyer des données à un serveur central, seules les mises à jour du modèle sont partagées avec un serveur central ou un agrégateur. Le serveur agrège ces mises à jour pour améliorer le mo-

dèle global tout en préservant la confidentialité des sources de données individuelles.

Plusieurs méthodes existent dans la littérature concernant la façon d'agrégation des modèles. La première méthode introduite est celle de *Federeated Averaging* (FedAvg) par McMahan et al., 2016. Dans cette méthode, avec K clients, chacun avec un sous-ensemble D_k de longueur n_k, l'agrégation sur les hyperparamètres ω des modèles est réalisée en appliquant l'équation suivante :

$$\omega_{t+1} = \sum_{k}^{K} \frac{n_k}{\sum_k^K n_k} \times \omega_{t+1}^k \tag{2.84}$$

Cet algorithme d'origine a subi plusieurs améliorations en terme de convergence tel que FedProx de T. Li, Sahu et al., 2020, renforcement d'équité tel que q-FedAvg de T. Li, Sanjabi et al., 2019 et adaptation du modèle aux besoins individuels des appareils participants tel que Per-FedAvg de Fallah et al., 2020.

Après avoir présenté les catégories et les approches d'apprentissage, nous passons aux applications de ces outils dans les domaines d'intérêt de cette thèse à savoir : la localisation et le diagnostic.

2.4.4.2 Usage de l'intelligence artificielle pour la localisation

La représentation efficace d'un système de localisation repose souvent sur sa bonne modélisation. Toutefois, en cas d'erreurs, les performances du système peuvent diverger. Cette divergence complique la prédiction précise du fonctionnement futur du système ou la classification de son état actuel au fil du temps. La réalité de la localisation est que les mesures des capteurs peuvent être imparfaites, la modélisation du système peut manquer de précision, l'environnement peut présenter une dynamique complexe, et des contraintes irréalistes peuvent affecter la précision et la fiabilité des systèmes évalués manuellement. En effet, dans les systèmes qui utilisent la vision par ordinateur, il est difficile de modéliser des facteurs tels que les zones dépourvues de repères, les variations d'éclairage, les flous de mouvement et l'étalonnage précis de la caméra, d'où la nécessité de l'automatisation pour éviter l'effort humain pour spécifier l'ensemble des connaissances sur les règles mathématiques et physiques, et assurer la résilience dans ces situations.

La localisation basée sur l'apprentissage profond a récemment fait l'objet d'une attention particulière, surtout lorsqu'elle est assistée par la vision par ordinateur. L'enquête de C. Chen et al., 2020 définit une taxonomie des approches d'apprentissage existantes, pertinentes pour la localisation et la cartographie, catégorisées en estimation odométrique, cartographie, localisation globale et SLAM, et fait une bonne étude du domaine.

Grâce aux bases de données à accès public comme KITTI-360, A2D2 d'Audi, Apolloscape, Waymo, et Cityscapes, l'étude et la comparaison des performances des méthodes est possible, avec le suivi de la citation et la mise en œuvre de ces

bases de données, grâce aux statistiques sur les plateformes telles que Papers with code.

Diverses méthodes ont été développées pour améliorer la localisation. Certaines sont basées sur l'odométrie visuelle, exploitant les capacités de la vision par ordinateur pour cette finalité, ou en ajoutant les informations des centrales inertielles, en se basant sur l'odométrie télémétrique, ou bien combinant les modèles physiques avec l'apprentissage.

Parmi ces approches, PoseNet de KENDALL et al., 2015 qui traite un algorithme de re-localisation qui utilise un CNN pour obtenir la pose 6-DoF de la caméra par rapport à une scène. Une autre approche était celle DeepVO de S. WANG et al., 2017, qui se sont intéressés par l'utilisation d'une combinaison des CNN et de RNN pour permettre l'apprentissage complètement guidé par les données de l'odométrie visuelle (voir Fig.2.27). GeoNet de YIN et al., 2018 adopte une approche différente, en divisant son processus d'apprentissage en deux sous-tâches en estimant séparément les structures statiques de la scène et la dynamique des mouvements grâce à un reconstructeur de structures rigides et à un localisateur de mouvements non rigides, suivant un apprentissage auto-supervisé. Une autre approche intéressante consiste en l'intégration de l'apprentissage multitâche, où les tâches de régression de la pose globale et d'estimation de l'odométrie visuelle sont apprises simultanément. C'est le cas de VLocNet de VALADA et al., 2018, permettant au modèle de tirer parti des similitudes entre les caractéristiques propres à chaque tâche et d'exploiter les caractéristiques complémentaires apprises dans le cadre de différentes tâches. Notamment, VLocNet se positionne comme l'une des premières techniques d'apprentissage en profondeur à rivaliser avec les approches de pointe basées sur le Scale-Invariant Feature Transform (SIFT) en vision par ordinateur, et dans certains cas, à les surpasser. Quant au LiDAR, Q. LI et al., 2020 présentent LO-Net, un réseau d'estimation de l'odométrie lidar appelé LO-Net, qui apprend simultanément à estimer les normales et un masque pour les régions dynamiques. Cet algorithme affiche des performances compétitives par rapport à l'algorithme conventionnel de pointe, à savoir l'algorithme LIDAR Odometry and Mapping (LOAM).

En complément aux capteurs mentionnés précédemment, certaines méthodes intègrent à la fois des caméras et des unités de mesure inertielle (IMU) pour améliorer la précision de la localisation. VINet de CLARK et al., 2017 a été pionnier dans la formulation de l'odométrie visuelle-inertielle comme un problème d'apprentissage séquentiel. Il utilise un encodeur visuel basé sur un réseau neuronal convolutionnel (CNN) pour extraire les caractéristiques visuelles à partir de deux images RVB consécutives, et un encodeur inertiel pour extraire les caractéristiques des données IMU grâce à un réseau de mémoire à long terme (LSTM). De plus, L. LI et al., 2023 ont proposé D-GLSNet, une méthode novatrice de localisation géographique en combinant des cartes satellites avec des nuages de points LiDAR. Cette méthode inclut un réseau de correspondance 2D-3D basé sur un transformateur. D-GLSNet permet une correspondance directe entre les nuages de points LiDAR et les images satellite grâce à un processus d'apprentissage complètement guidé par les données

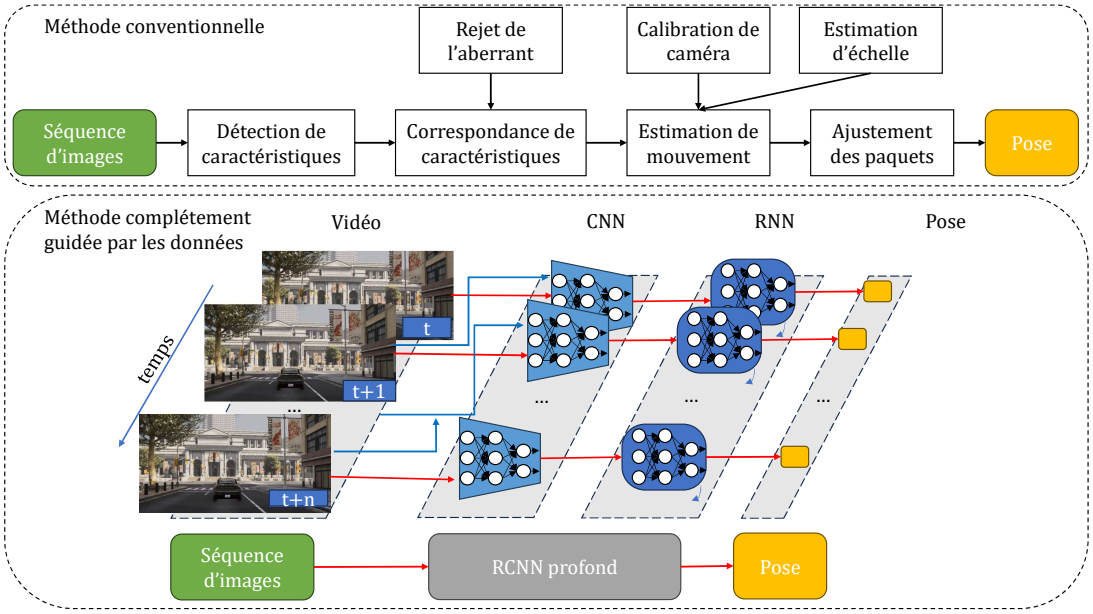


Figure 2.27 – Architecture et fonctionnement de DeepVO de S. WANG et al., 2017

(end-to-end).

D'autres travaux combinent des modèles de mouvement physique avec des réseaux de neurones. Une première application a été faite par JWO et al., 2004 qui présentent l'utilisation d'un réseau de neurones à rétropropagation (BP) pour améliorer les performances du filtre de Kalman dans le positionnement DGPS en estimant les variances variables dans le temps et les covariances du bruit. HAARNOJA et al., 2017 proposent une nouvelle approche pour l'estimation d'état dans les tâches visuelles en utilisant une combinaison de filtre de Kalman à rétropropagation (BKF) et de réseaux de neurones récurrents (RNN). L'objectif est de gérer l'incertitude dans l'estimation de l'état en incorporant la connaissance du domaine à partir du filtre de Kalman et en apprenant la dynamique à partir des données à l'aide des RNNs. D'autre part, JONSCHKOWSKI et al., 2018 présentent les filtres particulaires différentiables (DPF) comme une méthode permettant de combiner l'apprentissage complètement guidé par les données avec des antécédents algorithmiques pour l'estimation d'état en robotique. L'apprentissage complètement guidé par les données optimise les modèles en termes de performance, tandis que les priorités algorithmiques permettent d'expliquer et de régulariser l'apprentissage. REVACH et al., 2022 proposent le KalmanNet, un modèle qui combine des réseaux de neurones récurrents avec des modèles d'espace d'état pour apprendre l'opération de filtrage à l'aide de données et de connaissances partielles du domaine, plutôt que d'estimer explicitement les paramètres manquants du modèle d'espace d'état (voir Fig.2.28).

En conclusion, les modèles d'apprentissage sont incorporés dans les systèmes de localisation et de cartographie pour relier les données des capteurs d'entrée et les valeurs cibles, en extrayant automatiquement les caractéristiques

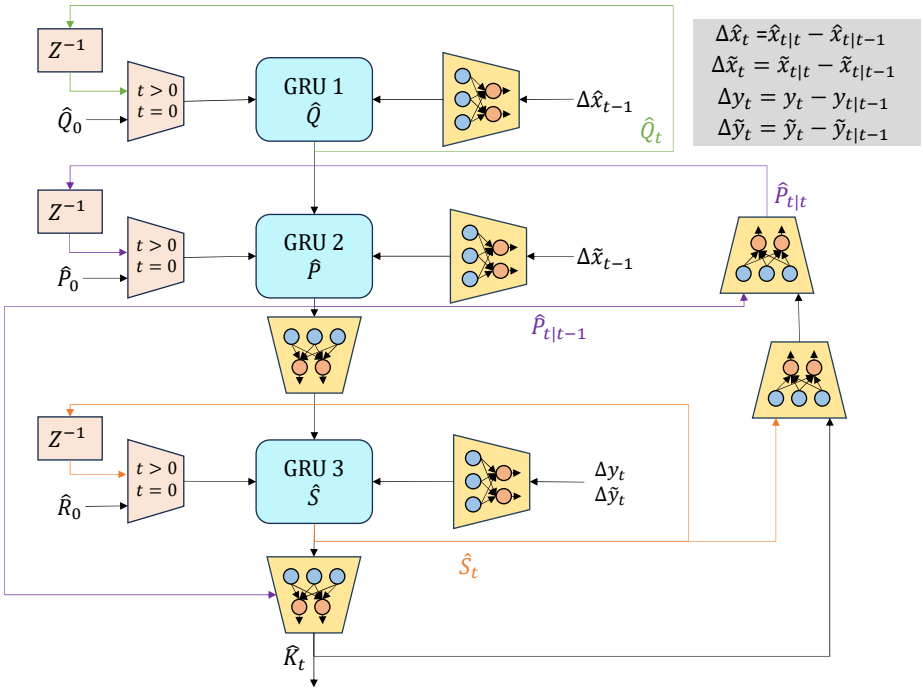


Figure 2.28 – Architecture de KalmanNet de REVACH et al., 2022

utiles des données brutes sans aucun effort humain. Les techniques basées sur l'apprentissage profond ont jusqu'à présent atteint des performances de pointe dans une variété de tâches, de l'odométrie visuelle à la localisation globale en passant par la reconstruction de scènes denses. Grâce à la capacité hautement expressive des réseaux de neurones profonds, ces modèles sont capables de modéliser implicitement des facteurs tels que la dynamique de l'environnement ou les bruits des capteurs, qui sont difficiles à modéliser à la main, et sont donc relativement plus robustes dans les applications du monde réel.

2.4.4.3 Usage de l'intelligence artificielle pour le diagnostic

Dans le domaine de la FDI de défauts, il est pertinent de souligner que les techniques d'intelligence artificielle (IA) ont été systématiquement mises en œuvre au fil du temps. À ses débuts, l'intérêt principal se concentrait sur les systèmes non linéaires et la détection de motifs pouvant entraîner des comportements anormaux dans ces systèmes. Cette orientation initiale est bien illustrée par les travaux de Y. CHEN et al., 1998 qui ont mis en évidence diverses méthodes de surveillance et de diagnostic largement utilisées à l'époque. Parmi lesquelles, pas uniquement les méthodes à base de modèle, mais aussi celles fondées sur la logique floue, les approches reposant sur les connaissances expertes ainsi que celles guidées par les données tel que les réseaux de neurones artificiels, les arbres de décision, les réseaux bayésiens et les systèmes neuro-flous. Il convient également de noter que, parallèlement à ces développements, de nombreuses investigations ont été menées, en particulier dans le domaine des processus chimiques, où le diagnostic occupe un très grand intérêt.

Quant aux processus industriels, le livre de SOBHANI-TEHRANI et al., 2009 présente les premières mises en œuvre et utilisations de ces outils. Initialement, la génération et la classification de résidus à modèles multiples basées sur des réseaux de neurones, proposée à l'origine par PATTON et al., 1999, suit l'idée d'un schéma d'FDI à modèles multiples où les modèles mathématiques ont été remplacés par des identificateurs dynamiques parallèles basés sur des réseaux de neurones. Il comporte deux phases : la génération de résidus à partir de modèles multiples basés sur des NN et la prise de décision au cas d'isolation de défauts (voir Fig.2.29).

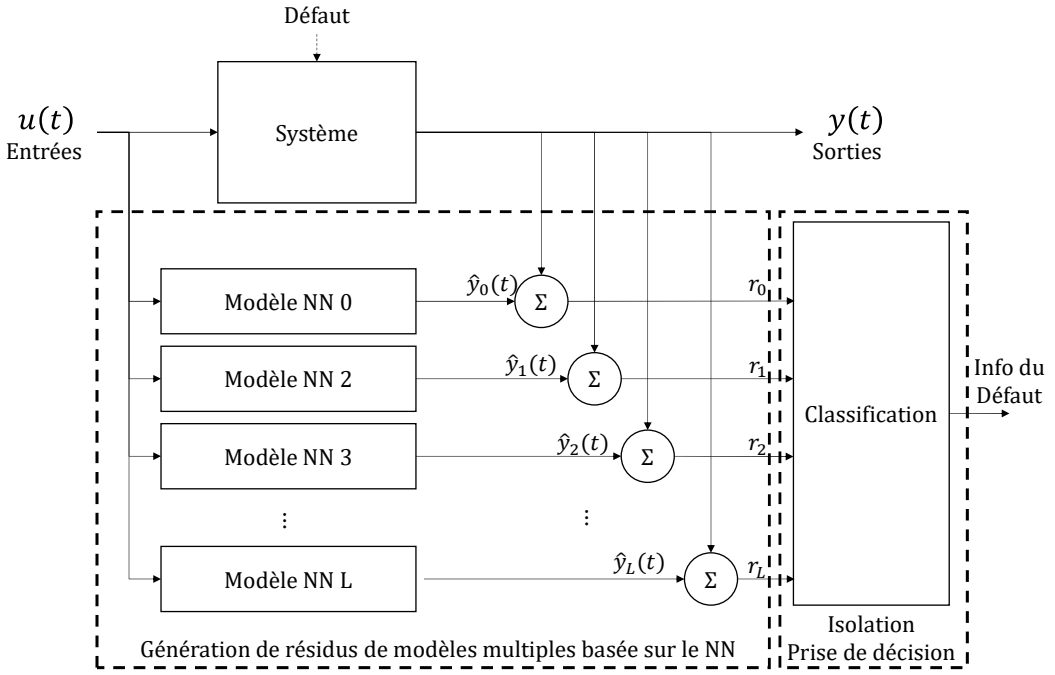


Figure 2.29 – Schéma générique de FDI de défauts à modèles multiples basé sur un réseau neuronal, par SOBHANI-TEHRANI et al., 2009

Un ajout à la génération de résidus à l'aide de données historiques quantitatives d'entrée-sortie du système a été introduit par J. CHEN et al., 1999, portant sur l'apprentissage (ou la détermination) des paramètres du modèle flou du système à partir de données quantitatives du système. Cette intégration des connaissances quantitatives et qualitatives du système est réalisée au moyen d'un système neuro-flou (ou d'un réseau neuronal flou) qui permet de combiner la capacité d'apprentissage des réseaux de neurones avec la représentation explicite des connaissances de la logique floue (voir Fig.2.30).

D'autres méthodes utilisent les arbres de décision étant donnée son adaptation au problème de décision et de diagnostic hiérarchisé comme l'arbre de classification à auto-apprentissage (SELECT) développé par FÜSSEL, 2002, qui combine des idées issues du domaine des arbres de décision avec les propriétés d'adaptation des réseaux de neurones et l'interprétation de la logique floue.

Depuis ces premières implémentations, le domaine s'est élargi et a continué à évoluer. Dans le domaine des véhicules autonomes par exemple, le diagnostic

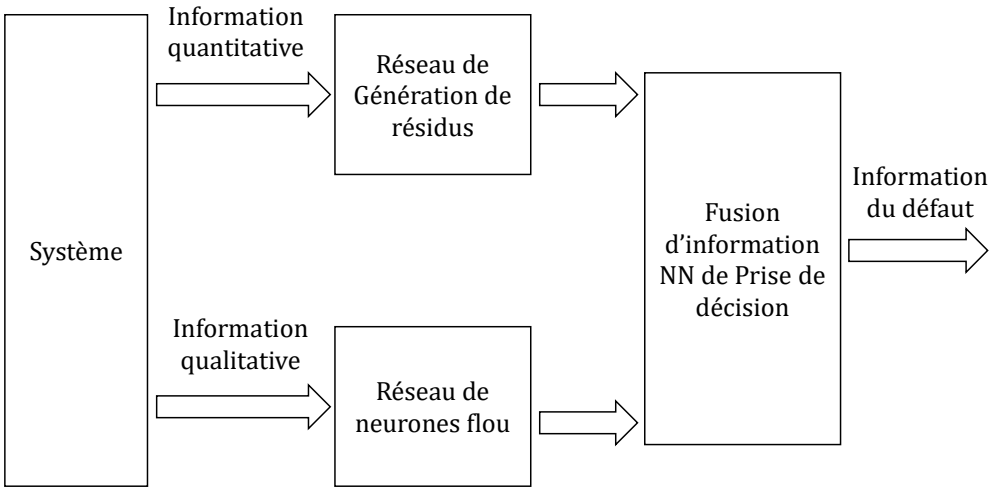


Figure 2.30 – Structure conceptuelle du diagnostic de panne basé sur l'informatique intégrée de J. Chen et al., 1999

de la méthode de Shi et al., 2021 porte sur le cas de données déséquilibrées et s'applique en comparant les valeurs estimées de la vitesse de lacet et de la vitesse latérale avec les valeurs mesurées par les capteurs du véhicule. Les vecteurs résiduels générés par cette comparaison sont ensuite classés à l'aide d'un algorithme amélioré de machine à vecteurs de support (SVM) pour diagnostiquer les défauts.

S. Chen, Yu et al., 2022 présentent une approche utilisant le 1-DCNN (réseau neuronal convolutionnel unidimensionnel) et le NAS (recherche d'architecture neuronale) basé sur l'apprentissage par renforcement pour la FDI de processus dans les processus industriels complexes. Le modèle 1-DCNN extrait des représentations de caractéristiques discriminantes des signaux de processus 1-D, qui peuvent être utilisées pour la classification, permettant au modèle d'apprendre des représentations de caractéristiques sensibles aux défauts de manière supervisée.

Par ailleurs, une avancée significative dans le domaine de la détection et de l'isolation de défauts réside dans l'exploitation des réseaux de neurones à perturbation. Ces réseaux, également connus sous le nom de réseaux de neurones adversaires (GANs - Generative Adversarial Networks), ont émergé comme un outil puissant pour améliorer la robustesse des systèmes de surveillance et de diagnostic. Les GANs consistent en deux réseaux de neurones antagonistes, un générateur et un discriminateur, qui s'affrontent dans un processus d'entraînement itératif. Le générateur tente de créer des données de défauts potentielles, tandis que le discriminateur vise à les distinguer des données normales. L'utilisation de réseaux de neurones à perturbation dans la FDI est particulièrement pertinente dans les systèmes où les défauts peuvent être masqués ou camouflés par des variations normales. Les GANs peuvent aider à générer des scénarios de défauts réalistes et difficiles à distinguer des données normales, permettant ainsi d'améliorer la sensibilité de la détection. Guo et al., 2020 propose un

réseau GAN unidimensionnelle multi-label (ML1-D-GAN) d'abord utilisé pour générer des données de défauts réels. Ensuite, les données générées et les données de défauts réels sont toutes deux utilisées pour former le classificateur de défauts (voir Fig.2.31).

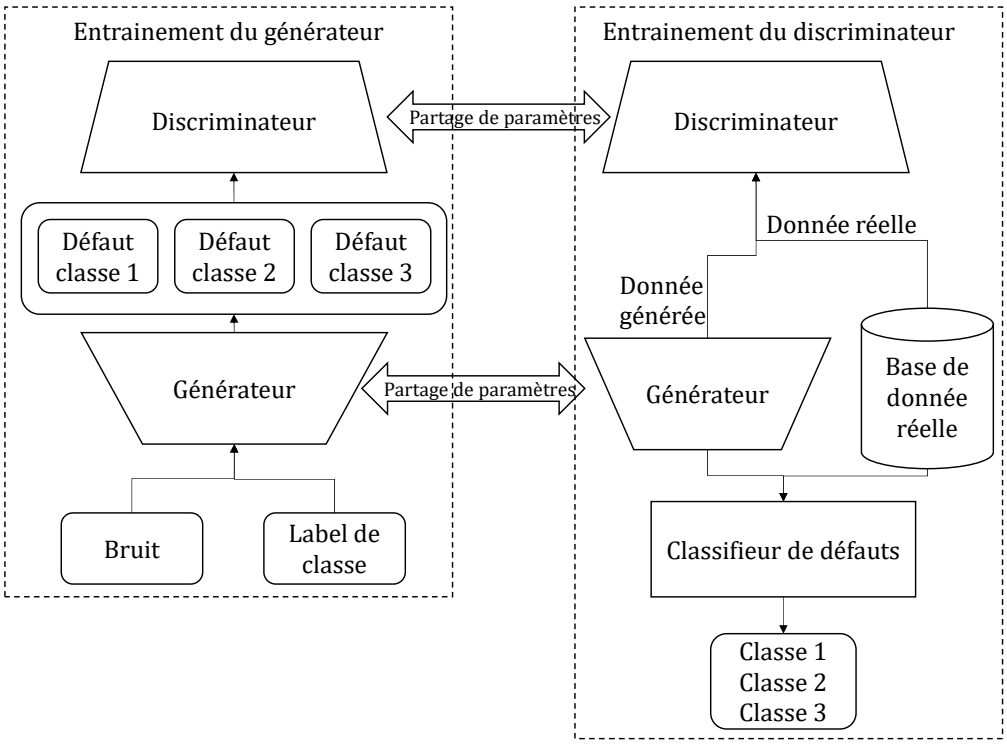


Figure 2.31 – Structure du ML1-D-GAN de Guo et al., 2020

En conclusion, l'évolution constante des techniques d'intelligence artificielle dans le domaine de la FDI de défauts reflète leur adaptabilité et leur capacité à répondre aux défis complexes de la sécurité et de la fiabilité des systèmes. Ces avancées ont permis de repousser les limites de la compréhension des comportements anormaux des systèmes, ouvrant ainsi la voie à des applications variées.

2.4.5 Combinaison des méthodes de diagnostic

Les systèmes dynamiques représentés par une méthode basée sur un modèle pour décrire son comportement attendu présentent un défi par rapport aux scénarios de défaillance. La limite des techniques basées sur les modèles est qu'elles dépendent fortement de modèles mathématiques précis et complets du système considéré. Les écarts ou les incertitudes dans ces modèles peuvent conduire à des diagnostics inexacts, en particulier dans une situation erronée qui peut s'accumuler. En outre, les approches fondées sur des modèles peuvent se révéler difficiles à mettre en œuvre face à des situations nouvelles ou imprévues qui sortent du cadre du modèle établi, ce qui peut conduire à des diagnostics manqués. D'autre part, les modèles complexes basés sur des données, tels que

les réseaux de neurones profonds, manquent souvent d'interprétabilité, ce qui rend difficile la compréhension du raisonnement qui sous-tend un diagnostic. Les systèmes devenant de plus en plus complexes, le besoin de techniques de diagnostic avancées est impératif.

Une approche qui a fait l'objet d'une attention considérable dans le domaine du diagnostic est la combinaison des techniques basées sur les modèles et des approches guidés par les données. La combinaison de ces deux approches permet d'exploiter leurs forces, car en intégrant des techniques basées sur des modèles avec des méthodes basées sur des données, les hypothèses du modèle et les comportements du système peuvent faire l'objet d'une validation croisée. En effet, les modèles sont conçus pour présenter les lois physiques, la dynamique du système et les interdépendances entre les différents composants du système. Cette représentation structurée permet aux techniques à base de modèles de fournir un contexte et une connaissance du domaine précieux aux techniques basées sur les données, en réduisant l'espace de recherche et en concentrant leur analyse sur des domaines d'intérêt spécifiques, et rendant le diagnostic capable de distinguer les scénarios normaux des scénarios défectueux. Cela relève des capacités de l'analyse discriminante qui se concentrent sur la modélisation explicite des limites entre les différentes classes et visent à classer les nouvelles instances sur la base de ces limites.

Une toute première mise en œuvre de ces techniques pour la détection des défauts des capteurs a été réalisée par Goel et al., 2000, où de multiples estimateurs parallèles du filtre de Kalman ont été déployés, chacun incorporant des modèles du comportement du système correspondant à différents types de défauts, provenant soit de capteurs, soit d'actionneurs. À partir de ces estimateurs, les résidus ont été calculés sur la base de la distance de Mahalanobis et utilisés comme entrée d'un réseau neuronal à rétro-propagation, traitant cet ensemble de résidus comme un modèle (voir Fig.2.32)

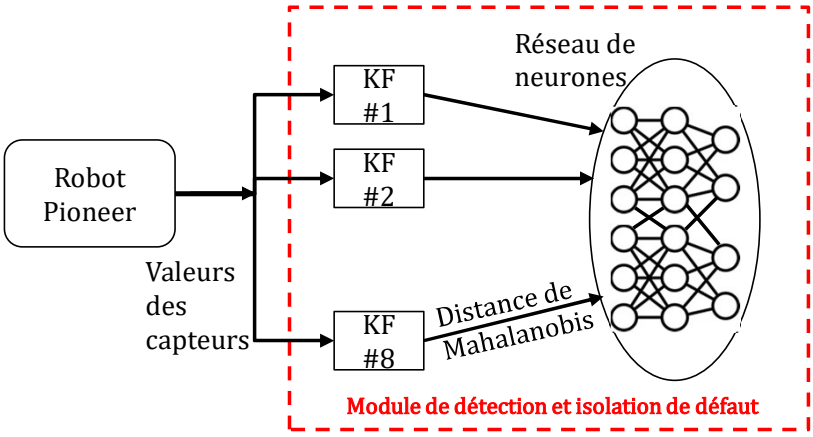


Figure 2.32 – Représentation graphique de l'algorithme de Goel et al., 2000

Plusieurs applications ont eu lieu dans ce domaine, et en utilisant des techniques variés d'intelligence artificielle :

Christensen et al., 2008 ont appliqué les réseaux de neurones pour la détection de défauts afin de déterminer l'état du système en contrôlant active-

ment le robot et en enregistrant le flux de données capteurs et de signaux de commande. Ils utilisent comme classificateurs des réseaux de neurones à retardement (TDNN), qui sont des réseaux à progression directe permettant de raisonner sur la base d'entrées variables dans le temps sans utiliser de connexions récurrentes.

C. Wang et al., 2009 utilisent un détecteur de réseau neuronal à rétropropagation (BP) à 3 couches avec une fonction d'activation sigmoid dans la conception et la mise en œuvre d'un système de surveillance pour le contrôle d'une formation multi-robots afin d'analyser les données des odomètres des robots et de détecter les dysfonctionnements. Le réseau de neurones est entraîné à l'aide d'exemples positifs et négatifs, et l'objectif du système de surveillance est de déterminer si un robot peut se rétablir rapidement ou s'il doit être retiré de l'équipe en fonction de ses performances.

Fang et al., 2020 présentent la détection des défauts d'état dans les véhicules autonomes à l'aide d'une machine à vecteur de support (SVM) à une classe. Elle forme une courbe limite qui sépare efficacement les domaines sûrs et dangereux du comportement du véhicule, dont la position actuelle est prédite à l'aide d'un observateur à filtre de Kalman conçu en utilisant le modèle cinématique linéaire du bicyclette du véhicule.

Y. Huang et al., 2022 proposent un modèle de fusion modifié pour le diagnostic des défauts des capteurs et l'estimation de la dégradation des performances des moteurs aéronautiques, combinant des modèles embarqués et un modèle piloté par les données pour l'obtention des résidus. Son cadre comprend un algorithme de transmission bidirectionnelle des informations pour faciliter la coordination des fonctions, avec un filtre de Kalman comme algorithme optimal, et un processus normalisé de sélection des paramètres des capteurs.

Un autre article combinant des approches basées sur des modèles et des approches basées sur des données est celui de H. Jin et al., 2022, où un filtre de Kalman adaptatif a été combiné à la méthode du rapport de vraisemblance généralisé et suivi d'une stratégie de mesure de l'écart basée sur les données pour traiter les défauts des composants, en particulier les défauts naissants avec des perturbations ou des bruits.

En conclusion, ces méthodes permettent de repousser les limites de la compréhension des systèmes complexes et de renforcer la sécurité et la fiabilité des processus et des systèmes. Elles incarnent ainsi une avancée significative dans le domaine du diagnostic, ouvrant la voie à de nouvelles opportunités pour une gestion proactive des risques et une amélioration continue des performances.

2.4.6 Conclusion du chapitre

En conclusion, ce chapitre traite des différents prérequis et notions initiales liés au domaine de la localisation coopérative tolérante aux fautes. Nous avons examiné chaque sous-domaine qui engendre cette solution : la localisation, les systèmes multi-robots en général, leur communication et leur perception. Ensuite, afin d'illustrer l'utilisation de ces informations, nous avons présenté les méthodes d'estimation de position. Ces solutions sont sujettes à des erreurs,

c'est pourquoi nous avons également présenté les méthodes de diagnostic disponibles dans la littérature, en mettant l'accent sur les méthodes basées sur l'intelligence artificielle et leur utilisation dans les deux domaines, à savoir la localisation et le diagnostic. Enfin, nous avons discuté des méthodes combinant le diagnostic basé sur un modèle et guidé par les données.

Dans le chapitre suivant, nous développerons l'approche déployée pour répondre à la question de la localisation en coopération, en augmentant simultanément la précision et la tolérance du système vis-à-vis des défauts, en appliquant le diagnostic combinant la théorie de l'information et l'intelligence artificielle.

Chapitre 3

Approche de localisation coopérative tolérante aux fautes proposée

Après avoir présenté les divers aspects nécessaires, ce chapitre présente l'approche adoptée afin de créer une solution de localisation coopérative, qui sera tolérante aux fautes. Tout d'abord, nous présentons la formulation et les hypothèses de travail, puis le protocole de communication et l'architecture choisie. Par la suite, nous présentons la modélisation du système en termes de modèle d'évolution et d'observation. Nous passons à la phase de diagnostic et la création des observateurs afin de synthétiser les indicateurs de défauts. Ensuite, nous définissons les exigences des données capteurs, afin de créer une base de données capteurs, à laquelle divers scénarios de défaut sont appliqués. Enfin, nous présentons les deux méthodes proposées pour la résolution du problème : la méthode classique à base de modèle et de théorie d'information, et la méthode mixte à base de modèle et guidée par les données.

3.1 Positionnement par rapport à l'état de l'art

Le titre de cette thèse comporte la création d'une solution de *localisation*[1] en *coopératif*[2], d'un système *multi-véhicules*[3], qui est *tolérante aux défauts*[4]. Dans la Fig.3.1, nous définissons chaque terme à part, puis nous regroupons afin de produire la définition finale.

Initialement, l'objectif est de "localiser", qui correspond dans notre cas à l'estimation de la pose du robot par rapport à un repère absolu de précision métrique. Afin de réaliser l'estimation, nous utilisons des filtres gaussiens, plus particulièrement le filtre informationnel étendu et le filtre d'intersection de covariances par lots. La raison pour laquelle nous choisissons ces filtres en particulier est leur adaptation au cas de fusion de données multi-capteurs.

Le second terme "coopérative" signifie que la tâche est accomplie par plusieurs véhicules agissant indépendamment afin d'apporter leur contribution respective sur l'ensemble pour atteindre des objectifs individuels. Avec ce mode de fonctionnement, l'architecture adoptée est décentralisée, signifiant que le

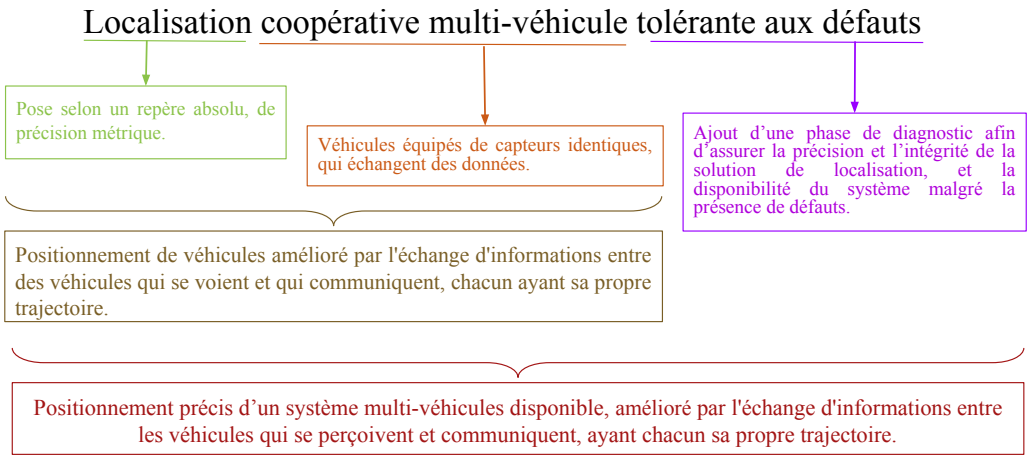


Figure 3.1 – Schéma montrant le positionnement par rapport à l'état de l'art

traitement se fait localement sur chaque véhicule, et seules les contributions informationnelles sont communiquées. La communication se fait via wifi et possède deux versions dont les protocoles seront détaillés dans la suite du chapitre. La perception utilisée est relative, signifiant que les capteurs de perception sont utilisés pour observer les autres véhicules, et non pas l'environnement.

En regroupant les trois premiers termes "localisation coopérative multi-véhicule", nous caractérisons que la coopération se fait au niveau de la localisation, avec une multitude de véhicules qui sont physiquement indépendants. Ces véhicules estiment leur propre pose. Cette estimation se fait d'une façon identique étant donné que les véhicules possèdent exactement les mêmes capteurs. De plus, ils bénéficient grâce à l'échange d'information avec d'autres véhicules que ces derniers perçoivent/les aperçoivent d'une correction de leur pose.

Le dernier terme de "tolérance aux défauts" signifie que les véhicules possèdent une phase de diagnostic, pour faire face aux défauts qui peuvent avoir lieu dans le système, et par suite assurer son fonctionnement en continuité. Dans notre cas, les méthodes de diagnostic utilisées sont de deux types : à base de modèles et guidés par les données. Pour le premier type, les observateurs à entrée inconnue sont utilisés afin de générer des résidus qui seront sensibles aux défauts des capteurs. Ces observateurs sont conçus sous l'hypothèse stochastique non déterministe, qui assimile la mesure des capteurs à des distributions de probabilité, dont la cohérence est quantifiée grâce à la notion de divergence de la théorie de l'information. La détection et l'isolation de défaut sont ensuite basées sur des résidus à base de divergences. La valeur de ces résidus par rapport à un seuil reflète la présence d'un défaut. Ce seuil est généré grâce à des critères de la théorie de l'information (la courbe ROC dans notre cas), ou bien tiré des données par l'usage de l'apprentissage machine. Cela se fait à l'aide d'algorithmes d'apprentissage supervisés de classification, qui prédisent le fonctionnement en défaut/sans défaut du système.

En regardant donc la phrase complète, l'objectif est de créer une solution de

localisation multi-capteurs, qui utilise des filtres gaussiens adaptés à la fusion de données, sous une architecture décentralisée. Cette localisation est faite sur un ensemble de véhicules terrestres homogènes, qui cherchent à réaliser la localisation d'une façon coopérative. Cette coopération n'est pas contraignante, c'est-à-dire que chaque véhicule possède sa propre cible et agit indépendamment des autres. Afin d'augmenter la précision de cette localisation, une phase de diagnostic est appliquée, qui est basée sur des observateurs à entrée inconnues adaptés à la théorie de l'information, grâce auxquels des résidus à base de divergence sont générés. La décision de la présence de défauts est faite de deux manières : par le seuillage de ces résidus, ou par la classification des décisions de fonctionnement tenant compte de données de scénario de défauts labellisés à partir desquels le comportement des résidus impliquant la présence de défauts est déduit.

3.2 Formulation et hypothèses du travail

Le problème à résoudre concerne le développement d'une solution de localisation précise qui dépend fortement des mesures des capteurs et qui est capable de vérifier si elles sont valides ou non. Dans un système de localisation multi-capteurs, les capteurs présentent une redondance naturelle car ils considèrent les mêmes informations mais utilisent différentes technologies pour observer l'environnement autour. Cette redondance est un aspect bénéfique car elle améliore la robustesse globale et la précision du système de localisation.

De plus, chaque capteur présente ses propres limitations inhérentes et incertitudes liées à ses mesures. Ces incertitudes peuvent être dues à divers facteurs tels que la qualité du capteur, les conditions environnementales ou les limitations technologiques. Par conséquent, la méthode d'estimation de position choisie doit être capable de fusionner efficacement les données provenant de plusieurs capteurs tout en prenant en compte leur nature stochastique. Pour répondre à la nature des données, des résidus à base de divergences sont conçus. Ces divergences tirées de la théorie d'information mesurent la dissimilarité entre deux mesures de capteurs, assimilés à des distributions de probabilités.

En dehors des limites de leurs incertitudes, les capteurs peuvent passer à des situations anormales. Ces défauts peuvent provenir de diverses sources, notamment de problèmes de calibration, de dommages physiques ou d' interférences de facteurs externes. Lorsque des erreurs se produisent, elles peuvent se propager dans le système et avoir un impact significatif sur la précision globale et la fiabilité de la solution de localisation. La précision de la solution de localisation est nécessaire étant donné qu'elle est l'entrée de l'algorithme de navigation autonome et de planification de trajectoire. Par conséquent, il devient crucial de développer une méthode capable de distinguer les états de fonctionnement du système (avec/sans défaut) et de localiser les éventuels capteurs défectueux (mesures erronées).

Pour tester et valider l'efficacité de cette approche à résoudre le problème, des données et des tests sont requises. Afin de construire cette solution, nous supposons les hypothèses comme suit :

Q1 : *De combien de véhicules le système est-il constitué ?*
=>**H1 : Un ensemble de N robots (N=3 dans ce travail), homogènes en termes de types et de technologies.**
Q2 : *Quels types de capteurs sont déployés ?*
=>**H2 : Les capteurs sont proprioceptifs et extéroceptifs, utilisés pour la localisation.**
Q3 : *Comment garantir l'application de la coopération dans le système ?*
=>**H3 : Chaque robot est capable d'observer au moins un autre robot et est lui-même observé par au moins un autre.**
Q4 : *Comment garantir la continuité de la solution de localisation de chaque véhicule ?*
=>**H4 : Au moins 2 capteurs doivent fonctionner pour garantir l'obtention d'une solution de localisation.**
Q5 : *Quelle architecture faut-il choisir pour augmenter la tolérance du système vis-à-vis des défauts ?*
=>**H5 : L'architecture est décentralisée, avec un traitement local.**
Q6 : *Comment est la communication maintenue ?*
=>**H6 : La communication entre les robots est maintenue en permanence (le WiFi est utilisé dans ce travail).**
Q7 : *A quoi sont les défauts capteurs assimilés ?*
=>**H7 : Les défauts des capteurs peuvent être assimilés à des mesures erronées, une confusion entre le robot et un autre objet, un biais, des mesures aberrantes et une absence d'information.**
Q8 : *Comment un défaut est-il défini ?*
=>**H8 : Les défauts injectés (ajoutés aux mesures capteurs) doivent être étiquetés en durée, temps d'occurrence, et sévérité.**
Q9 : *Quelle est l'abondance des défauts ?*
=>**H9 : Les défauts constituent des évènements rares, signifiant que leur durée est brève, entre 1% et 5% de la durée totale de l'essai.**
Q10 : *Quel est la sévérité du défaut ?*
=>**H10 : Les défauts sont d'une sévérité qui ne dépasse pas les 30% au-delà de l'incertitude du capteur.**

Nous prenons en considération ces hypothèses pour construire le système. Premièrement, nous présentons les protocoles de communication sous une architecture décentralisée. Puis nous modélisons le système en définissant le vecteur d'état utilisé, son évolution par le modèle odométrique, et ses corrections. Dès lors que le modèle est construit, nous présentons l'étape d'estimation de position avec les filtres gaussiens. Dans cette section, l'algorithme de la méthode est détaillé et expliqué. Ensuite, nous détaillons l'étape de diagnostic, qui comporte premièrement le calcul des résidus de détection et d'isolation. Par la suite, ayant discuté toutes les étapes, nous soulignons les exigences de la base de données capteurs à générer, en termes de scénarios, et de données. Enfin, nous présentons la méthode de détection

et isolation de défauts développée avec ses deux versions, classique et mixte.

3.3 Protocoles de communication et architecture

La coopération entre les véhicules est assurée grâce à la communication et la perception par le télémètre laser embarqué sur les véhicules. Afin d'augmenter la tolérance du système, on adopte l'architecture décentralisée de fusion de données. En adoptant cette architecture, chaque robot traite ses propres informations. Ce type de système ne nécessite pas d'installation centrale de fusion ou de communication, et il n'existe à aucun moment un lieu commun où la fusion ou les décisions globales sont prises. L'avantage est que chaque capteur/robot est entièrement autonome et peut fonctionner indépendamment de tout autre composant du système et, par conséquent, son erreur n'affecte que le robot dont il est équipé.

Il existe deux possibilités d'établir la coopération entre les véhicules :

1. Les véhicules observent les autres véhicules et leurs fournissent une correction de leur position tenant compte de l'observation relative et la position prédite du robot observant. Pour illustrer ce mode, pour un scénario de coopération sur une intersection avec trois véhicules, les informations communiquées sont présentées dans la Fig.3.2, dont la phase de communication est dressée dans la Fig.3.3.

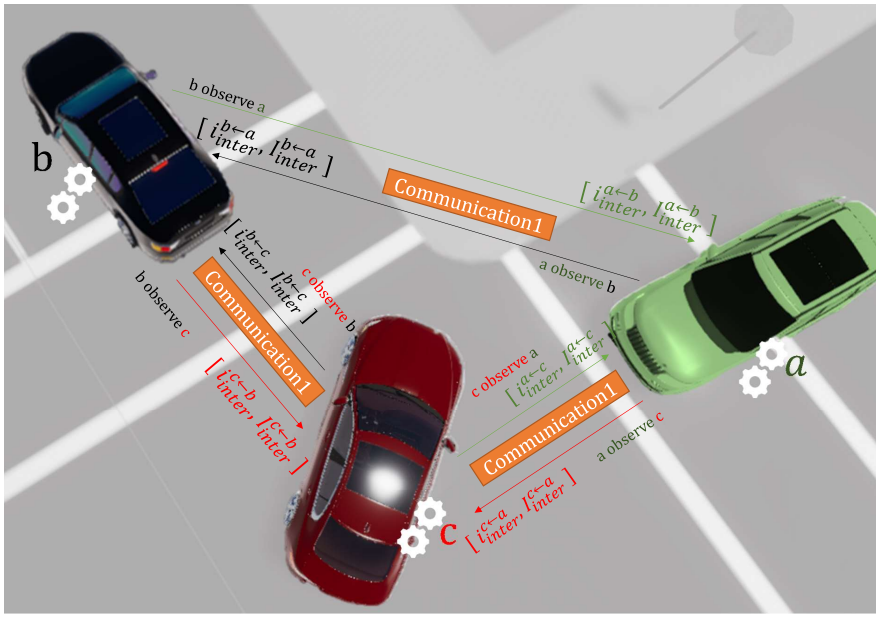


Figure 3.2 – Informations communiquées pour le cas de correction d'autres robots

Prenons par exemple le cas du véhicule R_a. R_a détecte la présence d'autres véhicules R_b et R_c en vue, et cherche à communiquer avec eux. Pour la communication avec R_b par exemple, R_a lui demande la

connexion. Une fois que R_b l'autorise, R_a qui observe R_b récupère l'information de son propre télémètre qui indique l'interdistance et l'angle de gisement vers R_b en vue. En utilisant cette information avec la prédiction de sa pose, R_a calcule le vecteur et la matrice contribution informationnelle de sa correction vers R_b. Puis, en suivant l'architecture décentralisée, R_a n'envoie que ces contributions vers R_b en vue. R_b reçoit cette correction et l'implémente dans la phase de fusion de données multi-capteurs pour calculer sa pose. Ensuite, R_b applique le diagnostic sur cette correction : si elle n'est pas cohérente avec la prédiction, il décide qu'il y a eu une détection de défaut et procède à la phase d'isolation afin de localiser le capteur défectueux. Si la source de défaut est la correction communiquée par R_a, R_b envoie à R_a un retour d'information en indiquant l'existence d'un défaut à bord de R_a. R_a reçoit cette information et la sauvegarde comme historique de communication avec le véhicule R_b.

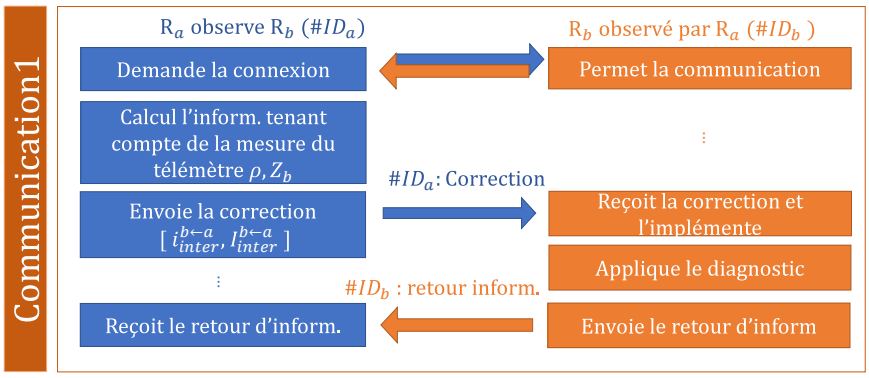


Figure 3.3 – Phase de communication pour le cas de correction d'autres robots

L'avantage de ce mode est qu'il permet d'une part en étant observé l'obtention d'une correction supplémentaire, et d'autre part en étant observateur un retour d'information sur ses corrections. De plus, en utilisant un véhicule équipé de capteurs de meilleure précision qui interagit avec d'autres véhicules, il est possible de réduire le coût global du système en permettant aux véhicules disposant de capteurs moins précis d'améliorer leur précision. D'autre part, en sollicitant l'avis d'un véhicule externe, cela permet au robot émettant cette correction d'évaluer la qualité de ses estimations, notamment dans les cas où son diagnostic a été établi sur une période prolongée et où des erreurs ont pu s'accumuler, ce que le véhicule seul ne serait pas en mesure de déterminer.

Cette méthode a cependant un inconvénient qui réside dans le fait que la contribution informationnelle transmise est issue de deux sources couplées provenant du même véhicule, et chacune de ces sources est susceptible de contenir des défauts. Plus précisément, il peut y avoir des défauts liés à la position du véhicule observant ainsi qu'aux mesures effectuées par le télémètre. Si, par exemple, le télémètre lui-même n'est

pas défectueux, mais que la correction calculée à partir de ses données est erronée, cela conduit à une diminution de l'efficacité de l'utilisation des composants disponibles dans le système.

2. Dans la seconde méthode, les véhicules diffusent l'estimation de la position sur le réseau partagé, que d'autres véhicules récupèrent pour les utiliser dans l'estimation de leur position, en les considérant comme des repères portables. L'information échangée sera la position existant sur le réseau et acquis par le véhicule qui se localise par rapport au second véhicule, puis ce véhicule diffuse l'estimation de la position sur le réseau. Cette méthode est visualisée dans la Fig3.4, et la phase de communication est détaillée dans la Fig.3.5.

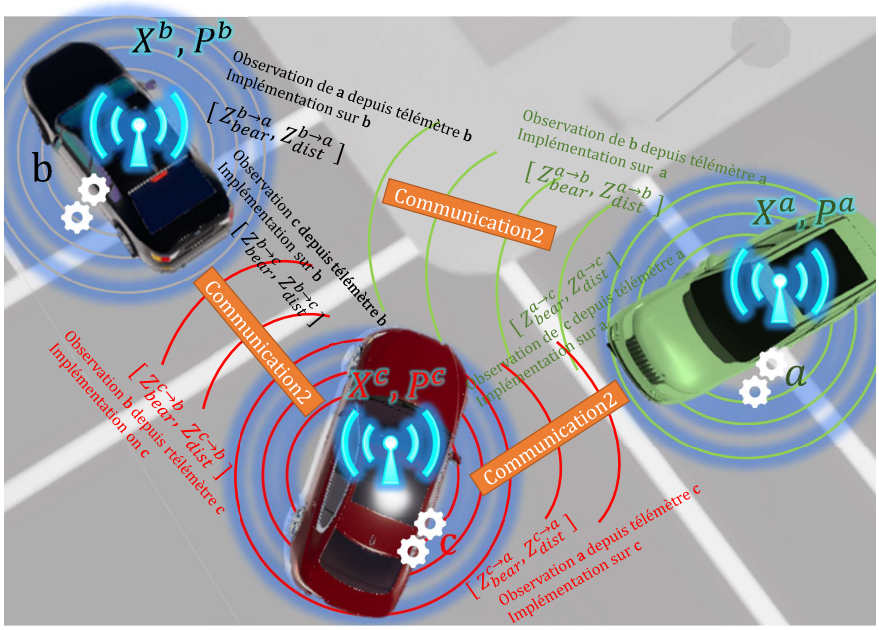


Figure 3.4 – Phase de communication pour le cas de repères portables

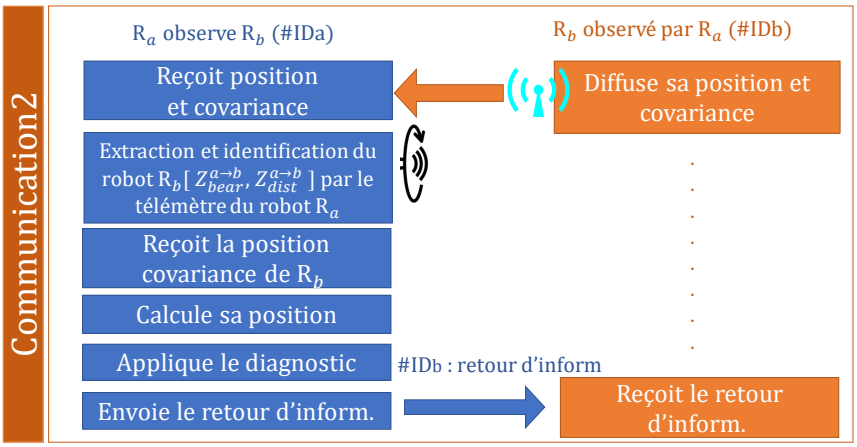


Figure 3.5 – Phase de communication pour le cas de repères portables

Partant du même scénario, prenons aussi l'exemple du véhicule R_a.

R_a détecte la présence d'autres véhicules en vue R_b et R_c, qu'il considère comme des points de repère mobiles. Après avoir prédit sa pose et sa covariance, R_a cherche à obtenir celles des autres véhicules, qui sont diffusés sur le réseau. Il observe premièrement R_b, et obtient sa pose X^b et sa covariance P^b. Ensuite, R_a obtient les mesures de données de son propre télémètre et calcule l'interdistance et l'angle de gisement vers R_b. En utilisant le pose et la covariance de R_b avec les informations du télémètre, R_a calcule la correction. R_a répète la même procédure avec le véhicule R_c, puis calcule l'autre correction qu'il intègre avec la première dans son algorithme de fusion de données multi-capteurs pour obtenir sa pose X^a et sa covariance P^a. R^a procède ensuite à la phase de diagnostic sur cette correction : si elle n'est pas cohérente avec la prédiction, il décide qu'il y a eu une détection de défaut et procède à la phase d'isolation afin de localiser le capteur défectueux. Si la phase d'isolation de défaut détecte un défaut sur la correction obtenue à partir du véhicule R_b, elle observe celle du véhicule R_c. Si les deux sont défectueux, ceci indique que l'information commune entre elles (provenant du télémètre laser) a induit l'erreur, le télémètre laser sera alors déclaré défectueux. D'autre part, si l'une des deux corrections induit le défaut, disons celui de R_b, ceci signifie que l'information du télémètre est correcte, et la valeur erronée est celle de la prédiction de R_b. Dans ce cas, le R_a envoie un retour d'informations à R_b en lui indiquant qu'un défaut a été détecté sur sa pose.

Cette seconde méthode offre les mêmes avantages que la première. Elle permet à un véhicule observé de partager sa position avec les autres véhicules, ce qui permet à ces derniers de la critiquer et de détecter d'éventuels défauts dans ces informations. De plus, cette approche permet d'observer d'autres véhicules et d'utiliser leur position pour se localiser. En outre, cette méthode présente également l'avantage d'améliorer la précision globale du système en intégrant au moins un véhicule doté de capteurs de meilleure précision que les autres.

La différence de cette méthode par rapport à la première est que les sources d'information sont de deux véhicules différents, et que leur dissociation est plus facile, surtout pour un nombre de véhicules qui est supérieur ou égal à trois.

Dans la suite, nous optons pour la deuxième méthode de communication, car nous utilisons la méthode des repères portables pour mettre en œuvre la coopération.

3.4 Modélisation du système

Le système de localisation coopérative étudié est composé de trois robots mobiles, où l'objectif est de trouver la pose de chacun. Pour le $Robot_a$, le vecteur d'état est composé de trois coordonnées, la position selon l'axe x, selon l'axe y

et la rotation autour de l'axe z, comme suit :

$$X^a = [x^a \quad y^a \quad \theta^a]^T \tag{3.1}$$

Pour valider l'approche proposée, l'évolution est déterminée à l'aide d'encodeurs montés sur les roues. Concernant les observations, et dans le cadre où les défauts détectés sont exclus, il est essentiel d'assurer une redondance d'information. Cette redondance est nécessaire pour implémenter efficacement la fusion de données issues de plusieurs capteurs, tout en minimisant le nombre de capteurs requis et en garantissant un degré minimal de redondance entre eux. En outre, dans le cadre de la coopération, une source de mesure relative est requise. Pour les coordonnées $[x, y]$, nous implémentons un système de positionnement interne absolu, et un capteur de mesure des interdistances. Quant à la coordonnée θ, elle est déterminée à l'aide d'un capteur spécifique pour mesurer l'orientation.

3.4.1 Modèle d'évolution

Afin de suivre l'évolution du système, un modèle cinématique basé sur les données des encodeurs des roues est déployé. Le modèle utilisé du véhicule est celui de *char*, comme illustré dans la Figure 3.6. Le vecteur d'entrée est noté $u_k^a = [\Delta_k^a \quad \omega_k^a]^T$, où Δ_k^a est le déplacement élémentaire du robot et ω_k^a est la rotation élémentaire effectuée. Ces grandeurs sont calculées à partir des ticks des encodeurs et les paramètres du robot déployé.

L'évolution des coordonnées est comme suit :

$$\begin{cases} x_{k+1}^a = x_k^a + \frac{D_r^a + D_l^a}{2} \times cos(\theta_k^a) = x_k^a + \Delta_k^a \times cos(\theta_k^a) \\ y_{k+1}^a = y_k^a + \frac{D_r^a + D_l^a}{2} \times sin(\theta_k^a) = y_k^a + \Delta_k^a \times sin(\theta_k^a) \\ \theta_{k+1}^a = \theta_k^a + \frac{D_r^a - D_l^a}{e} = \theta_k^a + \omega_k^a \end{cases} \tag{3.2}$$

avec D_r et D_l les distances parcourues par chaque roue, calculées à partir du nombre de ticks indiqué par l'encodeur entre deux itérations successives Δ_{ticks}, r le rayon de la roue et N le nombre de trous par révolution :

$$D = \frac{2\pi \times r}{N} \times \Delta_{ticks} \tag{3.3}$$

Par la suite, l'équation d'évolution est la suivante :

$$X_{k+1}^a = f(X_k^a, u_{k+1}^a) = I_{3\times 3} \times X_k^a + B_{k+1}^a \times u_{k+1}^a \tag{3.4}$$

où B_{k+1}^a la matrice d'entrée est définie comme :

$$B_k^a = \begin{bmatrix} cos(\theta_k^a + \frac{\omega_k^a}{2}) & 0 \\ sin(\theta_k^a + \frac{\omega_k^a}{2}) & 0 \\ 0 & 1 \end{bmatrix} \tag{3.5}$$

Le système est non-linéaire, ce qui aboutit à l'usage de la forme de Jordan en calculant les matrices jacobiennes $F_k^a = \frac{\partial f}{\partial X}|_{X=X_{k+1}^a}$ et $G_k^a = \frac{\partial f}{\partial u}|_{u=u_{k+1}^a}$

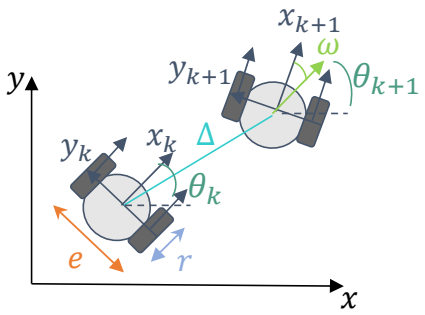


Figure 3.6 – Modèle odométrique

$$F_k^a = \begin{bmatrix} 1 & 0 & -\Delta_k^a \times sin(\theta_k^a + \frac{\omega_k^a}{2}) \\ 0 & 1 & \Delta_k^a \times cos(\theta_k^a + \frac{\omega_k^a}{2}) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$G_k^a = \begin{bmatrix} cos(\theta_k^a + \frac{\omega_k^a}{2}) & -\frac{1}{2}\Delta_k^a \times sin(\theta_k^a + \frac{\omega_k^a}{2}) \\ sin(\theta_k^a + \frac{\omega_k^a}{2}) & \frac{1}{2}\Delta_k^a \times cos(\theta_k^a + \frac{\omega_k^a}{2}) \\ 0 & 1 \end{bmatrix} \tag{3.7}$$

Concernant la matrice de variance covariance, elle est définie comme suit :

$$P_{k+1}^a = (F_k^a) \times P_k^a \times (F_k^a)^T + (G_k^a) \times Q_{u_k^a} \times (G_k^a)^T \tag{3.8}$$

avec $Q_{uk}^a = M \times M^T \times q_{ticks}$, où M est définie par :

$$M = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2e} & \frac{-r}{2e} \end{bmatrix} \tag{3.9}$$

et q_{ticks} est l'incertitude sur l'encodeur, composée du produit entre la résolution du capteur rapporté à la translation :

$$q_{ticks} = \text{Résolution}_{encodeur} \times \frac{r \times \pi}{180} \tag{3.10}$$

3.4.2 Modèles d'observation

En ce qui concerne les modèles d'observations, les mesures peuvent être exprimées dans le référentiel approprié dès le départ, ou elles peuvent nécessiter une transformation préalable. Dans la section suivante, nous examinerons deux types de capteurs, l'un nécessitant un prétraitement avec une transformation relative, tandis que l'autre nécessite simplement une transformation de repère.

3.4.2.1 Observations issues du système de localisation intérieure et du gyroscope

Les observations du système de positionnement interne et du gyroscope sont considérées absolues. Dans ce cas, une simple transformation de repère

est nécessaire pour rendre la mesure dans le bon référentiel. Pour des robots terrestres, le seul axe de rotation possible est autour de l'axe z. On considère la matrice de transformation T de la forme suivante :

$$T = \begin{bmatrix} Rot_{2\times 2}^{\phi_z} & | Translation_{2\times 1}^{(x,y)} \\ 0 \quad 0 & 1 \end{bmatrix}_{3\times 3} \tag{3.11}$$

Pour une observation $[x^m, y^m]^T$, de translation $[\delta_x^m, \delta_y^m]^T$ et de rotation ϕ_z^m autour de l'axe z, la matrice de transformation T devient :

$$T = \begin{bmatrix} cos(\phi_z^m) & sin(\phi_z^m) & \delta_x^m \\ -sin(\phi_z^m) & cos(\phi_z^m) & \delta_y^m \\ 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

Pour transformer ces coordonnées vers le référentiel d'origine, soit $[Z_x^m, Z_y^m]^T$, on applique la transformation suivante :

$$\begin{bmatrix} Z_x^m \\ Z_y^m \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\phi_z^m) & sin(\phi_z^m) & \delta_x^m \\ -sin(\phi_z^m) & cos(\phi_z^m) & \delta_y^m \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x^m \\ y^m \\ 1 \end{bmatrix} \tag{3.13}$$

En ce qui concerne l'angle du gyroscope noté θ^g, c'est un simple ajout de l'angle de rotation ϕ_z^g comme suit :

$$Z_\theta^g = \theta^g + \phi_z^g \tag{3.14}$$

Ces transformations n'affectent pas les valeurs des incertitudes des capteurs qui sont basées sur les erreurs de propagation des coordonnées présentées. Par la suite les matrices d'incertitudes sont les suivantes :

$$R^m = \begin{bmatrix} \sigma_{x^m,x^m} & 0 \\ 0 & \sigma_{y^m,y^m} \end{bmatrix}, \qquad R^g = \sigma_{\theta^g,\theta^g} \tag{3.15}$$

3.4.2.2 Observation issue du télémètre et de la prédiction de la pose d'un autre robot

Le vecteur d'observation $Z_k^{L^{a\to b}}$ est obtenu à partir des mesures du télémètre, composées de l'interdistance $Z_{dist}^{a\to b}$ et de l'angle de gisement $Z_{gis}^{a\to b}$ entre le robot observant R_a et le robot observé R_b. L'incertitude $R_k^{L^{a\to b}}$ de cette mesure est basée sur celle du capteur lui-même et de la position du robot observé partageant sa position.

$$R_k^{L^{a\to b}} = R_{Z^{L^{a\to b}}} + P_k^b \tag{3.16}$$

où $R_{Z^{L^{a\to b}}}$ contient les erreurs de propagation calculées comme suit :

$$R_{Z^{L^{a\to b}}}^{L^{a\to b}} = \begin{bmatrix} \sigma_{x,x} & \sigma_{x,y} \\ \sigma_{x,y} & \sigma_{y,y} \end{bmatrix} \tag{3.17}$$

avec :

$$\begin{cases} \sigma_{x,x} = \frac{\partial x^{L^{a\to b}}}{\partial Z_{dist}^{a\to b}} \times \sigma^2_{Z_{dist}^{a\to b}} + \frac{\partial x^{L^{a\to b}}}{\partial Z_{gis}^{a\to b}} \times \sigma^2_{Z_{gis}^{a\to b}} + \frac{\partial x^{L^{a\to b}}}{\partial \theta^a_{k|k-1}} \times \sigma^2_{\theta^a_{k|k-1}} \\ \sigma_{y,y} = \frac{\partial y^{L^{a\to b}}}{\partial Z_{dist}^{a\to b}} \times \sigma^2_{Z_{dist}^{a\to b}} + \frac{\partial y^{L^{a\to b}}}{\partial Z_{gis}^{a\to b}} \times \sigma^2_{Z_{gis}^{a\to b}} + \frac{\partial y^{L^{a\to b}}}{\partial \theta^a_{k|k-1}} \times \sigma^2_{\theta^a_{k|k-1}} \\ \sigma_{x,y} = \frac{1}{2} \times \left(\sigma_{x+y,x+y} - \sigma_{x,x} - \sigma_{y,y}\right) \\ \sigma_{x+y,x+y} = \left(\frac{\partial (x^{L^{a\to b}} + y^{L^{a\to b}})}{\partial Z_{dist}^{a\to b}}\right)^2 \times \sigma^2_{Z_{dist}^{a\to b}} + \left(\frac{\partial (x^{L^{a\to b}} + y^{L^{a\to b}})}{\partial Z_{gis}^{a\to b}}\right)^2 \times \sigma^2_{Z_{gis}^{a\to b}} + \left(\frac{\partial (x^{L^{a\to b}} + y^{L^{a\to b}})}{\partial \theta^a_{k|k-1}}\right)^2 \times \sigma^2_{\theta^a_{k|k-1}} \end{cases} \tag{3.18}$$

Le modèle appliqué est non-linéaire, et est défini par les équations suivantes :

$$\begin{bmatrix} Z_{x^{L^a}} \\ Z_{y^{L^a}} \end{bmatrix} = \begin{bmatrix} Z_{dist}^{a\to b} \times cos(Z_{gis}^{a\to b} + \theta^a_{k|k-1}) \\ Z_{dist}^{a\to b} \times sin(Z_{gis}^{a\to b} + \theta^a_{k|k-1}) \end{bmatrix} + \begin{bmatrix} x^b_{k|k-1} \\ y^b_{k|k-1} \end{bmatrix} \tag{3.19}$$

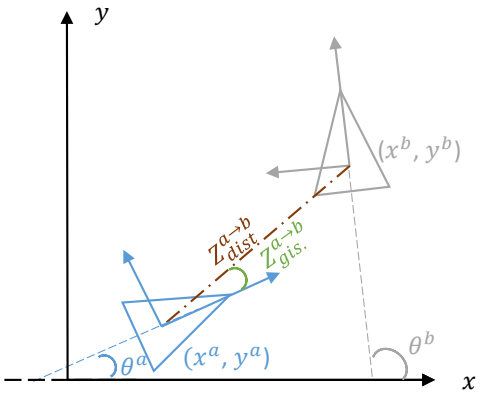


Figure 3.7 – Observation du télémètre entre $Robot_a$ et $Robot_b$

3.5 Estimation de la position et de l'orientation

Compte tenu de la position, l'usage est celui des filtres gaussiens, dont la prédiction est faite avec le modèle cinématique odométrique, et les corrections avec des capteurs proprioceptifs, extéroceptifs et relatifs. Deux types de filtres sont utilisés dans notre étude :

1. Batch Covariance Intersection Filter (B-CIF) : sous l'hypothèse de consanguinité de données, et bénéficiant de l'aptitude de ce filtre à éliminer l'effet d'écho. Le poids accordé à chaque correction est selon l'optimisation de la trace de la matrice de covariance.

$$(P^a_{k|k})^{-1} = (P^{a^{BCIF}}_{k|k})^{-1} = (\sum_{obs^a} \omega_{obs^a} (P^{obs^a}_k)^{-1})^{-1} \tag{3.20}$$

$$X^a_{k|k} = X^{a^{BCIF}}_{k|k} = P^{a^{BCIF}}_{k|k} \times (\sum_{obs^a} \omega_{obs^a} (P^{obs^a}_k)^{-1} \times X^{obs^a}_k) \tag{3.21}$$

2. Extended Informational Filter (EIF) : sous l'hypothèse du diagnostic qui n'emmène pas au cas de sur-convergence vers une solution incorrecte, vu que les corrections en défauts sont exclus du système.

$$Y^a_{k|k} = Y^a_{k|k-1} + \sum_{l}^{obs^a} (H^l_k)^T \times (R^l_k)^{-1} \times (H^l_k) \tag{3.22}$$

$$y^a_{k|k} = y^a_{k|k-1} + \sum_{l}^{obs^a} (H^l_k)^T \times (R^l_k)^{-1} \times Z^l_k \tag{3.23}$$

L'évolution de la méthode est présentée dans l'algorithme 1. Cet algorithme commence par définir les valeurs initiales et les dimensions du vecteur d'état et de la matrice de variance-covariance. Le vecteur d'état est composé des coordonnées $[x \quad y \quad \theta]^T$, de la position sur les axes x et y, et de l'orientation θ de l'axe z. Le repère utilisé pour la localisation est celui de la vérité terrain afin de pouvoir comparer l'estimation de la position produite par la méthode à la vérité terrain. Pour le $robot_a$ par exemple, le vecteur d'état est initialisé par la valeur $X^a_o \in \mathbb{R}^{3\times1}$, qui correspond à sa position dans la vérité terrain lors de l'initialisation de la méthode. Sa matrice de variance-covariance $P^a_o \in \mathbb{R}^{3\times3}$ correspond aussi à celle définie par la vérité terrain. De plus, la méthode s'intéresse au suivi de la probabilité de l'hypothèse de non-défaut P_0. La cause de cet intérêt est l'adaptation des seuils selon l'historique de fonctionnement du système. Cette valeur est calculée par le rapport entre le nombre d'itérations sans défauts C_0 sur le nombre total d'itérations avec et sans défauts $C_1 + C_0$. Son interprétation est la suivante : une valeur de $P_0 < 0.5$ signifie que durant son fonctionnement, le système a eu plus de cas de défaut que de sans défaut. L'implémentation de cette information dans la méthode peut agir sur la largeur de la marge acceptable des indicateurs de défauts. En effet, dans la méthode classique développée par Al Hage, 2016, plus le système possède de défauts, plus les niveaux des indicateurs de défauts peuvent être élevés même si aucun défaut n'existe actuellement sur le système. Par la suite, l'activation de résidu de détection de défauts reflète ceci, par une augmentation du seuil dans le cas de la méthode classique, et changement de la sortie du modèle d'apprentissage en méthode mixte. Cette grandeur nécessite elle aussi une initialisation, pour le comportement général du système ainsi que pour chaque capteur.

Après la phase d'initialisation, la boucle principale de l'algorithme est appliquée. Tout d'abord, l'étape de prédiction du filtre d'information est appliquée. Elle utilise le modèle odométrique basé sur les données de l'encodeur pour prédire la position du robot $X^a_{k|k-1}$ et sa matrice de covariance $P^a_{k|k-1}$ tenant compte du déplacement élémentaire et de la rotation effectués inclus dans le vecteur d'entrée $u^a = [\Delta^a, \omega^a]^T$, et de la modélisation de cette prédiction par les matrices B^a_k et F^a_k.

Après la prédiction, la projection dans l'espace d'information est appliquée, en utilisant (2.22). Cette étape constitue une préparation pour l'étape de correction. Si la correction calculée est celle d'un capteur intégré qui observe le robot lui-même, le vecteur et la matrice de contribution informationnelle sont calculés en tenant compte des paramètres du capteur, tels que l'incertitude

Algorithm 1 Algorithme de localisation coopérative tolérante aux fautes

Require: Initialisation de l'état avec X_o^a avec sa covariance P_o^a obtenues de la vérité terrain pour chaque robot.

Require: Fixer les valeurs initiales de P_0 et $P_0^{obs^a}$.

while $k \leq Nb_{iterations}$ **do**

* ***Appliquer l'étape de prédiction*** : (section 3.4.1)

Lire les données des encodeurs : $u^a \leftarrow [\Delta^a, \omega^a]^T$

Calculer $X^a_{k|k-1}$ en utilisant $X^a_{k-1|k-1}$, B^a_k avec l'équation (3.5)

Calculer $P^a_{k|k-1}$ en utilisant $P^a_{k-1|k-1}$, F^a_k avec l'équation (3.6).

* ***Appliquer l'étape de correction*** : (section 3.4.2)

Calculer la correction des capteurs *intra* $X_k^{obs^k}$ et $P_k^{obs^k}$ pour $obs^a \in \{m^a, g^a\}$ (section 3.4.2.1).

if $Robot_b$ ou $Robot_c$ en vue **then**

Obtenir leur position et leur covariance $\{X^{b,c}_{k|k-1}, P^{b,c}_{k|k-1}\}$, (méthode 2 section 3.3).

Obtenir l'observation relative $Z_k^{L^{a\rightarrow b,c}}$

Calculer la correction des capteurs *inter* $X_k^{obs^k}$ et $P_k^{obs^k}$ pour $obs^a \in \{L^{a\rightarrow b}, L^{a\rightarrow c}\}$ en utilisant les équations (3.16) (3.19). (section 3.4.2.2).

end if

Calculer $X^a_{k|k}$ et $P^a_{k|k}$ soit par EIF en utilisant les équations (3.22) (3.23), soit par B-CIF en utilisant les équations (3.20) (3.21) (section 3.5).

* ***Appliquer l'étape de Diagnostic*** : (section 3.6)

Calculer Rd^a_{JS} en utilisant l'équation (3.29) (section 3.6.1).

if $\{Rd^a_{JS}, P_0\}$ implique la présence d'une faute **then**

Calculer $Ri_{JS^{obs^a_{GOS}}}$ en utilisant l'équation (3.30) (section 3.6.2)

Obtenir les capteurs défectueux

Exclure les capteurs défectueux

Mettre à jour $X^a_{k|k}$ et $P^a_{k|k}$

Mettre à jour le compte des cas avec défaut : $C_1 = C_1 + 1$

Mettre à jour le compte des cas avec défaut pour les capteurs : $C_1^{obs^a} = C_1^{obs^a} + 1$

end if

if Aucun défaut n'est détecté **then**

Mettre à jour le compte des cas sans défaut : $C_0 = C_0 + 1$

Mettre à jour le compte des cas sans défaut pour les capteurs : $C_0^{obs^a} = C_0^{obs^a} + 1$

end if

Mettre à jour P_0 ($P_0 = C_0/(C_1 + C_0)$)

Mettre à jour $P_0^{obs^a}$

$k = k + 1$

end while

R^{obs^a} et la matrice d'observation H^{obs^a}, en appliquant la transformation présentée dans la section 3.4.2.1. Autrement, si la correction provient du LiDAR, en supposant que d'autres robots du système sont dans le champ de vision du LiDAR, la méthode des repères portables est appliquée tel que présenté dans la section 3.4.2.2. Ces robots (dans le champ de vision) diffusent leur position et leur matrice de covariance dans le réseau partagé entre les robots. Le robot qui observe les autres robots récupère cette position et, avec l'observation LiDAR indiquant la position relative de ce robot, il calcule la contribution informationnelle de cette observation. Lorsque toutes les contributions informationnelles disponibles sont calculées, le modèle de correction est appliqué soit par EIF (3.22) et (3.23), en additionnant ces valeurs avec le vecteur et la matrice d'information de manière appropriée, puis en les projetant dans l'espace de Kalman, soit par le B-CIF en pondérant les corrections avec les équations (3.20) et (3.21).

Dès lors que la correction est obtenue, le robot procède au diagnostic de l'estimation. Pour ce faire, il utilise la théorie de l'information et calcule la dissimilarité, par la divergence de Jensen-Shannon, entre la prédiction et la correction dans un premier temps afin de savoir s'il y a un défaut dans le système en général, ce qui constitue l'étape de détection d'éventuel(s) défaut(s). Quant à la phase d'isolation, deux schémas peuvent être utilisés : soit le schéma dédié DOS, où chaque observateur et par suite correction est généré par un capteur uniquement, soit le schéma généralisé, où chaque correction intègre toutes les observations des capteurs sauf une à la fois. La différence entre ces deux méthodes est dans la quantité d'informations. Avec la méthode généralisée, comme toutes les observations des capteurs sont intégrées sauf un à la fois, cela produit des indicateurs de valeurs proches les uns des autres. En utilisant ces indicateurs, si la paire $\{Rd_{JS^a},\ P_0\}$ indique la présence d'un défaut, donc la phase d'isolation est activée, en calculant les résidus d'isolation et en localisant les capteurs défectueux qui seront exclus de la solution finale de fusion de données produisant l'estimation de la position dans l'itération actuelle. Enfin, une mise à jour des valeurs de P_0 aura lieu selon le résultat de l'itération courante. Dans la suite, nous présentons le type de résidu choisi, pour la détection et l'isolation, et la façon avec laquelle la détermination de l'occurrence de défaut est réalisée.

3.6 Étape de diagnostic

Dans le cadre de cette étude, nous avons choisi d'utiliser des indicateurs basés sur l'évaluation des résidus pour la détection de défauts. Ces résidus reflètent la disparité entre l'approximation (la correction) obtenue et la référence (la prédiction). Une particularité de notre approche est l'utilisation de la divergence de Jensen-Shannon, qui est dérivée de la divergence α-Rényi avec α tendant vers 1. Cette stratégie implique un niveau élevé de confiance accordé à nos prédictions, dont les paramètres ont été minutieusement calibrés en utilisant des données de terrain, garantissant ainsi une précision exceptionnelle pour les robots déployés. Cette haute précision est d'autant plus cruciale

compte tenu de la résolution fine dont disposent ces robots.

Le processus de détection de défauts que nous avons élaboré repose sur une structure en deux étapes bien distinctes. Dans un premier temps, nous avons mis en place une phase dédiée à la détection initiale des défauts. Une fois cette détection initiale validée, elle déclenche automatiquement la phase subséquente d'isolation des défauts. Cette approche en deux étapes a été conçue dans le but d'optimiser les calculs en évitant de les exécuter à chaque itération, ce qui se traduit par une réduction significative de la charge de calcul, contribuant ainsi à l'efficacité globale du processus.

3.6.1 Calcul du résidu de détection

La divergence de Jensen Shannon D_{JS} entre deux distributions p et g est définie par :

$$D_{JS}(p(x)||g(x)) = h(M) - \frac{1}{2}\Big(h(p(x)) - h(g(x))\Big) \tag{3.24}$$

avec $h(p)$ l'entropie de Shannon définie par :

$$h(p) = -\sum_j p_j log(p_j) \tag{3.25}$$

et M est la moyenne des deux distributions p et g :

$$M = \frac{1}{2}(p(x) + g(x)) \tag{3.26}$$

Cette divergence peut être obtenue depuis celle de Kullback-Leibler (KL) :

$$D_{JS}(p(x)||g(x)) = \frac{1}{2}D_{KL}(p(x), M) + \frac{1}{2}D_{KL}(g(x), M) \tag{3.27}$$

Cette représentation permet de mieux comprendre la représentation de Jensen Shannon, puisque la divergence de Kullback-Leibler a été largement étudiée et appliquée dans le cas d'une distribution gaussienne.

$$\begin{cases} D_{JS}(p(x)||q(x)) = h\left(\frac{1}{2}p(x) + \frac{1}{2}q(x)\right) - \frac{1}{2}h(p(x)) - \frac{1}{2}H(q(x)) \\ \quad = -\sum\left(\frac{1}{2}(p(x) + q(x)) \times log(\frac{1}{2}(p(x) + q(x)))\right) \\ \quad\quad +\frac{1}{2}\sum p(x) \times log(p(x)) + \frac{1}{2}\sum q(x) \times log(q(x)) \\ \quad = \sum\left(\frac{1}{2}p(x) \times log\left(\frac{p(x)}{\frac{1}{2}(p(x)+q(x))}\right)\right) + \sum\left(\frac{1}{2}q(x) \times log\left(\frac{q(x)}{\frac{1}{2}(p(x)+q(x))}\right)\right) \\ \quad = \frac{1}{2}D_{KL}\left(p(x)||\frac{1}{2}(p(x) + q(x))\right)\right) + \frac{1}{2}D_{KL}\left(q(x)||\frac{1}{2}(p(x) + q(x))\right) \end{cases} \tag{3.28}$$

Dans notre cas, chaque robot a applique son propre diagnostic localement, entre sa prédiction et la somme finale des contributions informationnelles données dans l'équation (3.20) ou (3.22) dans l'étape de détection des défauts. On considère donc que la prédiction suit une loi normale $\mathcal{N}(X^a_{k|k-1}, P^a_{k|k-1})$ et que la correction finale suit à son tour une loi normale $\mathcal{N}(X^a_{k|k}, P^a_{k|k})$, qui seront comparées par la divergence globale de Jensen-Shannon, obtenue de la manière suivante :

$$Rd^a_{JS} = D_{JS}(\mathcal{N}(X^a_{k|k-1}, P^a_{k|k-1})||\mathcal{N}(X^a_{k|k}, P^a_{k|k})) \tag{3.29}$$

3.6.2 Calcul des résidus d'isolation

Après avoir détectée la présence d'une mesure défectueuse dans le robot, à l'aide du gD^a_{JS}, l'élément défectueux doit être isolé (c'est-à-dire que nous devons déterminer quelle mesure est défectueuse) et exclu de la procédure de fusion. Les résidus d'isolation créés doivent être conçus de manière à être sensibles à tout type de défaut dans le système. Ayant un vecteur d'état $X \in \mathbb{R}^{3\times 1}$, et 3 types d'observations, on a besoin de 3 résidus au moins pour que le système soit observable.

Afin d'avoir une quantité d'information équilibrée entre les résidus, le schéma avec lequel ils sont conçus est le schéma d'observation généralisé (GOS). Dans ce schéma, les résidus sont obtenus par le calcul de la divergence entre la prédiction et la correction qui combine toutes les observations sauf une à la fois, puis deux jusqu'à ce que nous ayons au moins deux capteurs qui fonctionnent.

Pour chaque robot ayant L observations, $(1 + \sum_{j=1}^{L-2} C_L^{L-j})$ indicateurs sont générés, un de détection, mettant en œuvre toutes les observations dans l'étape de correction. Le C_L^{L-j} implique les résidus d'isolement, dans le cadre du GOS. Dans notre cas, où nous avons $L = 2 + (3 - 1) = 4$ observations, $(1 + 6 + 4)$ résidus $= 11$ types de résidus sont générés pour chaque robot. Le résidu de détection déjà présenté (3.29), et 10 résidus pour l'isolation $JS^{obs^a_{GOS}}$ calculés par l'équation suivante :

$$Ri_{JS^{obs^a}} = D_{JS}(\mathcal{N}(X^a_{k|k-1}, P^a_{k|k-1})||\mathcal{N}(X^{obs^a}_{k|k}, P^{obs^a}_{k|k})) \tag{3.30}$$

Ces résidus suivent aussi la divergence de Jensen-Shannon sous le schéma de l'observateur généralisé (GOS), avec $obs^a \in \{m^a, g^a, L^{a\leftarrow b}, L^{a\leftarrow c}, [m^a, g^a], [m^a, L^{a\leftarrow b}], [m^a, L^{a\leftarrow c}], [g^a, L^{a\leftarrow b}], [g^a, L^{a\leftarrow c}], [L^{a\leftarrow b}, L^{a\leftarrow c}]\}$. La matrice de signature est ainsi présentée dans le Tableau3.1.

Table 3.1 – Matrice de signature du Robot "a"

Résidu	$Ri_{JS^{m^a}}$	$Ri_{JS^{g^a}}$	$Ri_{JS^{L^{a\to b}}}$	$Ri_{JS^{L^{a\to c}}}$	...
Marvel a	0	1	1	1	...
Gyro a	1	0	1	1	...
Lidar a	1	1	0	0	...
Pose R_b	1	1	0	1	...
Pose R_c	1	1	1	0	...
...	...	...	...	...	...

Cette matrice de signature se caractérise par le fait qu'elle peut être étendue ou bien contractée (d'où l'usage de trois points), et qu'elle est capable de détecter les défauts multiples simultanés. Dans le cas de la localisation sans la perception d'autres robots, les résidus $Ri_{JS^{m^a}}$ et $Ri_{JS^{g^a}}$ sont suffisants pour déterminer l'origine d'un défaut. En revanche, lorsque plusieurs robots coopèrent

dans le système, les défauts provenant du LiDAR peuvent être localisés grâce à l'extension de la matrice de signature, tandis que ceux provenant des positions des autres robots peuvent être détectés à partir de n $\geq$ 3 robots. Par exemple, pour localiser un défaut sur le robot $Robot_b$, on utilise les résidus $Ri_{JS^{m^a}}$, $Ri_{JS^{g^a}}$ et $Ri_{JS^{L^{a\to c}}}$. Plus il y a d'autres robots dans le système, plus cette localisation peut être vérifiée et validée, donnant ainsi naissance au concept de diagnostic collectif.

3.6.3 Exigences des bases de données capteurs

L'objectif initial est de détecter la survenue de défauts sur les capteurs afin d'améliorer la robustesse de la solution de localisation. Pour ce faire, il est nécessaire de créer une base de données de scénarios de défaillance des capteurs. Cette démarche exige certaines spécifications lors de la création de cette base de données :

1. En termes de données brutes : Les données brutes doivent être échantillonnées à une fréquence constante, être complètes, couvrir l'intégralité du domaine d'observation et présenter une cohérence temporelle. Cela garantit que les données de la base de données sont fiables et appropriées pour l'analyse des défauts.
2. En termes de données générés : Les défauts doivent être étiquetés en termes d'occurrence générale et par capteur spécifique. La durée des défauts doit être entre 1% et 5% de la durée totale, leur sévérité ne dépasse pas 30%, et la base de données doit inclure des cas de défauts individuels ainsi que des défauts simultanés. De plus, lors de l'acquisition des données, des défauts peuvent apparaître, ce qui nécessite une phase de pré-traitement pour les étiqueter correctement.
3. En terme de trajectoire : Dans le contexte des trajectoires, il est nécessaire que celles-ci présentent une diversité significative et couvrent plusieurs types de mouvements, en incluant différents scénarios de détection entre les véhicules. En observant les comportements des véhicules, nous identifions plusieurs modes de coopération, qui sont montrés dans la Fig.3.8, où la coopération réside dans :
   — Les carrefours : L'intérêt de la coopération se manifeste lorsque des véhicules s'approchent d'un carrefour, car ils doivent collaborer avec les véhicules déjà présents dans le carrefour. Cette coopération vise à garantir une meilleur précision de la solution de localisation, afin que l'algorithme de planification de trajectoire puisse éviter les collisions.
   — Les intersections : Ce sont des zones où les mouvements des véhicules sont complexes et potentiellement dangereux. La coopération s'avère bénéfique entre les véhicules qui traversent l'intersection afin que les véhicules venant en sens inverse sachent exactement la position des autres, surtout au cas où un virage dans la même direction que le véhicule en train de traverser l'intersection est planifié. En sachant ces positions, l'algorithme de planification de trajectoire saura s'adapter pour éviter la collision entre les deux véhicules.

— Les zones de fusion (merging) : Les zones de fusion sont des points où plusieurs voies de circulation convergent, créant une situation complexe pour les véhicules en cours de fusion. La coopération dans ce cas est bénéfique entre les véhicules qui s'engagent dans la zone de fusion, de manière à ce que les véhicules provenant d'autres voies sachent précisément leur position. Cela revêt une importance cruciale, surtout lorsque des manœuvres de fusion dans la même direction sont envisagées. En connaissant avec précision les positions des véhicules dans la zone de fusion, l'algorithme de planification de trajectoire peut s'adapter pour éviter les collisions.
— Les dépassements sur les autoroutes : La coopération se révèle bénéfique dans ce cas entre les véhicules impliqués dans le dépassement, ainsi que ceux qui sont dépassés. Lorsqu'un véhicule souhaite dépasser un autre, l'algorithme de planification de véhicule qui fait le dépassement prend en considération la position des autres véhicules grâce à la localisation coopérative. De même, le véhicule qui est sur le point d'être dépassé peut ainsi connaître la position précise du véhicule qui le dépasse afin d'éviter les collisions.
— Les virages : Ce sont des points critiques sur la route, où la visibilité peut être limitée, et où les mouvements des véhicules peuvent être moins prévisibles. Afin de clarifier l'intérêt de la coopération dans ce scénario, prenons le cas où un véhicule prévoit de tourner à gauche dans un virage. Ce véhicule coopère avec les véhicules venant en sens inverse qui l'observe, et cherche à affiner sa solution de localisation afin d'éviter de faire une collision avec eux, et vice versa.

Ces trajectoires peuvent être regroupées en deux catégories initiales : les trajectoires de type sinusoïdal/circulaire et les trajectoires de type transversal. Afin d'obtenir une base de données diversifiée, il est nécessaire de générer plusieurs exemples de chaque type de trajectoire.

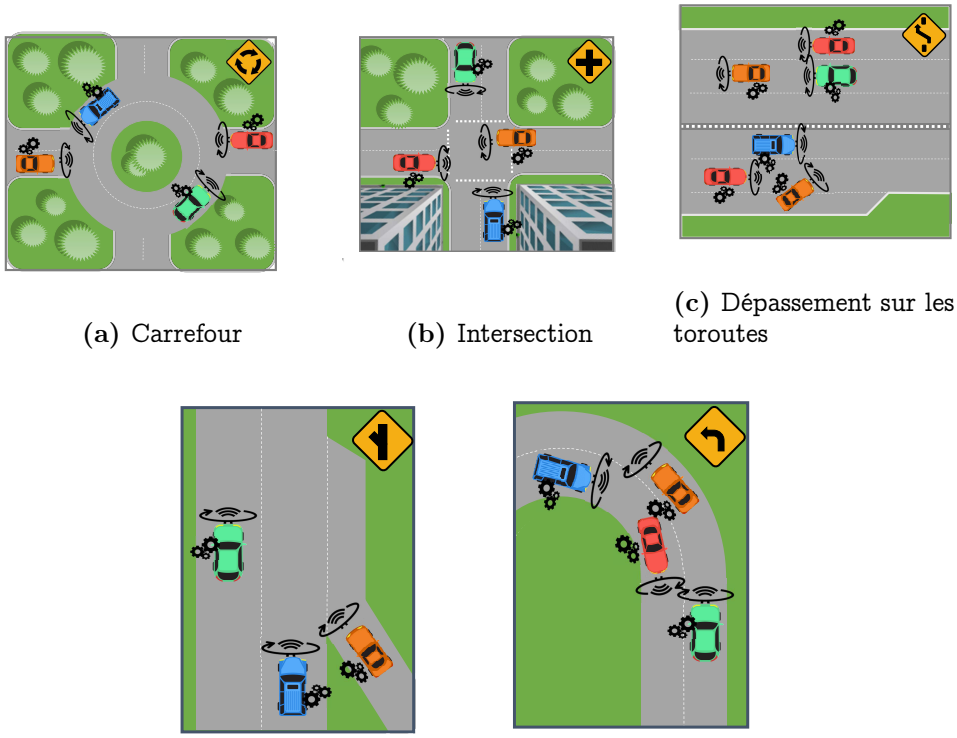
Dans la suite, nous détaillons la solution proposée pour la détection de ces défauts, qui joue un rôle crucial dans l'amélioration de la précision de la localisation.

3.7 Détection et isolation de défauts capteurs

Dans les parties précédentes, nous avons présenté chaque étape de l'approche proposée à part. Partant de l'architecture du système et sa communication, vers sa modélisation et la façon avec laquelle nous estimons la position du véhicule. Ensuite, nous avons détaillé l'étape de diagnostic et ses deux phases de détection et d'isolation.

Nous arrivons dans cette section à mettre toutes ces étapes dans un seul cadre afin de visualiser le fonctionnement du système depuis l'acquisition jusqu'à l'obtention de la solution de localisation tolérante aux fautes.

Dans le cadre de cette thèse, deux paradigmes différents sont développés. Le premier se base sur une méthode complètement basée sur le modèle, et



(a) Carrefour (b) Intersection (c) Dépassement sur les autoroutes

(d) Zones de fusion (e) Virage

Figure 3.8 – Scénarios routiers de coopération de véhicules

l'autre classifie les hypothèses de fonctionnement du système selon des décisions guidées par les données et des algorithmes d'apprentissage machine.

3.7.1 Méthode classique

La première mise en œuvre de la méthode qu'on nomme la méthode classique est similaire à ce qui a été fait au sein de l'équipe. Cette méthode se base complètement sur les modèles. La phase d'isolation de défaut se fait grâce à une table de signature qui indique le comportement des indicateurs de défauts selon le capteur qui provoque ce défaut. Cette décision est tirée grâce au seuillage des résidus. Le calcul du seuil se fait en utilisant des critères de la théorie d'information, qui cherche à minimiser le coût moyen associé à l'attribution d'une décision à une hypothèse. Cela vise à réduire à la fois la probabilité de fausse alarme et la probabilité de détection manquée.

Dans cette application, nous choisissons l'approche de la courbe ROC (Receiver Operating Curve) parmi les critères présentés dans la section 2.4.3.6. En temps que première implémentation, la méthode de communication utilisée est la première méthode de la section 3.3, où les robots obtiennent des corrections de leurs positions à partir d'autres robots qui les observent. Cette méthode est illustrée dans la Fig.3.9.

Pour décrire cette méthode, nous prenons le cas du $Robot_a$ à l'itération k.

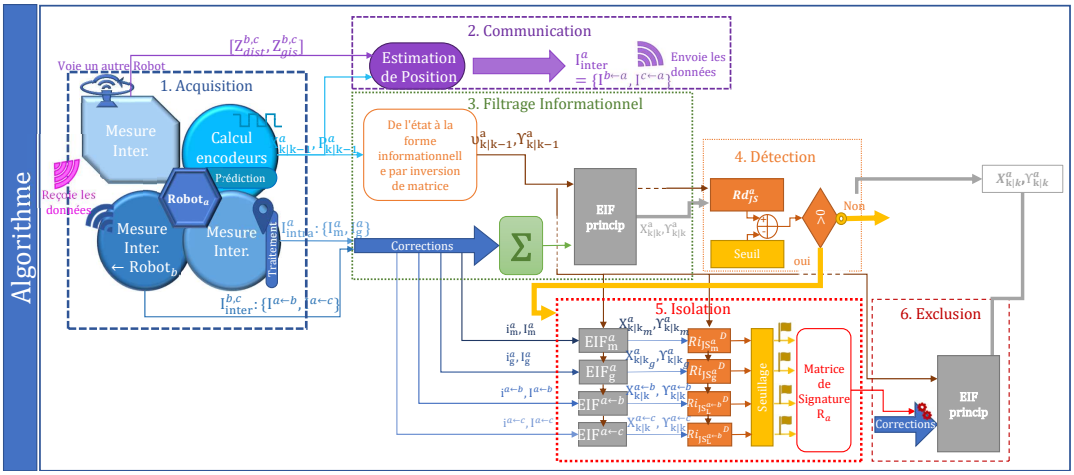


Figure 3.9 – Méthode classique basée sur les modèles uniquement

Cette méthode est divisée en 6 phases principales :

1. Acquisition de données : Dans cette étape, les données des capteurs se divisent en deux catégories distinctes : certains capteurs mesurent directement l'état du robot (par exemple, les systèmes GNSS, les gyromètres), d'autres capteurs (parfois combinés) permettent d'estimer la pose du robot relativement à un autre robot dans le champ de vision.
   Premièrement, les données acquises par des capteurs embarqués qui mesurent l'état du système lui-même. Ces capteurs sont directement intégrés dans le robot et recueillent des informations sur son propre état ou son environnement immédiat. Ils incluent les données des encodeurs, et celles du système de localisation intérieure (*Marvelmind*) et du gyroscope. Les données des encodeurs sont utilisés pour prédire l'état $X^a_{k|k-1}$ et la covariance $P^a_{k|k-1}$ en utilisant le modèle d'évolution de la section 3.4.1. Les données des deux autres capteurs sont utilisées pour l'étape de mise à jour de l'état, en utilisant les modélisations présentées dans la section 3.4.2.1.
   Deuxièmement, les données relatives échangées ou perçues entre robots, qui proviennent soit d'autres robots qui observent le robot actuel, soit sont destinées à être communiquées à d'autres robots qui les observeront à leur tour. Ces données sont conçues dans le cadre de la localisation coopérative, augmentant la disponibilité et l'utilisation du système en permettant aux robots d'obtenir des corrections depuis d'autres véhicules qui partagent le même environnement de navigation. Cette correction est obtenue grâce à la première méthode illustrée dans la section 3.3.

2. Communication : En observant son environnement, le $Robot_a$ détecte grâce à son LiDAR $Robot_b$ et $Robot_c$, et après identification récupère les interdistances $Z^{b,c}_{dist}$ et les angles de gisement $Z^{b,c}_{gis}$ vers ces deux robots en vue. Il calcule ensuite les contributions informationnelles issues de ces deux robots en utilisant les données relatives comme le montre la section

3.4.2.2, mais en utilisant sa prédiction, pour prédire la pose des deux autres robots, puis envoie ces contributions informationnelles suivant la première méthode de communication illustrée dans la section 3.3.

3. Filtrage informationnel : La fusion de données multi-capteur est réalisée dans cette méthode par l'usage du filtre informationnel. La première phase de cette étape est la projection de la prédiction de l'espace de Kalman vers l'espace informationnel par inversion de la matrice de covariance $P^a_{k|k-1}$ vers la matrice d'information $Y^a_{k|k-1}$, et de l'état $X^a_{k|k-1}$ vers le vecteur d'information $\nu^a_{k|k-1}$. La deuxième phase est le calcul des contributions informationnelles $\{i^{obs^a}, I^{obs^a}\}$ des corrections des capteurs du robot et des télémètres d'autres robots communiqués vers le $Robot_a$ que nous étudions. Dans ce cas, obs^a désigne les observations, et pour un système de trois robots, $obs^a \in \{m^a, g^a, L^{a\leftarrow b}, L^{a\leftarrow c}\}$, où la notion $L^{a\leftarrow b}$ désigne une correction calculée par le $Robot_b$, qui est communiquée au $Robot_a$ où elle sera implémentée. Le modèle de correction, comme indiqué dans la section 3.5 se fait par la sommation des contributions informationnelles avec la prédiction dans l'espace informationnel, produisant les corrections $\{\nu^a_{k|k}, Y^a_{k|k}\}$, qui sont ensuite projetées vers l'espace de Kalman pour obtenir $\{X^a_{k|k}, P^a_{k|k}\}$.
4. Détection de défauts : Une fois la correction obtenue, une vérification de sa cohérence est nécessaire. Pour cela, et tenant compte de la nature stochastique des données, les divergences sont utilisées afin de quantifier la dissimilarité entre la distribution de probabilité de la prédiction et celle de la correction. Dans cette étude, la divergence de Jensen-Shannon est utilisée pour calculer le résidu de détection Rd^a_{JS}, comme le montre la section 3.6.1. La décision de la présence (ou absence) d'un défaut est prise en comparant la valeur obtenue du résidu de détection à un seuil calculé grâce à la théorie de l'information.

   Afin de calculer ce seuil, la fonction de densité de probabilités (PDF) du résidu de détection Rd_{JS} sous l'hypothèse H_i est considérée comme étant $f(Rd_{JS}/H_i) : i = 0, 1$, où l'indication 0 désigne l'absence d'un défaut, et 1 indique sa présence. Ainsi, la formulation du problème de détection sera :

$$\begin{cases} H_0 : Rd_{JS} \sim f(Rd_{JS}/H_0) \\ H_1 : Rd_{JS} \sim f(Rd_{JS}/H_1) \end{cases} \tag{3.31}$$

   Et les probabilités de fausse alarme et de bonne détection s'obtiennent par :

$$\begin{cases} P_F = \int_{seuil}^{\infty} f(Rd_{JS}/H_0) dRd_{JS} \\ P_D = \int_{seuil}^{\infty} f(Rd_{JS}/H_1) dRd_{JS} \end{cases} \tag{3.32}$$

   Dans cette méthode, la courbe ROC est utilisée, où la valeur du seuil est obtenue par l'indice de Youden, indiquant le point de la courbe ROC le plus loin verticalement de la ligne de chance, comme présenté dans (2.72). Si la valeur du résidu Rd^a_{JS} dépasse ce seuil constant, un défaut est détecté.

5. Isolation de défaut : La détection de la présence d'un défaut fait appel à la phase d'isolation de ce défaut. Dans cette phase, des observateurs à entrée inconnue sont utilisés, avec le schéma dédié DOS. Avec ce schéma, les contributions informationnelles de chaque capteur sont utilisées à part pour calculer la correction, en n'intégrant qu'un seul à chaque fois, en suivant le schéma DOS d'observateurs. Ainsi, nous obtenons $2+(N-1) = 2+(3-1) = 4$ résidus d'isolation $Ri_{JS^{obs^a}}$ sur chaque robot du système comportant ici $N = 3$ robots. De la même façon que la détection, les seuils de chaque résidu sont calculés graâce à l'indice de Youden. Le vecteur de signature généré est par la suite comparé à la matrice de signature DOS présenté dans le tableau 3.2.

Table 3.2 – Matrice de signature DOS du $Robot_a$

Résidu	$Ri_{JS^{m^a}}$	$Ri_{JS^{g^a}}$	$Ri_{JS^{L^{a\leftarrow b}}}$	$Ri_{JS^{L^{a\leftarrow c}}}$	...	$Ri_{JS^{L^{b\leftarrow a}}}$	$Ri_{JS^{L^{c\leftarrow a}}}$
Marvel a	1	0	0	0	...	-	-
Gyro a	0	1	0	0	...	-	-
Pos_b	0	0	1	0	...	-	-
Pos_c	0	0	0	1	...	-	-
Lidar a	0	0	1	1	...	-	-
Pos_a	-	-	-	-	...	1	1

Prenons par exemple le cas de défaut sur le gyroscope du $Robot_a$. Ce défaut est détecté par constatation d'un dépassement du résidu d'isolation $Ri_{JS^{g^a}}$ par rapport à son seuil, et d'un non-dépassement des autres résidus d'isolation par rapport à leurs seuils respectifs. Cette table de signature se caractérise par son aptitude à détecter des défauts multiples simultanés. En effet, un défaut simultané sur le gyroscope et le système de localisation intérieure ($Marvelmind$) génère un vecteur de signature $[1, 1, 0, 0]$, où un dépassement simultané des deux résidus $Ri_{JS^{g^a}}$ et $Ri_{JS^{m^a}}$ de leurs seuils respectifs est observé. Par analogie, nous pouvons détecter la présence d'un défaut sur l'un des LiDARs selon leurs résidus respectifs. Ce type de défaut est communiqué comme retour d'information sur sa correction au robot qui a transmis sa prédiction. De la même façon, l'occurrence d'un défaut sur le LiDAR du $Robot_a$ est repéré grâce au retour d'information des $Robot_b$ et $Robot_c$, qui ont eu un dépassement de leur résidus $Ri_{JS^{L^{b\leftarrow a}}}$ et $Ri_{JS^{L^{c\leftarrow a}}}$ par rapport leurs seuils respectifs.

6. Exclusion de défaut : Dès lors que les capteurs défectueux sont localisés, une autre fusion de données multi-capteurs est appliquée, excluant ces capteurs défectueux pour fournir une correction finale de l'itération courante qui est plus précise.

Cette méthode est ainsi capable de détecter les défauts injectés et d'améliorer la solution de localisation en excluant le(s) capteur(s) produisant les défauts.

Par contre, une limitation identifié des méthodes basées sur les critères (pour le seuillage) est qu'elles ne sont adaptables que par la mise à jour de la

probabilité d'absence de défaut sur la base de l'historique du robot au cours de la navigation. De plus, elles nécessitent la fixation d'une valeur préalable pour l'initialisation, qui peut être loin du cas réel du robot. En effet, le module de l'erreur peut varier en fonction de la situation, et nous ne pouvons pas toujours présumer avoir un cas non défectueux pour la trajectoire sur laquelle nous basons notre étude pour le calcul des probabilités de fausse alarme et de détection, car l'apparition d'erreurs réelles ne peut pas être évitée pendant l'acquisition. C'est pourquoi l'utilisation de l'intelligence artificielle peut être un atout important dans ce domaine pour reconnaître le comportement des erreurs en apprenant de multiples cas d'erreurs avec différents modules pour détecter un comportement anormal du système.

3.7.2 Méthode mixte

En situation de défaillance, les performances du système divergent de son comportement attendu, ce qui complique la prédiction précise de sa localisation et la classification de son état de fonctionnement actuel au fil du temps.

Pour surmonter ce défi, les techniques d'apprentissage automatique offrent une solution prometteuse en termes de déduction de décisions. Pour effectuer une classification entre les différentes hypothèses de fonctionnement, il est nécessaire de déployer un modèle efficace capable de discriminer entre les scénarios normaux et défectueux. Cela relève des capacités des modèles discriminatifs qui se concentrent sur la modélisation explicite des critères de distinction entre les différentes classes et visent à classer de nouvelles instances en fonction de ces lois déduites. Par analogie avec la méthode classique présentée dans la section précédente, ceci est assimilé à la phase de seuillage pour la détection et l'isolation de défauts. Cette phase cherche à localiser le capteur qui est à l'origine du défaut observé, à partir du comportement d'indicateurs de défaillances conçu selon une table de signature de défauts.

Dans l'approche que nous proposons dans cette section, le mode "mixte" signifie que la solution de localisation coopérative tolérante aux fautes proposée n'est pas entièrement basée sur les modèles, ni uniquement guidée par les données, mais plutôt créée par la combinaison des deux approches. Comme nous l'avons présenté dans la section 2.4.5, l'intégration des techniques basées sur des modèles permet de fournir un contexte et une connaissance du domaine à celles guidées par les données. C'est pourquoi, nous gardons les parties d'estimation d'état, avec la création des indicateurs de défauts à base de divergence répondant à la nature stochastique des données. Le changement se fait dans les étapes de détection et isolation de défauts, où un modèle discriminatif est utilisé pour chacune de ces deux étapes. Le schéma général englobant les entrées-sorties est présenté dans la Fig.3.10, avec une structure commune pour les trois types d'apprentissage développés.

Cette méthode comporte aussi 6 étapes, et elle est très similaire à la méthode classique détaillée dans la section précédente. Le changement par contre est dans :

— Ajout de la possibilité d'utiliser le filtre d'intersection de covariance par

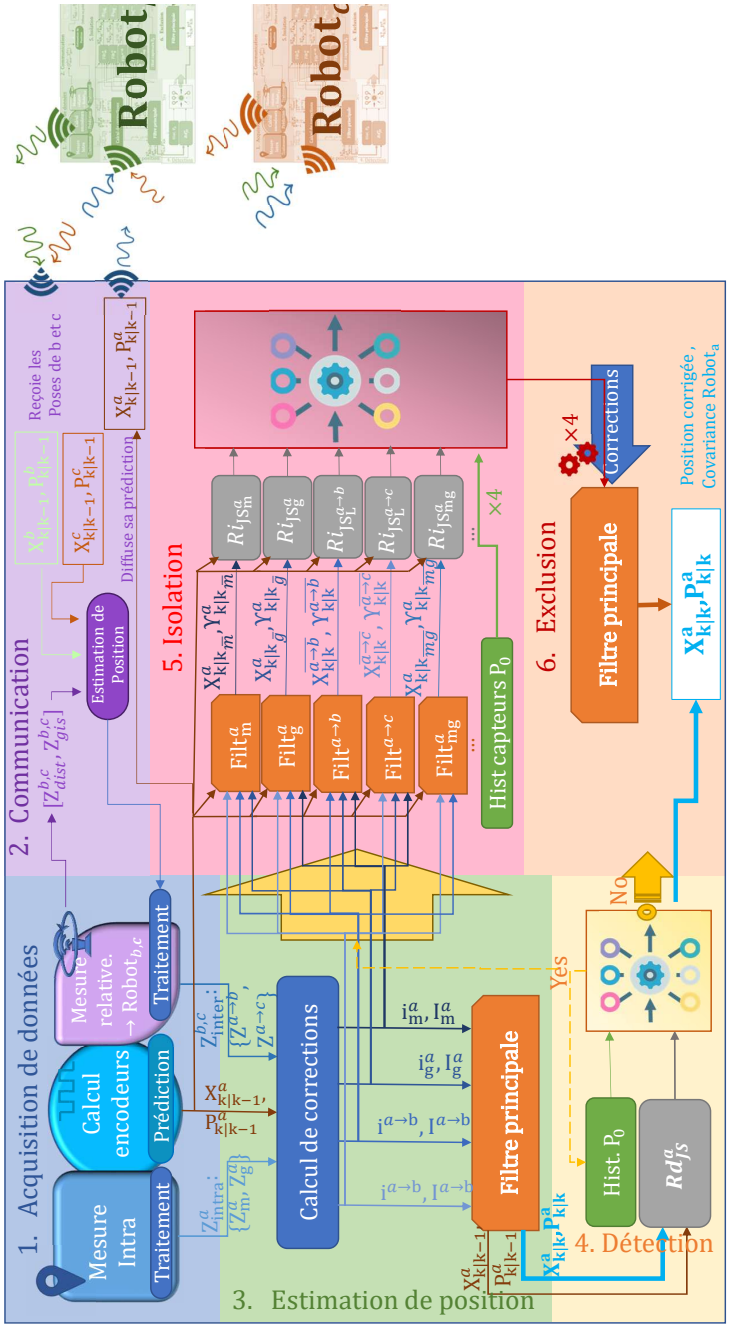


Figure 3.10 – Schéma présentant les différentes parties de l'approche

lot B-CIF,
— Changement du schéma d'observateurs de dédié DOS vers généralisé GOS, afin d'avoir une quantité d'informations similaire entre les résidus,
— Changement de la méthode de communication de la méthode 1 vers la méthode 2,
— Modification des deux étapes de diagnostic : 4. Détection de défaut, et 5. Isolation de défaut.

La partie algorithmique est présentée dans l'algorithme 1, afin d'aider à la compréhension du schéma. La modification apportée à la méthode de diagnostic est la suivante :

4. Détection de défauts : Cette étape consiste à détecter l'occurrence de défauts, à partir de la mesure de dissimilarité entre la prédiction et la correction. Etant donné que les divergences sont des outils adaptés à la nature stochastique des données, elles sont utilisées comme entrées du modèle discriminatif, puisqu'elles portent cette information nécessaire pour pouvoir détecter une incohérence dans les données, qui n'est pas directement visible en apprenant les prédictions et les corrections. Par contre, étant donné que les divergences sont concernées par les estimations d'une seule itération, et que leur valeur peut changer selon l'historique de fonctionnement du système, elles ne suffisent pas. Nous avons besoin de cette information supplémentaire qui peut caractériser le changement dans les niveaux des valeurs des résidus. C'est pourquoi nous avons choisi d'ajouter en entrée du modèle discriminatif de détection, la probabilité préalable de l'hypothèse de non-défaut P_0 de l'itération précédente. Cette probabilité est calculée en divisant le nombre de cas sans défaut par le nombre total d'itérations : $P_0 = C_0/(C_0 + C_1)$ où C_0 représente la somme accumulée des cas non défectueux actuels tandis que C_1 représente celle des cas défectueux. Par la suite, notre modèle de détection possède les entrées / sorties suivantes (voir Fig.3.11) :
   — Entrées du modèle de détection de défauts (de dimension 2) : le résidu de détection Rd^a_{JS} et la probabilité préalable de l'hypothèse de non-défaut P^a_0.
   — Sortie du modèle de détection de dimension 1 et $\in \{0, 1\}$: la décision de détection de défaut. Cette sortie est de type binaire, ayant la valeur 0 si aucun défaut n'est détecté et 1 si un défaut est détecté.

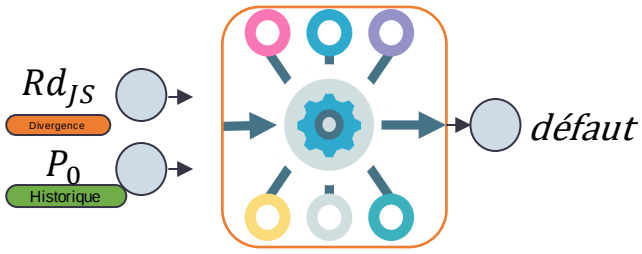


Figure 3.11 – Entrées et sorties du modèle de détection

5. Isolation des défauts : Si l'étape de détection indique la présence d'un défaut, l'étape suivante est l'isolation du capteur défectueux dans le système. Pour ce faire, des observateurs à entrée inconnue de schéma généralisé GOS sont utilisés. Dans ce cas, des corrections intégrant tous les capteurs sauf un à la fois sont calculés, puis des résidus d'isolation comparant ces corrections à la prédiction sont générés. Pour chaque robot ayant L observations où $L = 2 + (3 - 1) = 4$, $(\sum_{j=1}^{L-2} C_L^{L-j}) = 6 + 4 = 10$ résidus d'isolation $Ri_{JS}^{obs_{GOS}^{a}}$ peuvent être calculés, à savoir : $\{m^a, g^a, L^{a\leftarrow b}, L^{a\leftarrow c}, [m^a, g^a], [m^a, L^{a\leftarrow b}], [m^a, L^{a\leftarrow c}], [g^a, L^{a\leftarrow b}], [g^a, L^{a\leftarrow c}], [L^{a\leftarrow b}, L^{a\leftarrow c}]\}$, dont nous choisissons les 4 premiers : $\{m^a, g^a, L^{a\leftarrow b}, L^{a\leftarrow c}\}$.

   Par analogie au choix des paramètres d'entrée de la partie de détection, les résidus d'isolation sont eux aussi évolutifs et leurs valeurs dépendent de l'historique de fonctionnement du système. Pour identifier l'information supplémentaire, identifions initialement les sorties désirées. L'étape d'isolation cherche à trouver le capteur défectueux, donc elle doit indiquer à la fin l'état de chaque correction, si elle est en défaut et doit être exclue, ou bien si elle ne l'est pas, et peut être utilisée dans la fusion de données. Ayant $L = 4$ capteurs, nous aurons un vecteur de sortie de dimension 4. Cela signifie que nous avons besoin de 4 variables qui décrivent l'historique du système. Nous utilisons donc la probabilité préalable de l'hypothèse de non-défaut $P_0^{estim^a}$ de l'itération précédente pour chaque capteur qui estime l'état du système, où $estim^a \in \{marvel^a, gyro^a, Lidar^a, odom^a\}$.

   Donc le modèle d'isolation possède (illustré dans la Fig.3.12) :

— Entrées du modèle d'isolation de dimension 8 : 4 résidus d'isolation $Ri_{JS}^{obs_{GOS}^{a}}$, et 4 probabilités préalables de l'hypothèse de non-défaut $P_0^{estim^a}$.
— Sorties du modèle d'isolation de dimension 4 : la décision de détection de défaut sur un capteur. Cette sortie est de type binaire, ayant la valeur 0 si aucun défaut n'est détecté et 1 si un défaut est détecté.

3.7.2.1 Librairies utilisées

Le langage de programmation utilisé pour la partie apprentissage est le Python3, étant donné qu'il est de source ouverte, et la diversité des librairies qu'il propose permet de développer divers types de modèles. Dans le cadre de ce travail, la librairie la plus utilisée est celle du *scikit-learn*. Souvent abrégée en *sklearn*, cette librairie offre une multitude d'outils et d'algorithmes pour la création, l'entraînement, l'optimisation, l'évaluation et le déploiement de modèles d'apprentissage machine. C'est dans cette librairie que nous trouvons les classes :

1. *"metrics"* : classe dans laquelle nous trouvons toutes les métriques de comparaison détaillées dans la section 3.7.2.2.
2. *"model_selection"* : utilisée pour l'algorithme d'optimisation GridSearchCV (Grid Search Cross-Validation) que nous détaillons dans la

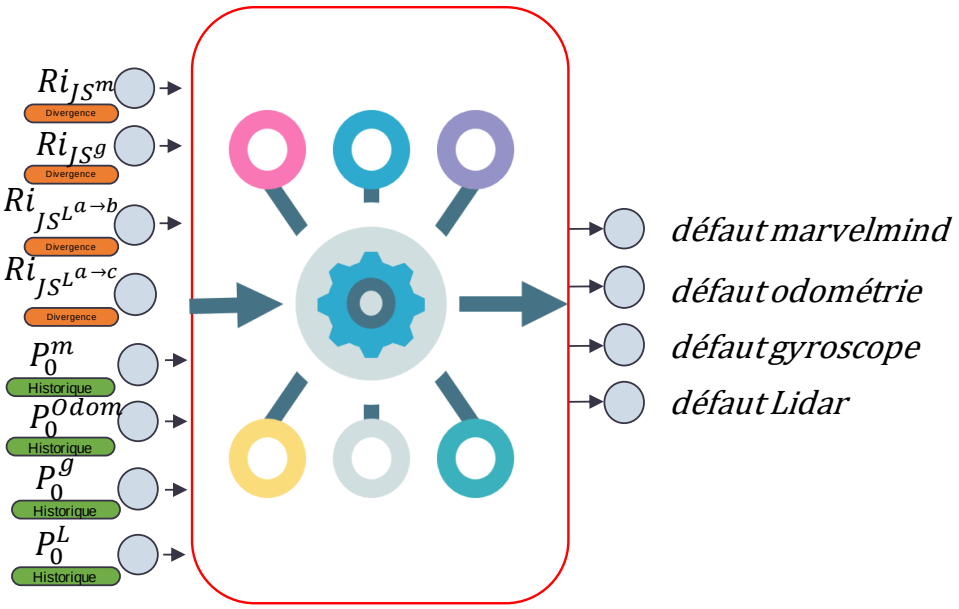


Figure 3.12 – Entrées et sorties du modèle d'isolation de défaut(s)

section 3.7.2.4.

3. *"neural_network"* : comportant la fonction de création du perceptron multi-couche MLP, dont l'usage est détaillé dans la section 3.7.2.5.
4. *"tree"* et *"ensemble"* : classes utilisées pour les classifieurs arbres de décision et forêt aléatoire, développés dans la section 3.7.2.6.
5. *"linear_model"* : utilisée pour le classifieur de la régression logistique présenté dans la section 3.7.2.7.

3.7.2.2 Métriques de comparaison des modèles

Afin de comprendre le comportement des modèles par rapport aux données, nous introduisons des métriques de comparaison utilisées dans le domaine de l'apprentissage machine, afin de qualifier la performance du modèle implémenté. Nous identifions premièrement les classes suivantes de la matrice de confusion :

1. vrais positifs : lorsque le modèle identifie un capteur en bon état comme non-défectueux.
2. faux positifs : lorsque le modèle identifie un capteur en bon état comme étant défectueux.
3. vrais négatifs : lorsque le modèle identifie un capteur défectueux comme étant en bon état.
4. faux négatifs : lorsque le modèle ne détecte pas un capteur en bon état comme défectueux.

Les métriques de comparaison souvent utilisées sont les suivantes :

— **Précision (*Precision*) :** représente intuitivement la capacité du classificateur à ne pas étiqueter comme positif un échantillon qui est en réalité négatif.

$$Precision = tp/(tp + fp)$$

où tp est le nombre de vrais positifs et fp le nombre de faux positifs.

Dans notre cas, cette métrique reflète la capacité de nos modèles d'apprentissage à ne pas noter comme défectueux un échantillon qui ne l'était pas.

— **Exactitude (*Accuracy*) :** signifie que l'ensemble des étiquettes prédites pour un échantillon doit correspondre exactement à l'ensemble correspondant d'étiquettes de l'ensemble de données de test.

$$Exactitude(y, \hat{y}) = \frac{1}{n_{\text{échantillons}}} \times \sum_{i=1}^{n_{\text{échantillons}}-1} 1(\hat{y}_i = y_i) \tag{3.33}$$

Dans notre cas, cette mesure est utilisée afin de trouver la proportion totale de prédictions correctes, c'est-à-dire où le modèle a pu classifier avec succès les hypothèses de fonctionnement selon leurs étiquettes, par rapport à l'ensemble des échantillons.

L'exactitude seule peut ne pas être suffisante pour évaluer pleinement la performance d'un tel modèle. En particulier, lorsque les classes de capteurs en bon état et de capteurs défectueux sont déséquilibrées (c'est-à-dire lorsque l'une des classes est beaucoup plus fréquente que l'autre), l'exactitude peut être trompeuse. Dans de tels cas, d'autres métriques telles que la précision, le rappel et la F-mesure peuvent être plus appropriées pour évaluer la performance du modèle.

— **Rappel (recall) :** représente intuitivement la capacité du classificateur à trouver tous les échantillons positifs (présence de défauts).

$$Rappel = tp/(tp + fn)$$

où tp est le nombre de vrais positifs et fn le nombre de faux négatifs.

Le rappel se concentre sur la minimisation des faux négatifs, ce qui est critique dans notre application où la sécurité est en jeu. Dans le contexte de la détection de défauts de capteurs, un rappel élevé signifie que le modèle parvient à identifier la plupart des capteurs défectueux.

— **Score F1 :** peut être interprété comme une moyenne harmonique de la précision et du rappel, où un score F1 atteint sa meilleure valeur à 1 et sa pire valeur à 0. La contribution relative de la précision et du rappel au score F1 est égale.

$$F1 = 2 \times (Precision \times Rappel)/(Precision + Rappel)$$

Cette métrique est particulièrement utile lorsque nous souhaitons équilibrer à la fois la précision et le rappel. Elle est généralement utilisée lorsque les classes de données sont déséquilibrées, c'est-à-dire lorsque l'une des classes est beaucoup plus fréquente que l'autre.

L'introduction de telles métriques de comparaison vise à qualifier le comportement d'un algorithme d'apprentissage de détection et d'isolation des défauts capteurs dans un système de localisation. Leur importance réside dans :

1. Évaluation complète : l'exactitude seule ne donne qu'une vue limitée de la performance d'un modèle. Dans le contexte de la détection et d'isolation de défauts de capteurs, il est crucial de comprendre comment le modèle d'apprentissage gère les cas où les capteurs sont réellement défectueux (vrais positifs) et comment il évite de classer incorrectement des capteurs en bon état comme défectueux (faux positifs).
2. adaptation aux cas déséquilibrés : dans de nombreuses situations réelles, les cas positifs (capteurs défectueux) peuvent être rares par rapport aux cas négatifs (capteurs en bon état).

3.7.2.3 Gestion des données d'entraînement et de test des algorithmes d'apprentissage

Afin de créer les données, plusieurs scénarios de défauts capteurs sont générés, et par trajectoire. Supposons qu'on ait n robots, équipés chacun de L capteurs, navigant selon tr trajectoires de k_{tr} itérations, et que nous générons scn scénarios sur chacune. Les scénarios indiquent les défauts qui seront injectés sur les données des capteurs acquises. Les défauts sont définis par une durée et une sévérité.Pour chaque trajectoire, cette occurrence est étiquetée par 1 dans les $L \times scn$ vecteurs de sortie de taille k_{tr} chacun, selon les temps de début et fin indiqués par le scénario généré, et leur correspondance dans la durée de la trajectoire. Pour chaque scénario de défaut injecté, Ces données erronées constituent les entrées de l'étape d'estimation de la pose. Après chacune de ces estimations, une étude de cohérence entre la prédiction et la correction est faite, pour générer les indicateurs de défaillance à base de divergence de Jensen-Shannon, un Rd_{JS} et $\sum_{j=1}^{L+n-2} C_L^{L-j}\ Ri_{JS}^{obs}$ (sans le LiDAR du robot donc $L-1$ et tenant compte des $n-1$ autres robots qu'il observe). Le suivi de l'historique est aussi fait par le calcul de la probabilité préalable de l'hypothèse de non-défaut P_0, un pour la détection, et L pour chaque capteur. Ces données sont sauvegardées sous forme de vecteurs qui ont la même taille égale au nombre d'itérations k_{tr} dans lesquels la fusion multi-capteurs est faite. La visualisation de ces données est dans la Fig.3.13.

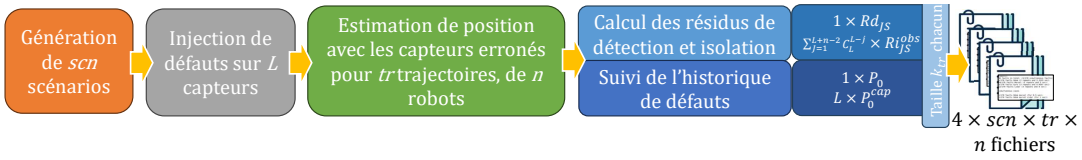


Figure 3.13 – Présentation de la génération de la base de données

Donc pour le modèle de détection, les entrées sont les $n \times tr \times scn$ vecteurs contenant les résidus de détection Rd_{JS}, et $n \times tr \times scn$ vecteurs contenant les probabilités préalables de l'hypothèse de non-défaut P_0, de dimension k_{tr}

chacun. Quant aux sorties, ce sont les $n \times tr \times scn$ vecteurs d'occurrence de défaut, qui sont booléens et obtenus par l'application de l'opération "ou" entre ceux de chaque capteur.

Pour le modèle d'isolation, les entrées sont les $n \times tr \times scn \times \sum_{j=1}^{L+n-2} C_L^{L-j}$ vecteurs contenant les résidus d'isolation Ri_{JS}^{obs}, et $n \times tr \times scn \times L$ vecteurs contenant les probabilités préalables de l'hypothèse de non-défaut P_0, de dimension k_{tr} chacun. Quant aux sorties, ce sont les $n \times tr \times scn \times L$ vecteurs d'occurrence de défaut par capteur.

Les robots et les capteurs sont identiques. Donc pour chaque trajectoire, les poses sont estimées de la même manière, un seul modèle peut ainsi être entraîné et implémenté pour tous les robots. Pour créer les données d'entraînement, constitué de 70% des scénarios, nous faisons une agrégation de données d'entrée et de sortie en un seul vecteur par type d'entrée. Cette agrégation est faite premièrement par l'agrégation des vecteurs des robots, puis par scénario, puis par trajectoire. Une visualisation de cette agrégation est présentée Fig.3.14. Nous aurons ainsi 2 modèles de diagnostic, un pour la détection et un pour l'isolation.

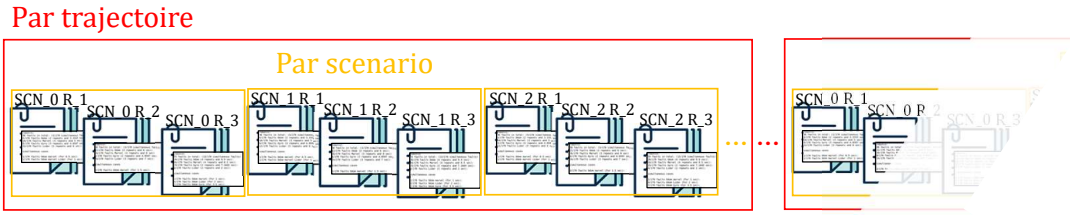


Figure 3.14 – Agrégation des données

Les modèles entraînés sont ensuite communiqués à chaque robot, qui l'intègre dans son étape de diagnostic. Selon ce concept, l'évaluation des données de test constituées des 30% des scénarios restants, se fait par robot, et par trajectoire.

3.7.2.4 Optimisation d'hyper-paramètres

Afin d'améliorer la performance d'un modèle, une optimisation de ses hyper-paramètres est nécessaire. Le terme *hyper-paramètres* désigne les paramètres du modèle qui ne sont pas appris directement à partir des données d'entraînement. Ils sont définis avant le processus d'entraînement et influencent la manière dont le modèle est entraîné et comment il fonctionne. Pour les optimiser, nous utilisons la fonction *GridSearchCV* qui sert à trouver les meilleurs hyper-paramètres selon le type du modèle.

Il fonctionne de la manière suivante :

1. Sélection des Hyper-paramètres à optimiser : tout d'abord, il faut spécifier les hyper-paramètres à optimiser pour le modèle, tels que la valeur de régularisation, la profondeur d'un arbre de décision, le taux d'apprentissage d'un algorithme d'apprentissage, etc.

2. Définition de la grille de recherche : pour chaque hyper-paramètre, il faut spécifier une liste de valeurs possibles. GridSearchCV crée alors une grille de toutes les combinaisons possibles de ces valeurs. Par exemple, si nous optimisons deux hyper-paramètres, A et B, avec trois valeurs possibles pour A et quatre valeurs possibles pour B, la grille de recherche contiendrait 3 x 4 = 12 combinaisons différentes.

3. Validation croisée : pour chaque combinaison d'hyper-paramètres de la grille, GridSearchCV effectue une validation croisée. La validation croisée est une technique qui divise les données en plusieurs ensembles d'entraînement et de test. Elle permet d'évaluer la performance du modèle de manière robuste en évitant le sur-apprentissage. GridSearchCV utilise souvent une validation croisée en *k-fold*, où les données sont divisées en k sous-ensembles, et le modèle est entraîné k fois en utilisant k-1 sous-ensembles comme ensemble d'entraînement et le reste comme ensemble de test (voir Fig.3.15).

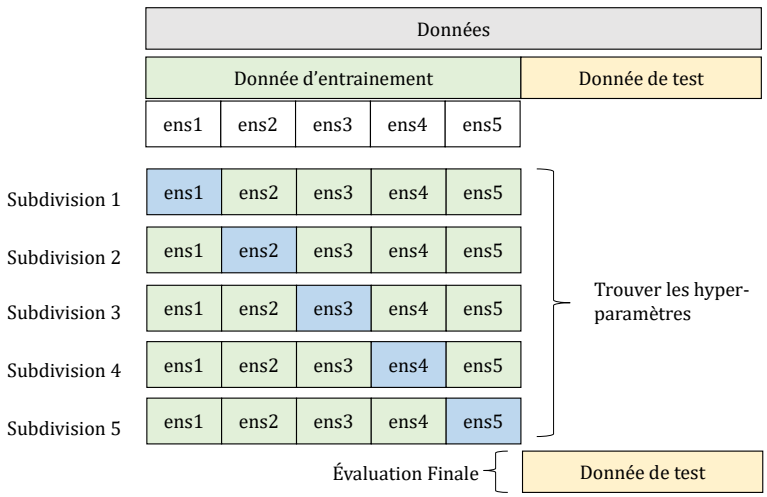


Figure 3.15 – Visualisation de la validation croisée k-folds avec k=5

4. Évaluation de la performance : pour chaque combinaison d'hyper-paramètres, GridSearchCV évalue la performance du modèle en utilisant la métrique spécifiée par l'utilisateur sur les ensembles de test de chaque partie de validation croisée.

5. Sélection du meilleur modèle : une fois que toutes les combinaisons d'hyper-paramètres ont été évaluées, GridSearchCV sélectionne la combinaison qui a donné la meilleure performance selon la métrique définie. Cela constitue le modèle final avec les hyper-paramètres optimisés.

6. Entraînement du modèle final : le modèle final est ensuite entraîné sur l'ensemble complet de données (non divisé) en utilisant les hyper-paramètres optimisés.

En suivant le théorème du "No free lunch" [1] de l'apprentissage, trois types différents de modèles ont été déployés, avec différents problèmes visés et approches pour chacun afin de comparer l'impact de cette modélisation sur les performances du système.

Les trois modèles utilisés sont les suivants :

3.7.2.5 Modèle à base de multi-layer perceptron

La première implémentation intuitive d'un modèle d'apprentissage est celle de NN et parsuite du MLP, en raison de sa capacité à modéliser des relations non linéaires complexes dans les données, qui sont dans notre cas la combinaison des valeurs des résidus et de la probabilité préalable de non-défaut P_0 indiquant la présence d'un défaut.

Ce type de modèle reçoit les données dans sa couche d'entrée, et les transmets au(x) couche(s) cachée(s) qui effectuent des transformations non linéaires sur les données, dont la taille[1] est précisé selon la complexité désirée du modèle. Ces transformations sont représentées par des fonctions d'activation[2], utilisées pour modéliser des relations complexes, créer des représentations abstraites des données, et s'adapter à des problèmes non linéaires. Des exemples de ces fonctions sont :

- la tangente hyperbolique (tanh(x) Fig.C.3) qui écrase les valeurs d'entrée dans l'intervalle [-1, 1]. Elle est centrée sur 0, et attribue des valeurs proches de -1 aux entrées proches d'un extrême de la plage de la fonction et se rapproche progressivement de 1 pour les entrées plus proches de l'autre extrême.
- le sigmoid ($\sigma(x)$ Fig.C.1) qui comprime les valeurs d'entrée dans l'intervalle [0, 1], possédant une forme en S. Il se caractérise par sa capacité à produire des valeurs de sortie proches de 0 pour des valeurs d'entrée faibles et s'approchant progressivement de 1 pour des valeurs d'entrée plus élevées.
- le ReLU (f(x) Fig.C.2) met à zéro les réponses pour les entrées qui ne dépassent pas le seuil et permet de conserver les valeurs plus élevées.

La modélisation de l'entrée du neurone par rapport à sa sortie est présentée par l'équation suivante :

$$y = f\left(\sum_{i=1}^{n}(x_i \cdot w_i) + b\right) \tag{3.34}$$

où :

- y est la sortie du neurone,
- f est la fonction d'activation du neurone,
- x_i sont les entrées du neurone,
- w_i sont les poids associés aux entrées, qui déterminent l'importance de l'entrée dans le calcul de la sortie d'un neurone.

1. The No Free Lunch Theorem : dans les algorithmes d'optimisation et de recherche, ce théorème affirme qu'aucun algorithme n'excelle pour tous les problèmes ; la performance sur un problème se fait souvent au détriment de la performance sur un autre problème.

— b est le biais (bias) ajouté à l'entrée pondérée.

L'entraînement dans le cas de MLP désigne la décision de la valeur du poids et du biais associés à chaque entrée de neurone. Le processus d'optimisation des poids (solver[3]) dans un MLP est généralement réalisé en utilisant la descente de gradient stochastique (SGD). L'objectif est de trouver l'ensemble optimal de poids qui minimise une fonction de perte prédéfinie. Les poids sont mis à jour de manière itérative en utilisant le gradient de la fonction de perte par rapport aux poids.

Soit $\mathbf{W}^{(l)}$ qui représente les poids de la couche l dans le MLP, et soit L la fonction de perte. La règle de mise à jour des poids par SGD peut être exprimée comme suit :

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \nabla_{\mathbf{W}^{(l)}} L \tag{3.35}$$

où :

— η est le taux d'apprentissage, un hyper-paramètre qui contrôle la taille de pas des mises à jour de poids,

— $\nabla_{\mathbf{W}^{(l)}} L$ est le gradient de la fonction de perte par rapport aux poids de la couche l.

Le gradient $\nabla_{\mathbf{W}^{(l)}} L$ est généralement calculé en utilisant la rétropropagation et la règle de la chaîne. Cela implique de calculer les gradients de la fonction de perte par rapport aux activations des couches, puis d'utiliser ces gradients pour calculer les gradients par rapport aux poids.

Le processus d'optimisation des poids se poursuit pendant un nombre spécifié d'époques[4] ou jusqu'à la convergence. Son objectif est de minimiser la fonction de perte, permettant au MLP d'apprendre et de généraliser correctement sur l'ensemble de données donné.

Par les définitions ci-dessus, nous avons identifié 4 hyper-paramètres à optimiser. Le modèle de détection est présenté à la Fig.3.16 et le modèle d'isolation à la Fig.3.17. Les valeurs optimisées et leurs significations dans notre cas sont :

1. *hidden_layer_sizes* : Lors de l'optimisation du modèle de détection, nous avons fait l'étude d'une façon progressive, par l'ajout de neurones puis par l'ajout de couches. En effet, augmenter le nombre de neurones dans une couche augmente la capacité de cette couche à capturer des détails spécifiques entre les données, tandis que l'ajout de couches cachées permet au modèle d'apprendre des représentations de données hiérarchiques et abstraites. En suivant ce principe d'augmentation progressive, nous arrivons pour le modèle de détection D-MLP à une architecture de (150,100,50,50) neurones, et pour le modèle d'isolation I-MLP à une architecture de (200, 100, 100, 50, 25) neurones. Le modèle d'isolation est plus complexe que le modèle de détection. En effet, le modèle d'isolation possède 4 sorties tandis que le modèle de détection en possède une seule.
2. *activation function* : Les fonctions d'activation utilisées sont le ReLU pour le modèle de détection D-MLP et la tangente hyperbolique pour le modèle d'isolation I-MLP. ReLU est capable d'introduire de la sparsité en supprimant les valeurs négatives. Un fonctionnement sans défaut de

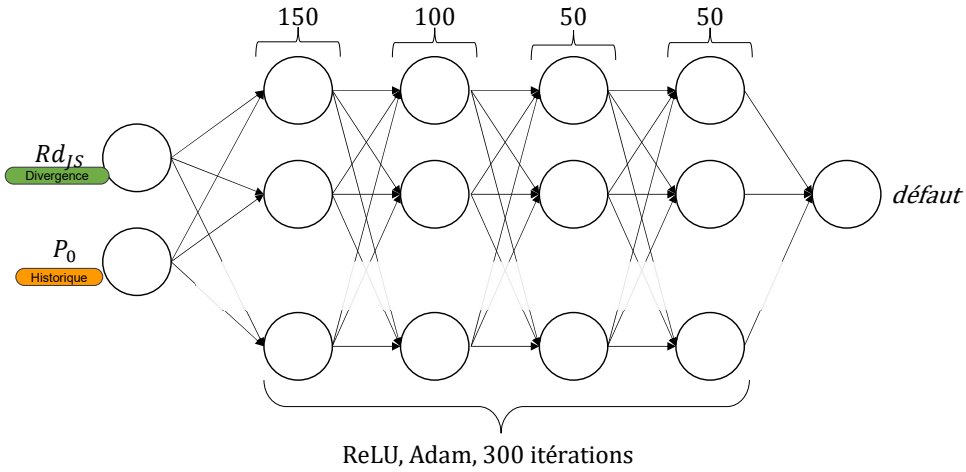


Figure 3.16 – Modèle de détection D-MLP

l'ensemble de capteurs du véhicule produit généralement des données de localisation cohérentes, tandis qu'en cas de défaut, ces données seront atypiques ou aberrantes. En utilisant ReLU, le modèle peut être adapté pour identifier les caractéristiques inhabituelles dans les données des capteurs qui indiquent un défaut. Cette fonction est utilisée pour le modèle de détection.

D'autre part, l'usage de la fonction tangente hyperbolique pour l'isolation suggère une recherche de symétrie et de variations autour d'une valeur de référence (zéro) dans les données des capteurs. Si un capteur présente un défaut, cela pourrait se traduire par des résidus significativement positifs par rapport à zéro, ce qui serait bien capturé par tanh. Cette fonction est utilisée pour le modèle d'isolation.

3. *solver* : Le processus d'optimisation de poids choisi pour les deux modèles de détection et d'isolation est Adam (Adaptive Moment Estimation) plutôt que le SGD. Ce processus est conçu pour améliorer la convergence de l'optimisation. Il utilise deux moments, un moment de premier ordre (moyenne mobile des gradients), et un moment de second ordre (moyenne mobile des carrés des gradients). Puis, il effectue des biais-correctifs pour compenser les biais introduits par les moments estimés lors des premières étapes de l'entraînement pour calculer les mises à jour des poids.
4. *max iteration* : Une itération correspond à un seul passage à travers l'ensemble de données d'entraînement, en utilisant un mini-batch (un sous-ensemble des données) à la fois. Une époque est atteinte lorsque toutes les données d'entraînement ont été parcourues une fois par le modèle, c'est-à-dire lorsque le modèle a effectué un nombre d'itérations égal au nombre total de données d'entraînement divisé par la taille du mini-batch. Le nombre maximal d'itérations est la limite fixée sur le nombre

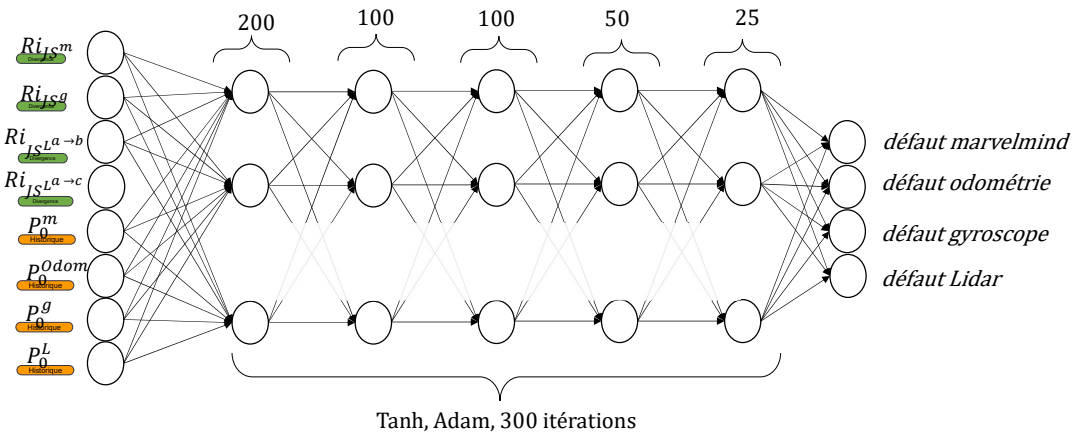


Figure 3.17 – Modèle d'isolation I-MLP

total d'itérations que l'entraînement du réseau de neurones peut effectuer. Si le nombre d'itérations atteint ce maximum avant que d'autres critères d'arrêt (comme la convergence de la fonction de coût) ne soient satisfaits, l'entraînement s'arrête. Cela permet d'éviter que l'entraînement ne se poursuive indéfiniment, ce qui pourrait être inefficace ou conduire à un sur-apprentissage (overfitting), et de contrôler la durée de l'entraînement. Dans notre situation, nous avons présenté une série de valeurs croissantes, parmi lesquelles l'algorithme d'optimisation a choisi un maximum de 100 itérations pour le modèle de détection D-MLP et 300 itérations pour le modèle d'isolation I-MLP.

3.7.2.6 Modèle à base d'arbre de décision / forêt aléatoire

Comme montré dans l'état de l'art, l'arbre de décision était une des premières applications des approches de ML dans le diagnostic. La raison pour laquelle elle a été choisie est son apprentissage par partitionnement récursif, qui est adapté et explicable pour le problème que nous résolvons, à savoir fixer un seuil pour les résidus afin de classer les données et de détecter les erreurs.

L'apprentissage d'un arbre de décision consiste à sélectionner de manière itérative les caractéristiques qui partitionnent le mieux l'ensemble de données. L'objectif est de maximiser la pureté des nœuds fils par rapport au nœud parent.

Ce type a été choisi pour la détection de défauts. Quant à l'isolation, nous avons choisi d'établir une forêt aléatoire, qui est constituée d'un ensemble d'arbres de décision. La raison est que la sortie du modèle d'apprentissage est multi-label multi-classe, ce qui nécessite plusieurs estimateurs.

Afin d'optimiser ces deux modèles, les hyper-paramètres ajustés sont :

— *criterion* : Le critère de division détermine la mesure de qualité utilisée pour choisir la meilleure division à chaque nœud de l'arbre. Bien qu'il existe de multiples façons de sélectionner le meilleur attribut à

chaque nœud, deux méthodes, le gain d'information et l'impureté de Gini, servent de critères de division populaires pour les modèles d'arbres de décision.

Le gain d'information représente la différence d'entropie avant et après une partition sur un attribut donné. L'entropie mesure le désordre dans un nœud. L'attribut présentant le gain d'information le plus élevé produira la meilleure répartition, vu qu'il aura la plus petite quantité d'entropie, et donc le moins de désordre.

L'impureté de Gini mesure la pureté des classes dans un nœud. Plus l'indice de Gini est bas, plus le nœud est pur, c'est-à-dire qu'il contient principalement des échantillons appartenant à une seule classe. L'indice de Gini est calculé en évaluant la probabilité qu'un échantillon pris au hasard dans ce nœud soit mal classifié. L'attribut qui entraîne la plus grande réduction de l'indice de Gini après la partition est choisi comme critère de division.

Dans notre cas, le critère choisi par l'algorithme d'optimisation pour les deux modèles de détection et d'isolation est le critère d'impureté de Gini. Cela signifie que le modèle privilégie la séparation nette des classes dans les nœuds de l'arbre, entre les deux classes avec et sans défaut. De plus, l'impureté de Gini est moins sensible au bruit ou aux valeurs aberrantes par rapport au gain d'information. Cela indique que le modèle possède une certaine robustesse par rapport aux données bruitées ou aux valeurs aberrantes.

— *max_ depth* : Il limite la profondeur de l'arbre en spécifiant le nombre maximal de niveaux entre la racine et les feuilles. Cela aide à prévenir le sur-apprentissage (overfitting). Dans notre cas, nous avons proposé à l'algorithme d'optimisation une marge entre 4 et 10. Pour l'arbre de décision de détection, la profondeur était de 5, tandis que pour la forêt aléatoire, la profondeur maximale par estimateur était de 7. Un arbre de profondeur 5 peut être raisonnablement gérable en termes de compréhension des règles qu'il utilise pour prendre des décisions. La profondeur de l'arbre indique que l'algorithme s'appuie sur plusieurs caractéristiques et leurs combinaisons pour prendre des décisions.

— *n_ estimators* : Cet hyper-paramètre concerne la forêt aléatoire d'isolation. Il fait référence au nombre d'arbres de décision individuels qui seront utilisés pour construire l'ensemble. Chaque arbre dans la forêt aléatoire est construit à partir d'un sous-ensemble aléatoire des données d'entraînement. Dans notre cas, le nombre d'estimateurs est 8.

L'arbre de décision de détection possède comme entrée le résidu informationnel de détection Rd_{JS}, et la probabilité préalable de l'hypothèse de non-défaut P_0. Quant à la forêt aléatoire d'isolation, ses entrées sont les résidus d'isolation $Ri_{JS^{obs^a}}$ ainsi que la probabilité préalable de l'hypothèse de non-défaut de chaque capteur $P_0^{obs^a}$. Cette forêt est activée par l'arbre de décision de détection (D-DT) qui implique la présence de défauts, lesquels sont localisés par la forêt aléatoire d'isolation afin de les éliminer du processus de fusion.

3.7.2.7 Modèle à base de la régression logistique

La régression logistique fournit des résultats interprétables sous forme de coefficients et de rapports de probabilités relatives ou "odds", qui peuvent être utiles pour comprendre l'importance de chaque caractéristique dans la détection de défauts.

Premièrement, les données d'entrée subissent une transformation logarithmique en utilisant la fonction logistique inverse connue sous le nom de fonction logit. En projetant vers cet espace, le problème devient un modèle de régression linéaire de la forme :

$$z = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \tag{3.36}$$

où x_1 à x_n représentent les valeurs des n attributs ou variables d'entrée et β_0 à β_n représentent les poids ou l'interception.

Cette droite est ensuite ajustée par l'usage d'une régularisation, afin de minimiser les distances entre les échantillons et leurs projections vers cette droite. Une fois que c'est réalisé, une conversion inverse vers l'espace d'origine est nécessaire, par l'usage de la fonction sigmoïde. Les étapes de cet ajustement sont présentées dans la Fig.3.18.

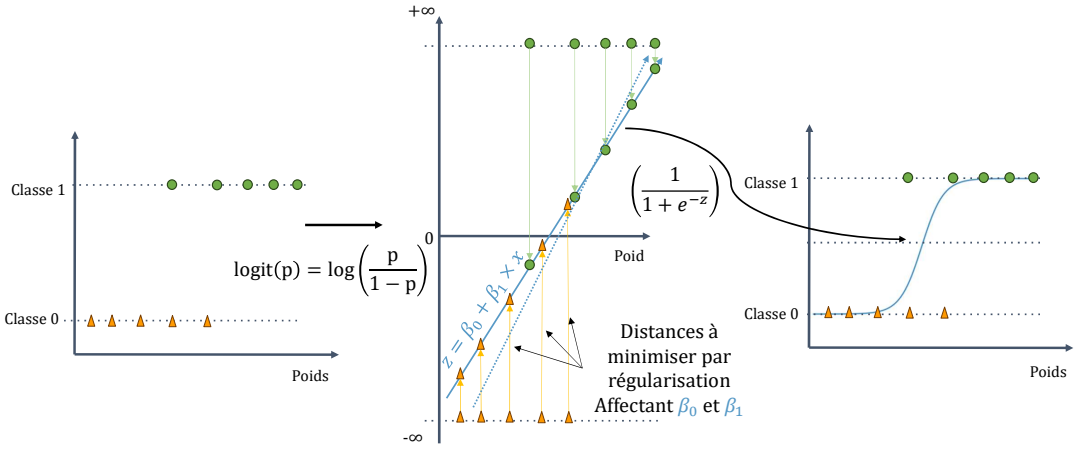


Figure 3.18 – Illustration des opérations de la régression logistique

Dans notre cas, nous rappelons que le modèle de détection possède deux entrées : P_0 la probabilité préalable de l'hypothèse de non-défaut et Rd_{JS} le résidu de détection suivant la divergence de Jensen-Shannon, ce qui signifie que l'équation a trois paramètres, l'ordonnée à l'origine β_0 et les pentes β_1 et β_2, avec une sortie, l'existence ou non d'un défaut.

En ce qui concerne l'isolation, ayant un total de 10 résidus d'isolation $Ri_{JS^{obs^a}}$, 10 pentes β_i au maximum sont alors calculées, avec la probabilité préalable de l'hypothèse de non-défaut P_0 de chaque capteur, ainsi que l'ordonnée à l'origine β_0, ce qui produit 10+4+1 = 15 paramètres β_i à optimiser.

Les hyper-paramètres de ce type d'apprentissage sont présentés dans l'Annexe A.

Dans notre cas, le modèle de détection possède les paramètres suivants :

— *Penality* : La régularisation L2 est choisie pour les deux modèles de détection et d'isolation. Cette régularisation, également connue sous le nom de régularisation Ridge, n'effectue pas de sélection de caractéristiques (contrairement à la régression L1 ou Lasso), elle réduit simplement la magnitude des coefficients. Cela signifie que toutes les caractéristiques d'entrée restent dans le modèle, mais leur importance est pondérée en fonction de leur contribution à la prédiction.
— *solver* : Le solver sélectionné est le "lbfgs" (Limited-memory Broyden-Fletcher-Goldfarb-Shanno). C'est un algorithme d'optimisation numérique utilisé pour ajuster les paramètres du modèle. Il est adapté aux problèmes de régression logistique et est capable de converger vers une solution rapidement.

Dans un système où les données sont produites par différents membres et afin de former un modèle, il existe de nombreuses façons d'organiser les données. Dans ce travail, nous nous concentrons sur deux d'entre elles : l'apprentissage centralisé et l'apprentissage fédéré.

1. **Apprentissage centralisé :** Dans la méthode d'organisation centralisée des données, toutes les données des membres du réseau sont collectées et regroupées en un lieu central. Le modèle d'apprentissage est ensuite formé à l'aide de cet ensemble de données centralisé, puis fourni à chaque membre. Cette méthode est avantageuse en raison de sa facilité de mise en œuvre et d'utilisation de divers algorithmes d'apprentissage. En outre, elle permet une analyse et un traitement complets de l'ensemble des données et peut potentiellement permettre d'obtenir de meilleures performances si l'ensemble de données centralisé est diversifié et représentatif.

   Cependant, elle nécessite le transfert des données des membres individuels vers une unité centrale, ce qui peut soulever des problèmes de confidentialité et de sécurité, et elle lie le fonctionnement du système à la réactivité et à la disponibilité de l'unité centrale. Elle suppose également que l'ensemble de données centralisé représente l'ensemble de la population et peut ne pas tenir compte des variations locales.
2. **Apprentissage fédéré :** En changeant l'architecture pour une architecture décentralisée, l'apprentissage devient fédéré. Cette approche permet d'entraîner un modèle directement sur des données distribuées sans qu'il soit nécessaire de transférer les données brutes vers un emplacement central. Dans l'apprentissage fédéré, le modèle est déployé sur chaque membre du réseau et la formation est effectuée localement sur leurs données respectives. Les paramètres du modèle sont ensuite agrégés ou moyennés à travers le réseau pour créer un modèle global qui bénéficie de la connaissance collective tout en préservant la confidentialité des données. Dans notre cas, ayant le même nombre d'échantillons sur les trois robots pour chaque trajectoire, l'application de la méthode FedAvg correspond à trouver la valeur moyenne entre les paramètres de chaque robot.

   Cette approche préserve la confidentialité et la sécurité des données

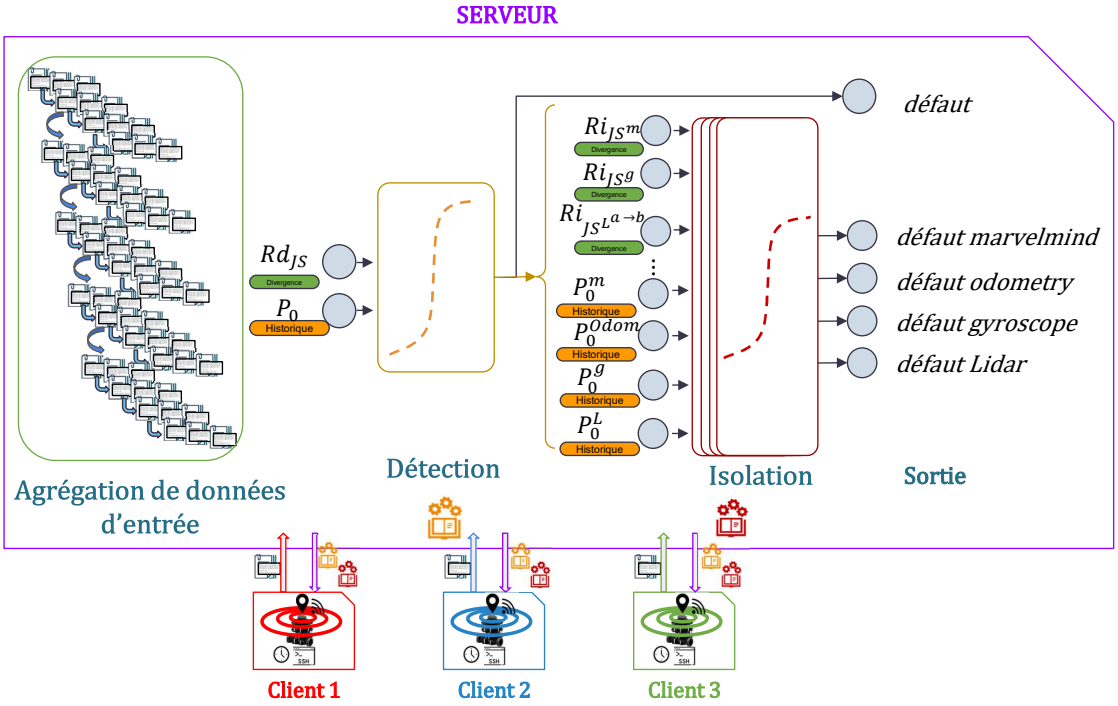


Figure 3.19 – Flux de données de l'apprentissage centralisé

en formant le modèle localement sur les données de chaque membre sans transférer de données brutes. Elle permet également d'apprendre à partir d'un grand nombre d'appareils ou de nœuds distribués, ce qui permet une participation plus large, et de réduire les frais généraux de communication puisque seules les mises à jour du modèle ou les gradients sont partagés, plutôt que les données brutes. Cependant, elle nécessite la synchronisation et l'agrégation des mises à jour du modèle, ce qui peut introduire une complexité supplémentaire. En outre, il peut être confronté à des problèmes de distribution de données hétérogènes entre les membres du réseau, ce qui entraîne des variations dans les performances du modèle, en particulier lorsque les systèmes ont des antécédents de fonctionnement différents, imposant que leur comportement soit quelque peu différent de celui des autres, et donc non adapté au reste du système. La qualité du modèle global dans cette méthode dépend également de la participation et de la qualité des données de tous les membres du réseau.

3.7.3 Conclusion du chapitre

Dans ce chapitre, nous avons présenté les différentes parties de l'approche proposée pour la détection et l'isolation de défauts capteurs pour la localisation coopérative. Nous avons commencé par une formulation du problème et par la suite évoqué les hypothèses de travail. Ensuite, nous avons montré la

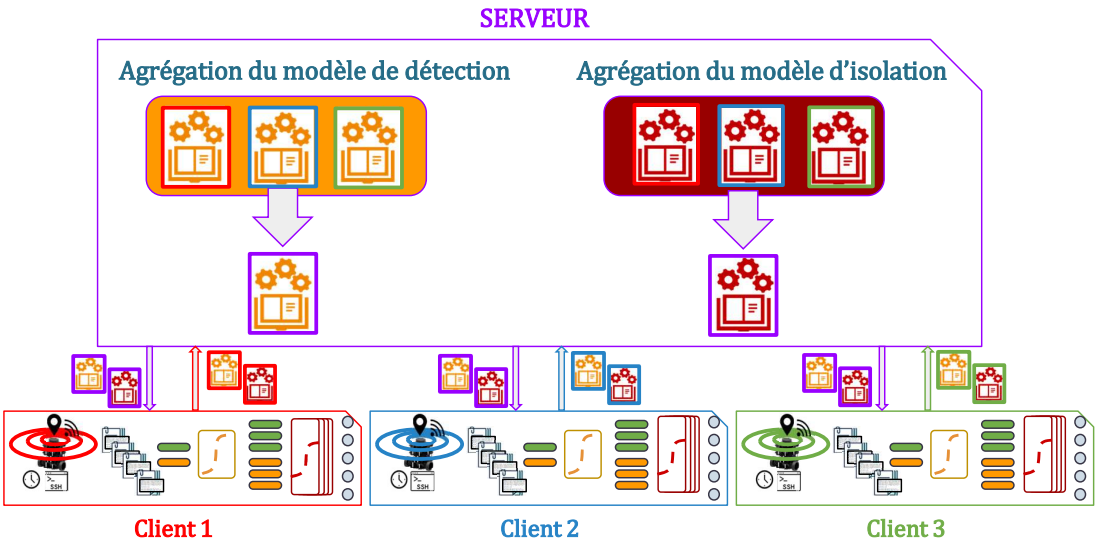


Figure 3.20 – Flux de données de l'apprentissage fédéré

modélisation du système en termes d'évolution et d'observation des capteurs. Puis l'implémentation des modèles afin d'estimer la position du véhicule. Dans cette section, nous avons présenté l'algorithme général de la méthode, et son évolution. Par la suite, la façon de générer les indicateurs de défaillance. Enfin, une présentation des différentes approches de détection et isolation de défauts a été présentée.

Dans le chapitre suivant, nous montrons la mise en place et la création de la base de données capteurs réalisés afin de valider les méthodes proposées, et nous évoquons les résultats de chaque méthode, vis-à-vis des données, la détection et l'isolation de défauts et amélioration de la solution de localisation.

Chapitre 4

Expérimentation et résultats

Pour valider l'approche proposée, une plateforme expérimentale a été conçue, pour permettre l'acquisition des données capteur, avec différents scénarios de navigation. À ces données, on injecte des défauts afin de créer la base de données de scénarios de défauts.

Dans ce chapitre, nous montrons la mise en œuvre de cette plateforme, ses éléments et la manière avec laquelle les défauts ont été injectés. Ces scénarios servant comme des mesures de l'environnement sont utilisés par l'approche proposée de localisation coopérative tolérante aux défauts. Par la suite, nous présenterons les performances de la méthode envers les défauts.

4.1 Plateforme robotique

La plateforme expérimentale conçue est composée de trois robots de type Turtlebot 3 Burger sous Robotic Operating System (*ROS*). Les horloges des robots sont synchronisées en utilisant le protocole de temps réseau NTP et chrony, et l'acquisition des données est assurée par une connexion SSH vers les robots. Le système est observé par un OptiTrack (système de suivi visuel), qui est utilisé comme vérité de terrain. Les balises du système de localisation intérieure (*marvelmind*) sont placées dans des positions spécifiques par rapport au champ visuel de l'OptiTrack afin de faciliter la transformation des repères de l'une à l'autre, et les émetteurs sont installés sur le dessus de chaque robot, en les alignant avec l'origine du robot. Une vue globale de la plateforme est présentée dans Fig.4.1.

Dans cette section, nous détaillons la plateforme de datage et synchronisation de données capteurs *ROS*, les robots Turtlebot utilisés et leurs capteurs embarqués, le système OptiTrack utilisé comme vérité terrain.

4.1.1 Robotic operating system

Malgré l'existence et le développement de nombreux frameworks destinés à la robotique au cours des dernières décennies, le Robot Operating System (ROS) est devenu une référence et le plus populaire et utilisé en raison d'une

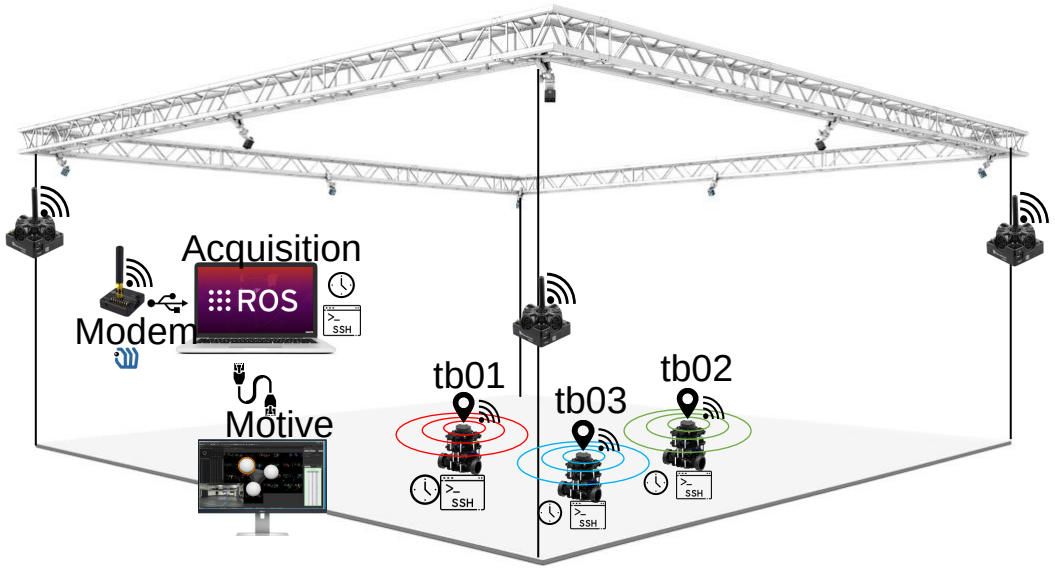


Figure 4.1 – Plateforme prévisionnelle d'acquisition

série de caractéristiques qu'il englobe, et le plus proche de devenir la norme dont la communauté de la robotique avait besoin de toute urgence.

ROS favorise la réutilisation du code avec différents matériels en fournissant une grande quantité de bibliothèques disponibles pour la communauté, comme le SLAM 2D basé sur le laser, la reconnaissance d'objets basée sur un nuage de points 3D, entre autres, ainsi que des outils pour la visualisation 3D (rviz), l'enregistrement d'expériences et la lecture de données hors ligne (rosbag), et bien d'autres.

Les nœuds du système (ROS) fonctionnent principalement selon deux paradigmes de communication : le modèle publisher/subscriber et le modèle client/serveur. En général, les actionneurs agissent en tant que souscripteurs, attendant des commandes, tandis que les capteurs publient les données qu'ils recueillent de l'environnement à une fréquence d'échantillonnage donnée. La transmission d'informations au sein de ROS se réalise à travers un protocole similaire à celui de MQTT (Message Queuing Telemetry Transport), couramment utilisé dans le domaine de la domotique et des objets connectés (IoT). Ce protocole implique la création de "topics" qui contiennent les messages émis par les capteurs. Il convient de noter que ces messages peuvent être standardisés ou personnalisés en fonction des besoins spécifiques de l'application.

4.1.2 Robot Turtlebot3

Le robot choisi pour l'étude est le Turtlebot3 burger développé par ROBOTIS, une entreprise spécialisée dans les robots éducatifs et la robotique open source. L'une des caractéristiques les plus intéressantes du Turtlebot3 est sa modélisation simulable dans Gazebo. Cette simulation réaliste permet de tester

et de développer des algorithmes de contrôle et de navigation sans avoir besoin du matériel physique, et d'injecter des défauts sans détruire le matériel, ce qui économise du temps et des ressources. Ce robot est équipé de :

1. encodeur DYNAMIXEL de résolution 4096 impulsion/tour, utilisé comme entrée du modèle odométrique.

2. RPLIDAR 360 degrés, utilisé pour les interdistances entre les robots,

3. IMU accéléromètre et gyroscope, utilisé comme correction de l'orientation du robot,

4. balise *marvelmind*, utilisée comme correction de la position du robot, pour une solution de localisation intérieure,

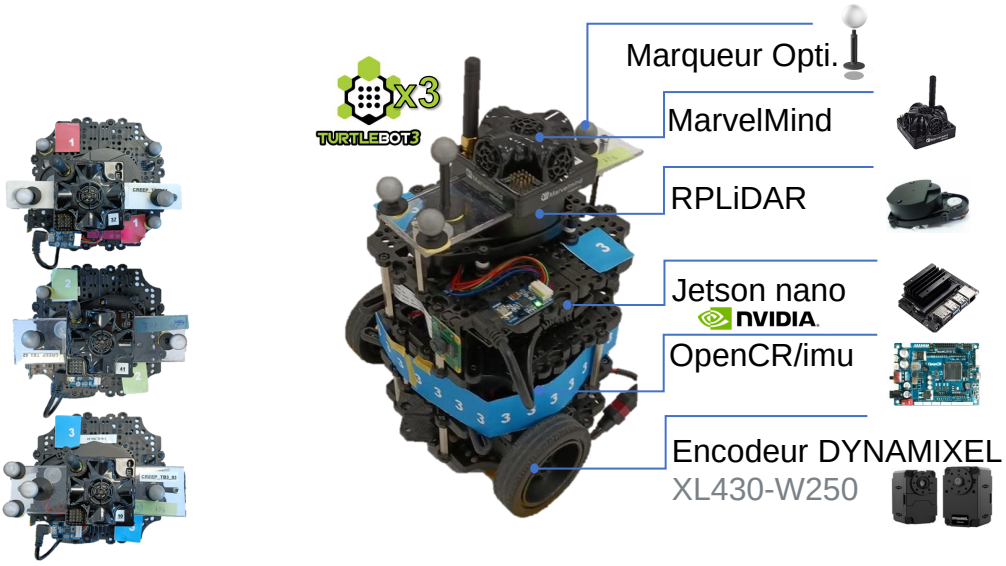
5. caméra RPI3 sur les robots, pour documenter les trajectoires générées.

L'origine du référentiel lié du robot est celui des encodeurs, situé au milieu des roues arrières. Concernant les fréquences d'échantillonnage et la position de chaque capteur relativement à l'origine du référentiel de localisation, ils sont présentés dans le Tableau 4.1.

Paramètre	Valeur	Unité
Rayon de roue	0.0330	m
Distance entre roues	0.16	m
Masse	1.05	kg

Table 4.1 – Paramètres du robot

Au niveau de processeur intégré, par défaut, la Raspberry Pi 3 B+ est équipée. Cependant, nous l'avons remplacé par une Jetson nano de nvidia afin d'améliorer les performances de l'acquisition de données et de bénéficier de sa capacité à connecter des capteurs supplémentaires via des ports USB, comme le cas de la balise du *marvelmind*, une capacité qui n'est pas disponible sur la Raspberry Pi de base. De plus, afin de poursuivre son mouvement, des marqueurs sont installés avec des constellations différentes, dont chacune est définie comme corps rigide sur le logiciel de suivi visuel utilisé comme vérité terrain.



(a) Robots utilisés **(b)** Robot avec les capteurs embarqués

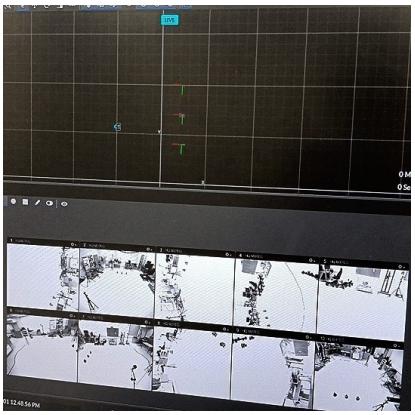
Figure 4.2 – plateforme robotique utilisée

4.1.3 Mise en oeuvre de la plateforme

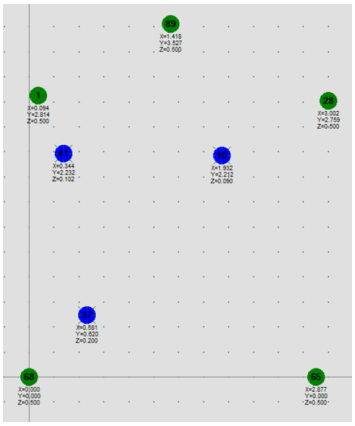
Afin de suivre la trajectoire des robots et créer une référence à laquelle la solution de localisation sera comparée, le système OptiTrack de suivi visuel du mouvement est utilisé.

La mise en place de cette plateforme commence par le calibrage des caméras de l'OptiTrack, en tenant compte du champ d'opération prévu pour les robots. En parallèle, un alignement est réalisé entre les balises du système *marvelmind* et les repères générés du système Optitrack. Les interfaces utilisées pour ces deux systèmes sont présentées à la Fig.4.3. De plus, deux raspberry pi 3 portant des caméras sont installées, une assurant une vision de dessus installée sur les rails des caméras de l'OptiTrack, l'autre assurant une vision locale, installée sur un trépied observant le champ d'opération (voir Fig.4.4). Concernant les robots eux-mêmes, il faut d'abord calibrer les capteurs qui sont installés sur chaque robot. Ensuite, la synchronisation des horloges des robots et des raspberry portant les caméras est effectuée à l'aide de Chrony, garantissant ainsi que toutes les données acquises sont alignées avec le NTP. Enfin, une connexion simultanée SSH est établie pour assurer la synchronisation de la transmission des données depuis les divers éléments vers l'unité d'acquisition.

Avant la mise en place de la trajectoire dans le système OptiTrack, la conception des commandes a été faite dans la simulation Gazebo, pour définir les interdistances initiales, vitesses linéaires et angulaires, afin d'éviter les collisions dans l'acquisition réelle (voir Fig.4.5). Comme le montre la Fig.4.6, le début de chaque trajectoire est marqué, les balises des *marvelmind* sont alignées avec les axes de l'OptiTrack après le calibrage de la caméra.



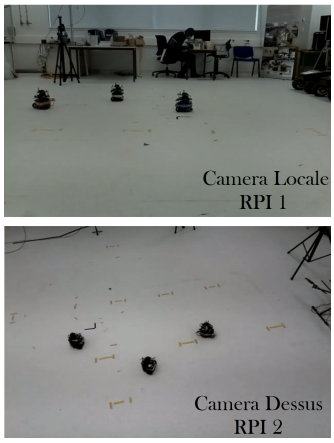
(a) Interface de l'optitrack



(b) Interface de *marvelmind*

Figure 4.3 – Interfaces utilisées pour la création de la base de données

(a) Caméra pour la vue globale du dessus



(b) Vues obtenues par les caméras

(c) Caméra statique vue de côté

Figure 4.4 – Différentes vues de la salle d'expérimentation et d'acquisition

4.2 Préparation de la base de données capteurs

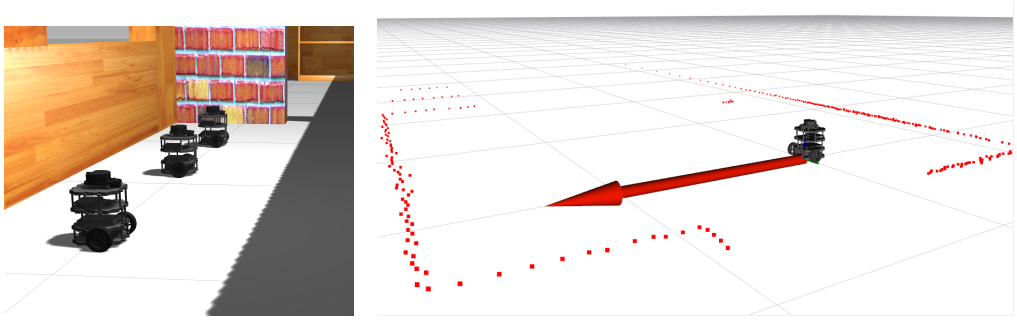
Après le montage de la plateforme et la définition des capteurs utilisés, l'étape suivante est de concevoir la base de données capteurs utilisée. Cette partie comporte deux phases :

1. l'acquisition des données capteurs à travers une plateforme de datage et synchronisation,
2. la génération de la base de données avec défauts capteurs.

À l'issue de ces étapes, cinq trajectoires ont été enregistrées, et sont présentées dans la section 4.2.3.

4.2.1 Acquisition des données

Les données des capteurs des différentes trajectoires sont acquises à travers une connexion SSH vers le robot du PC d'acquisition. Les données des capteurs



(a) Simulation sur Gazebo (b) Données de RVIZ

Figure 4.5 – plateforme en simulation sur Gazebo et RVIZ

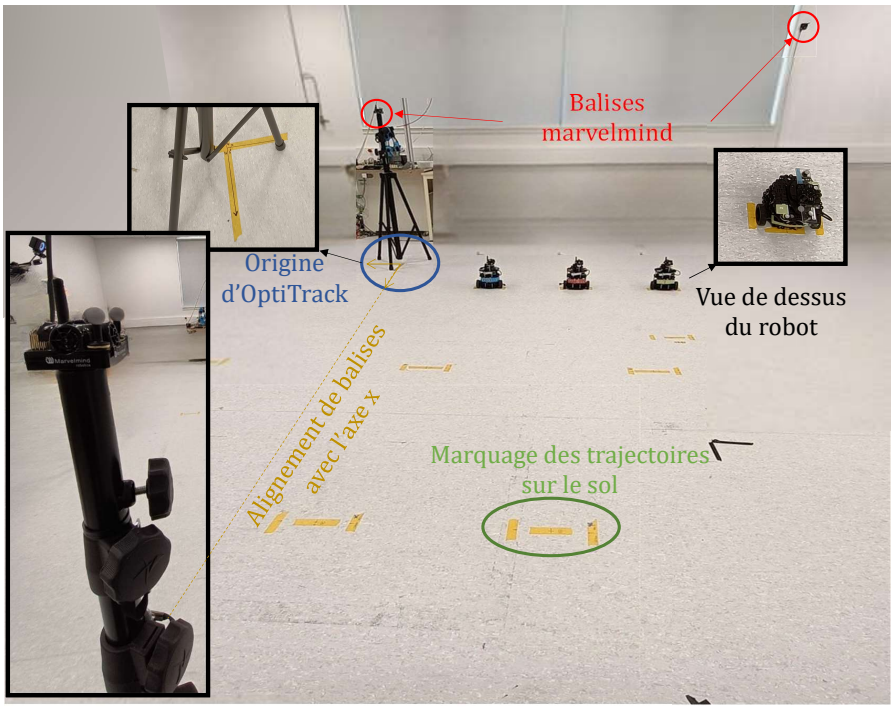


Figure 4.6 – plateforme mise en place

sont datées et synchronisées grâce au format rosbag de sauvegarde de la plateforme ROS. Les données enregistrées sont stockées dans un format spécialisé préservant la structure des messages ROS et incluant des horodatages pour maintenir la séquence temporelle des événements. Rosbag est également utilisé pour rejouer les données enregistrées, ce qui facilite le débogage, le développement et l'analyse des systèmes robotiques sans nécessiter le matériel physique. Ce format possède une extension graphique, le "rqt bag", qui permet une lecture interactive, et de sélectionner et exporter des intervalles de temps de la durée complète, et de visualiser les données capteurs. Cette interface avec les données acquises est présentée dans la Fig.4.7.

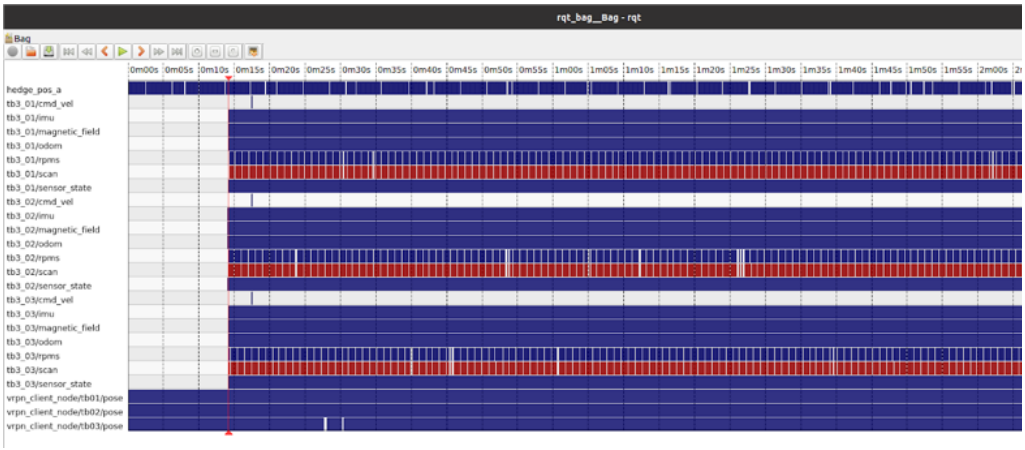


Figure 4.7 – Présentation des données acquises sur l'outil Rqt bag pour la $Trajectoire_1$

4.2.2 Création de la base de données avec défauts capteurs

La simulation de l'apparition de défauts sur les capteurs est réalisée en injectant une erreur en fonction du type de capteur et de sa technologie. Le défaut peut être, soit additif, soit multiplicatif. Un défaut de capteur est généralement considéré comme un défaut additif et peut être assimilé à :

1. un biais :

$$z(k) = z_c(k) + e(k) \quad \text{avec } e = 0 \ \forall \ k \neq k_{\text{défaut}} \text{ et } e \neq 0 \text{ pour } k = k_{\text{défaut}} \tag{4.1}$$

2. une dérive :

$$z(k) = z_c(k) + e(k) \quad \text{avec } |e(k)| = c \times k,\ 0 < c << 1,\ \forall \ k \geq k_{\text{défaut}} \tag{4.2}$$

3. un blocage :

$$z(k) = z_c(k) + e \quad \text{avec } e = \text{ constante } \forall \ k \geq k_{\text{défaut}} \tag{4.3}$$

4. une dégradation des performances :

$$z(k) = z_c(k) + e(k) \quad \text{avec } |e(k)| \leq \hat{e} \ \forall \ k \geq k_{\text{défaut}} \text{ et } \hat{e} > 0 \tag{4.4}$$

5. une erreur de calibrage :

$$z(k) = T \times z_c(k) + e \quad \text{avec } 0 < \hat{T} < T < 1,\ \forall \ k \geq k_{\text{défaut}} \tag{4.5}$$

avec :

— z_c est la valeur de la mesure sans défaut,
— $k_{\text{défaut}}$ est l'itération d'apparition du défaut sur le capteur,
— e est le coefficient de précision du capteur : $e \in [-\hat{e}, \hat{e}]$,
— $T \in [\hat{T}, 1]$ où $\hat{T} > 0$ est l'efficacité minimale du capteur.

Dans le cadre des capteurs utilisés :

1. Le *marvelmind* est extéroceptif, donc le défaut injecté est un biais,
2. Le *gyroscope* est un capteur incrémental se basant sur les vitesses linéaires et angulaires fournies par l'IMU, donc le défaut injecté est une dérive.
3. L'odomètre se base sur les encodeurs incrémentaux, donc le défaut injecté est une dérive.
4. Le *LiDAR* est un capteur de perception, donc il peut être affecté par le blocage, l'absence d'information ou bien le biais provenant d'une onde réfléchie.

Les caractéristiques des capteurs sont les suivantes :

Paramètre	Valeur	Unité
Fréquence odométrie (encodeurs)	30	hz
Fréquence *marvelmind*	2	hz
Écart type pose *marvelmind*	0.05	m
Fréquence *gyroscope*	134	hz
Écart type angle *gyroscope*	0.03	rad
Fréquence *LiDAR*	5	hz
Fréquence de l'OptiTrack	120	hz

Table 4.2 – Caractéristiques des capteurs

En termes de temps de défauts, sa durée est entre 1% et 5%. En cas de défaut unique, un temps de garde est défini entre chaque scénario de capteur, et avec le début du scénario pour la convergence des divergences. Cette occupation temporelle est présentée dans la Fig.4.8, et détaillée dans la Fig.4.9.

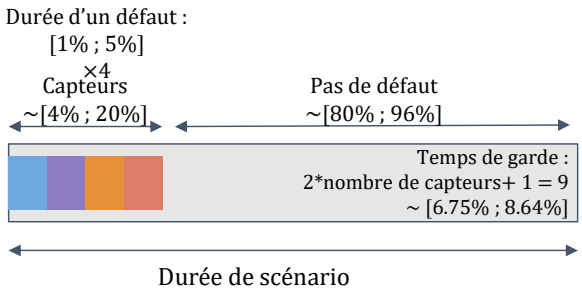


Figure 4.8 – Pourcentage de la durée de défaut dans un scénario

Les données générées sont déséquilibrées, ce qui signifie que la présence de défauts constitue un événement rare par rapport aux données sans défaut. Pour traiter ce problème, deux solutions sont proposées :

1. *Sur-échantillonnage des événements rares/sous-échantillonnage des événements abondants :* L'objectif principal de ces techniques est d'améliorer les performances d'un modèle d'apprentissage machine lorsqu'il est confronté à des ensembles de données où certaines classes sont sous-représentées. Cela permet de réduire le risque que le modèle ignore

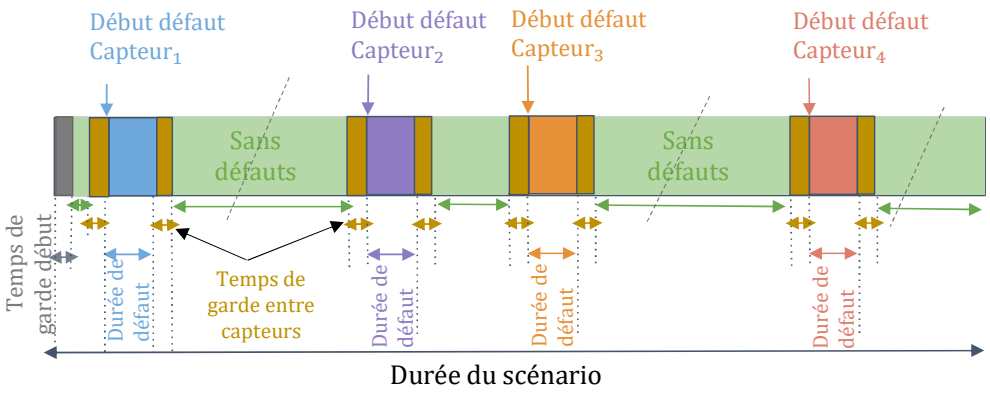


Figure 4.9 – Division d'un scénario

complètement les événements rares en se concentrant uniquement sur les catégories majoritaires.

2. *L'apprentissage sensible au coût :* prend en compte le coût d'une mauvaise classification des différentes classes dans un problème de classification multi-classes. Dans ce cas, un poids doit être accordé à chacune des étiquettes utilisées.

Un exemple de l'injection de défauts sur le *marvelmind* est présenté dans la Fig.4.10. La première ligne de cette figure illustre l'impact de l'injection de défauts sur les coordonnées x et y du *marvelmind*, tandis que la deuxième ligne présente le biais ajouté à chaque coordonnée. La Fig.4.11 présente l'impact de l'injection d'une dérive sur l'angle fourni par le gyroscope. L'injection de défaut sur chaque capteur pour cinq scénarios est visualisée à la Fig.4.12, où le premier de ces scénarios pour les trois robots est présenté à la Fig.4.13.

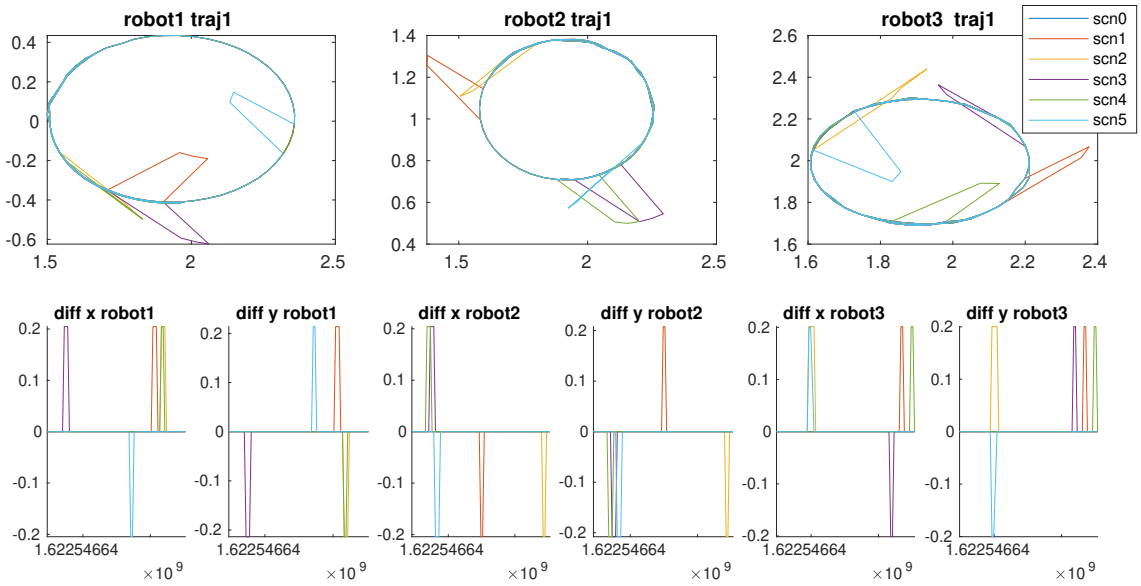


Figure 4.10 – Défauts injectés sur le *marvelmind*, $trajectoire_1$, sévérité 30%, durée entre 1% et 5%

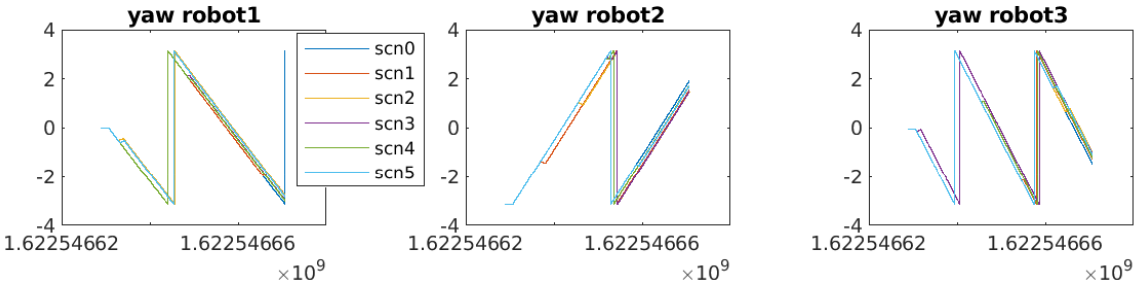


Figure 4.11 – Défauts injectés sur le gyroscope, $trajectoire_1$, sévérité 30%, durée entre 1% et 5%

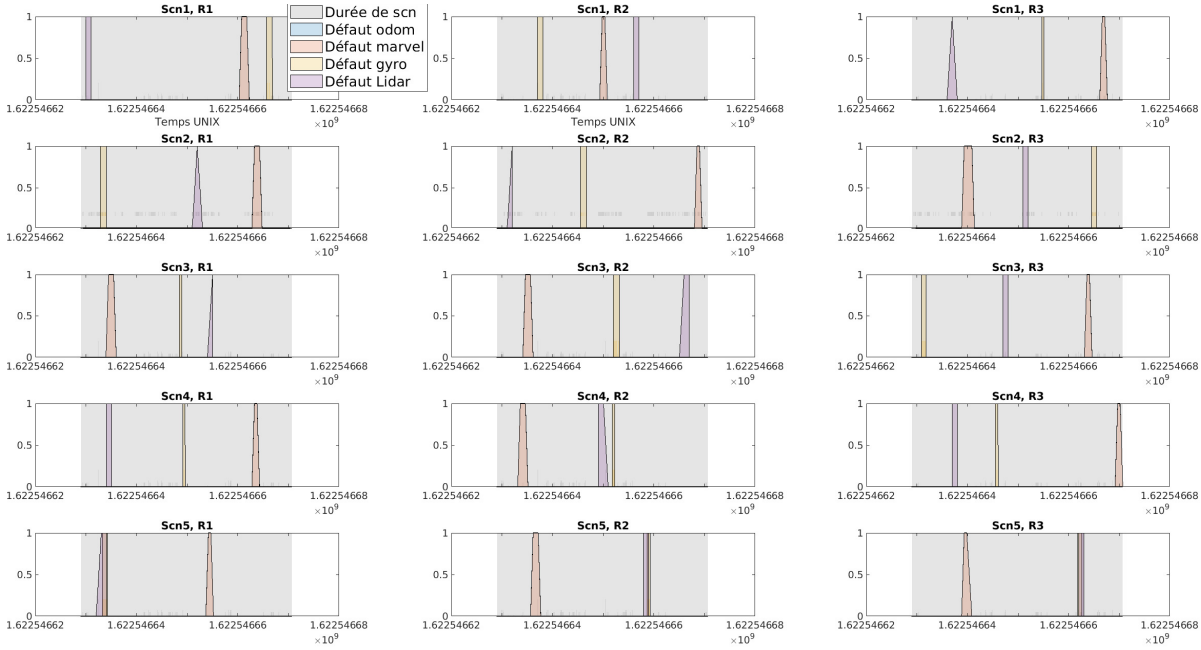


Figure 4.12 – Présentation de l'occurrence de défaut sur les capteurs d'un essai comportant 5 scénarios

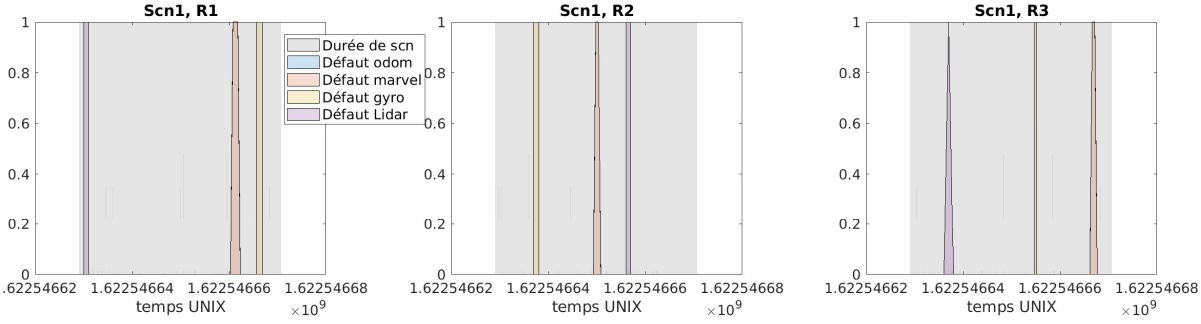


Figure 4.13 – Présentation du premier scénario de défaillance de pour les trois robots

4.2.3 Présentation de la base de données

Cinq trajectoires ont été enregistrées :

— $Trajectoire_1$ (surnommée *Audi*) : elle tente de reproduire l'effet d'une navigation proche et circulaire, et que les robots se rapprochent les uns des autres. Le Tableau 4.3 montre les paramètres de cette trajectoire, et la Fig.4.14 montre son allure.

Table 4.3 – Caractéristiques de la $Trajectoire_1$ Audi

Paramètres	$Robot_1$	$Robot_2$	$Robot_3$
Distance parcourue(mètres)	4.0006	3.7877	4.553
Durée de la trajectoire(secondes)	41.6698	41.6698	41.6798
Nombre d'échantillons des encodeurs	1236	1236	1238
Nombre d'échantillons du *gyroscope*	5585	5847	5600
Nombre d'échantillons du *marvelmind*	80	82	79
Nombre d'échantillons du *LiDAR*	205	207	206
Nombre d'échantillons d'OptiTrack (VT)	5001	5001	4892
Décalage temporel gyroscope et VT (secondes)	0.000942	-0.0068631	0.79473
Décalage temporel *marvelmind* et VT (secondes)	-0.35636	-0.018689	-0.6945
Décalage temporel *LiDAR* et VT (secondes)	0.04348	0.043461	0.043444

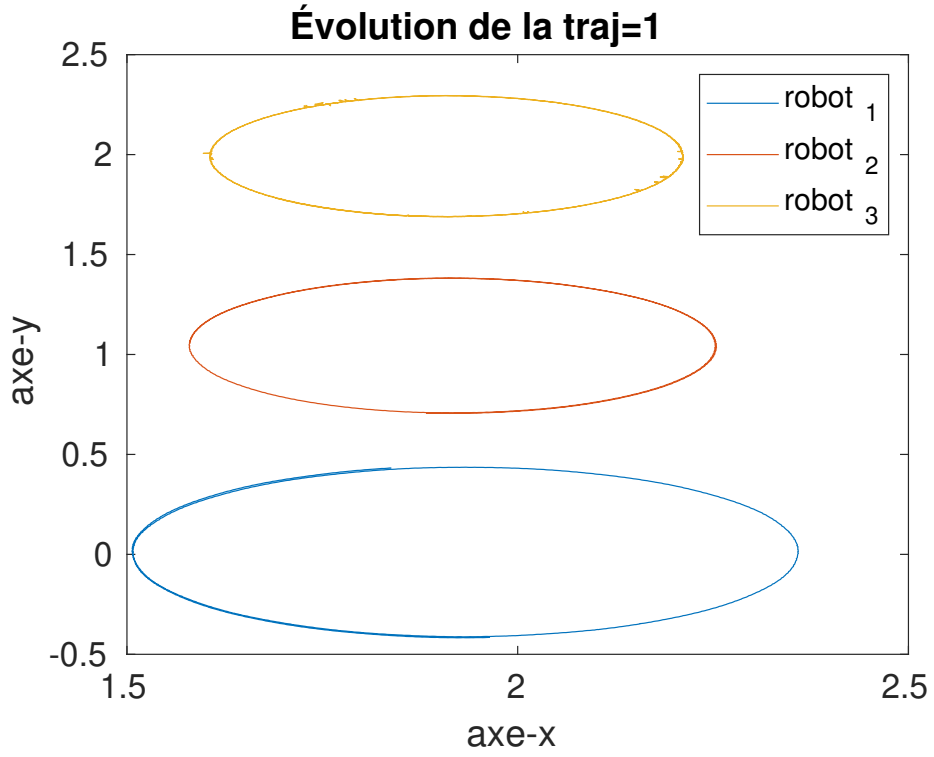


Figure 4.14 – Allure de la $Trajectoire_1$ audi

— $Trajectoire_2$ (surnommée *Circulaire*) : Comporte la navigation concentrique en allure circulaire. Le Tableau 4.4 montre les paramètres de cette trajectoire, et la Fig.4.15 montre son allure.

Table 4.4 – Caractéristiques de la $Trajectoire_2$ Circulaire

Paramètres	$Robot_1$	$Robot_2$	$Robot_3$
Distance parcourue (mètres)	11.7046	10.6605	10.1275
Durée de la trajectoire (secondes)	85.3703	85.3801	85.3801
Nombre d'échantillons d'encodeurs	2524	2535	2532
Nombre d'échantillons du *gyroscope*	11449	11957	11494
Nombre d'échantillons du *marvelmind*	166	169	161
Nombre d'échantillons du *LiDAR*	424	423	419
Nombre d'échantillons d'OptiTrack	10246	10247	9320
Décalage temporel *gyroscope* et VT (secondes)	0.0016313	0.0056024	-0.0067337
Décalage temporel *marvelmind* et VT (secondes)	-0.20813	-0.37629	-0.039353
Décalage temporel *LiDAR* et VT (secondes)	0.093186	0.093219	0.093136

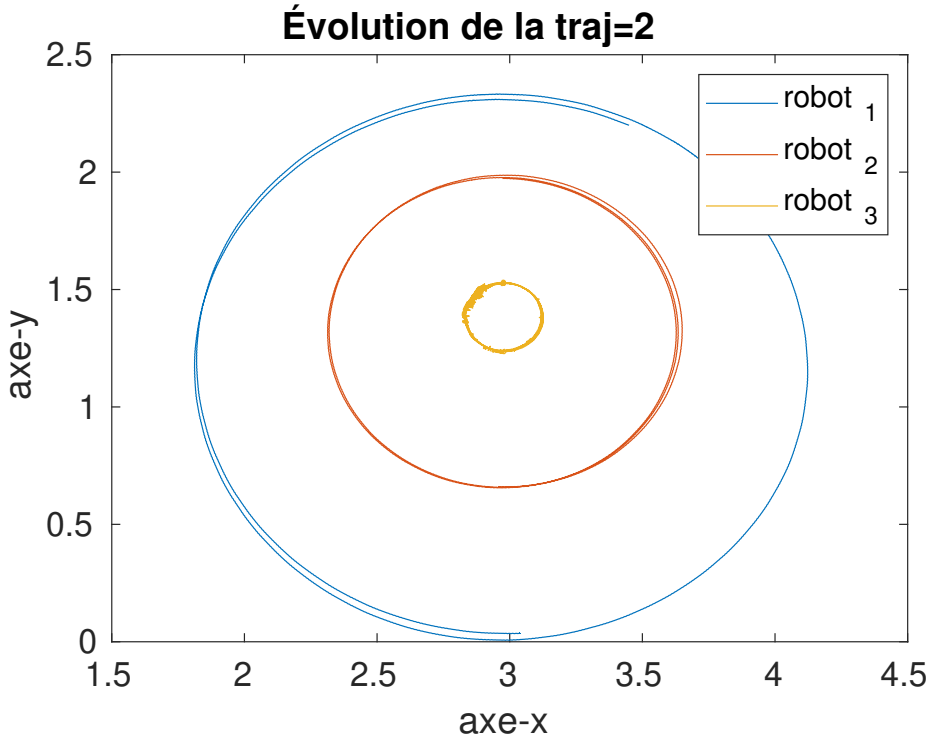


Figure 4.15 – Allure de la $Trajectoire_2$ Circulaire

— $Trajectoire_3$ (surnommée *Carrefour*) : Repose sur le scénario de véhicules entrant simultanément dans un carrefour. Le Tableau 4.5 montre les paramètres de cette trajectoire, et la Fig.4.16 montre son allure.

Table 4.5 – Caractéristiques de la $Trajectoire_3$ Carrefour

Paramètres	$Robot_1$	$Robot_2$	$Robot_3$
Distance parcourue (mètres)	11.5582	10.1319	13.2737
Durée de trajectoire (secondes)	102.4204	102.4204	101.5898
Nombre d'échantillons d'encodeurs	3024	3044	3023
Nombre d'échantillons du *gyroscope*	13600	14508	13624
Nombre d'échantillons du *marvelmind*	202	201	193
Nombre d'échantillons du *LiDAR*	508	503	508
Nombre d'échantillons d'OptiTrack	12225	12177	9606
Décalage temporel *gyroscope* et VT (secondes)	0.000942	-0.006863	0.79473
Décalage temporel *marvelmind* et VT (secondes)	-0.48293	-0.14475	0.47532
Décalage temporel *LiDAR* et VT (secondes)	-0.14693	-0.14693	1.6434

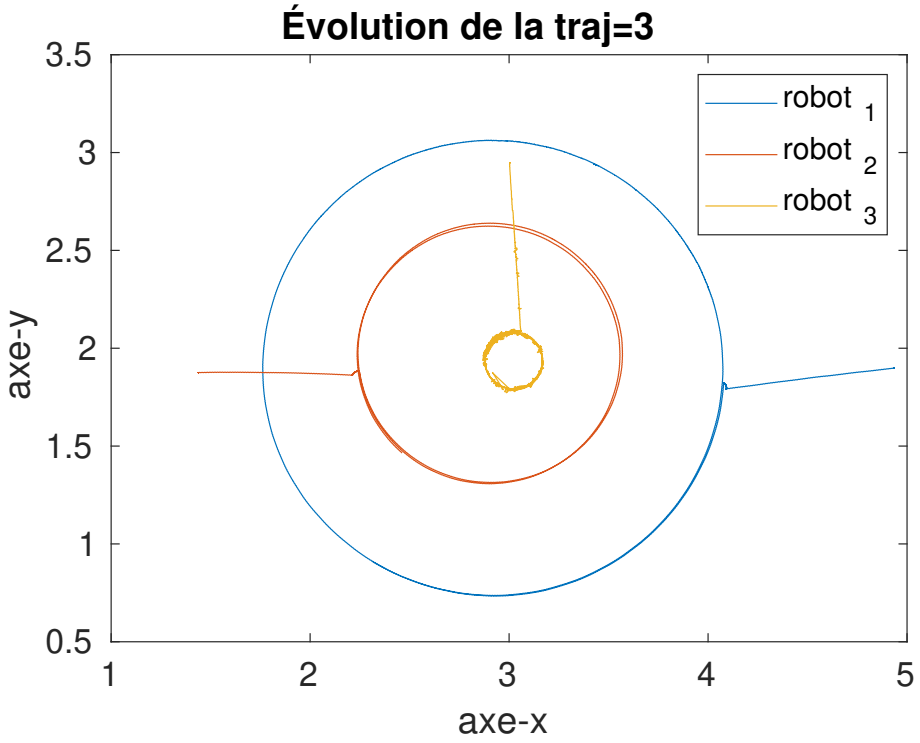


Figure 4.16 – Allure de la $Trajectoire_3$ Carrefour

— $Trajectoire_4$ (surnommée *ZigZag1*) : Un scénario de navigation rectiligne, reproduisant des véhicules sur une autoroute. Le Tableau 4.6 montre les paramètres de cette trajectoire, et la Fig.4.17 montre son allure.

Table 4.6 – Caractéristiques de la $Trajectoire_4$ zigzag1

Paramètres	$Robot_1$	$Robot_2$	$Robot_3$
Distance parcourue(mètres)	4.0283	4.4572	4.5625
Durée de trajectoire(secondes)	48.2702	48.2701	48.2702
Nombre d'échantillons d'encodeurs	1421	1431	1420
Nombre d'échantillons de *gyroscope*	6330	6733	6319
Nombre d'échantillons de *marvelmind*	93	96	87
Nombre d'échantillons de *LiDAR*	241	237	238
Nombre d'échantillons d'OptiTrack	5794	5794	5663
Décalage temporel *gyroscope* et VT (secondes)	-0.0040805	-0.012284	-0.015025
Décalage temporel *marvelmind* et VT (secondes)	-0.41734	-0.081093	-0.2483
Décalage temporel *LiDAR* et VT (secondes)	0.79331	0.79329	0.79325

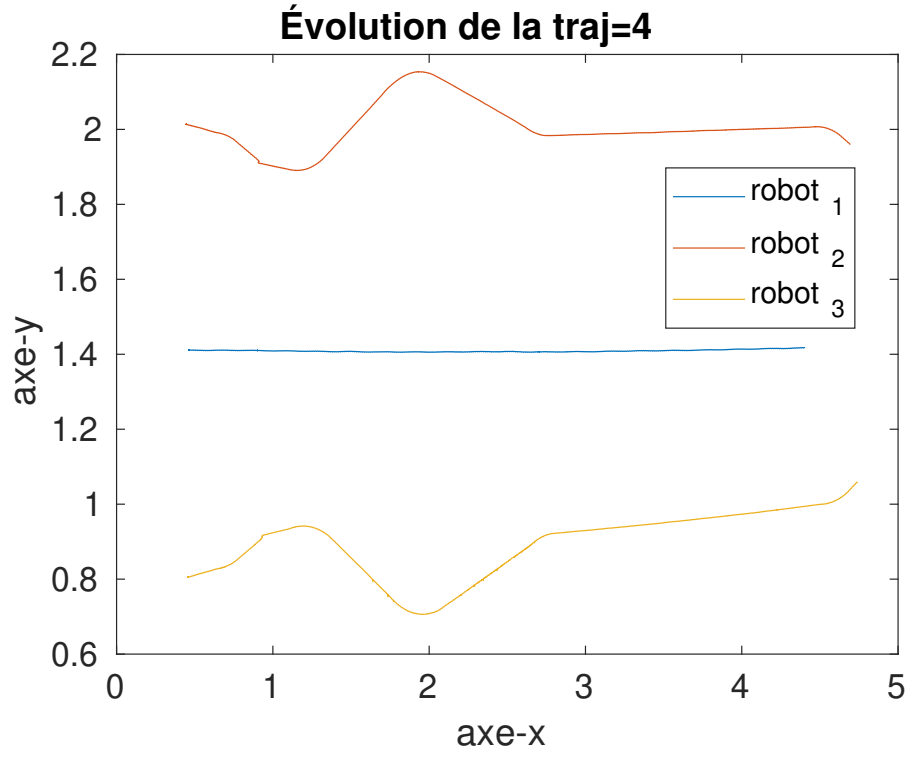


Figure 4.17 – Allure de la $Trajectoire_4$ zigzag1

— $Trajectoire_5$ (surnommée *ZigZag2*) : Un autre scénario de navigation rectiligne, mais avec un excès de rotations. Le Tableau 4.7 montre les paramètres de cette trajectoire, et la Fig.4.18 montre son allure.

Table 4.7 – Caractéristiques de la $Trajectoire_5$ zigzag2

Paramètres	$Robot_1$	$Robot_2$	$Robot_3$
Distance parcourue(mètres)	4.0293	4.4974	4.6788
Durée de trajectoire(secondes)	49.1795	49.1797	49.1797
Nombre d'échantillons d'encodeurs	1449	1459	1447
Nombre d'échantillons de *gyroscope*	6467	6819	6483
Nombre d'échantillons de *marvelmind*	97	96	93
Nombre d'échantillons de *LiDAR*	241	242	243
Nombre d'échantillons d'OptiTrack	5903	5877	5794
Décalage temporel *gyroscope* et VT (secondes)	0.0038135	0.0026867	-0.0090883
Décalage temporel *marvelmind* et VT (secondes)	-0.19823	-0.36834	-0.030271
Décalage temporel *LiDAR* et VT (secondes)	0.78394	0.78376	0.78368

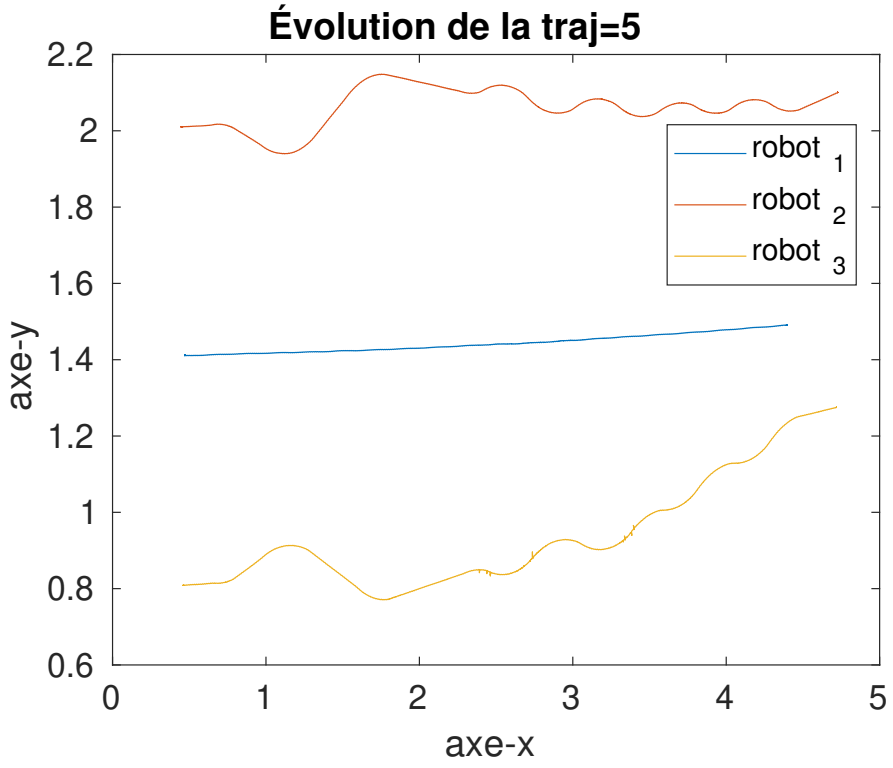


Figure 4.18 – Allure de la $Trajectoire_5$ zigzag2

En conclusion, afin de valider l'approche proposée, les cinq trajectoires enregistrées enregistrent une distance de 93.7177 mètres et une durée de 972.6949 secondes pour l'ensemble des 3 robots. Des erreurs d'acquisition ont également été présentes dans les données. Elles ont été étiquetées en fonction du capteur qui les génèrent par rapport à la vérité terrain.

4.3 Résultats

Suite à la description de la configuration de l'infrastructure d'acquisition, des spécificités des robots, ainsi que des bases de données enregistrées et générées, nous abordons l'étape d'intégration de ces données capteurs dans le processus de localisation. Cette démarche vise à démontrer la capacité des méthodes proposées à identifier et isoler les défauts qui ont ici été intentionnellement introduits.

4.3.1 Résultats de la méthode classique

Dans un premier temps, nous présentons l'application de la méthode classique définie à la section 3.7.1. Pour la trajectoire circulaire ($Trajectoire_2$) choisie, des défauts ont été injectés de la manière suivante :

1. Dérive sur le *gyroscope* (indiquée en bleu sur la Fig.4.21) du :
   — robot 1 aux instants [136 :141]
   — robot 2 aux instants [144 :149]
   — robot 3 aux instants [4 :10] et [74 :81]
2. Biais sur le *marvelmind* (indiqué en orange) du :
   — robot 1 aux instants [102 :108] et [133 :138]
   — robot 2 aux instants [20 :26] et [50 :55]
   — robot 3 aux instants [41 :47] et [120 :127]
3. Absence de données sur le $LiDAR$ (indiquée en magenta) du :
   — robot 1 aux instants [52 :57] et [84 :88]
   — robot 2 aux instants [69 :77]
   — robot 3 aux instants [129 :135].

La trajectoire obtenue en présence des défauts injectés, sans application de diagnostic, est présentée dans la Fig.4.19. On peut observer comment elle diverge et à quel point elle diffère de la vérité terrain.

Pour ce cas, les résidus de détection sont présentés dans la Fig.4.21, et les résidus d'isolation sont présentés dans la Fig.4.22, où, pour chaque robot, il y a trois résidus et deux indicateurs reçus des robots voisins. Ces résidus sont comparés aux seuils calculés par l'indice Youden de la courbe ROC. Ensuite, le motif de leur apparition est comparé à la matrice de signature présentée dans le Tableau 3.2 pour déterminer le type d'erreur en cours.

La fonction de densité de probabilité (PDF) des résidus pour les cas avec et sans défauts est présentée dans la Fig.4.23a, pour le résidu $Ri_{JS}^{m^3}$, le résidu d'isolation basé sur la divergence de Jensen-Shannon pour le *marvelmind* du robot 3. Dans cette figure, il est évident que dans le cas de défaut, elle a tendance à s'aplatir davantage et à s'éloigner de zéro. Ceci est dû au fait que dans un cas de défaut, un résidu, qui est l'indicateur de la dissimilarité, a tendance à avoir une valeur plus élevée. Nous devons souligner que la divergence de Jensen-Shannon, qui est une forme symétrique de la divergence de Kullback-Leibler, suit une distribution de données de Fisher-χ^2.

L'application de cet algorithme a permis l'identification de certains des défauts injectés ainsi que des erreurs natives qui n'ont pas été volontairement

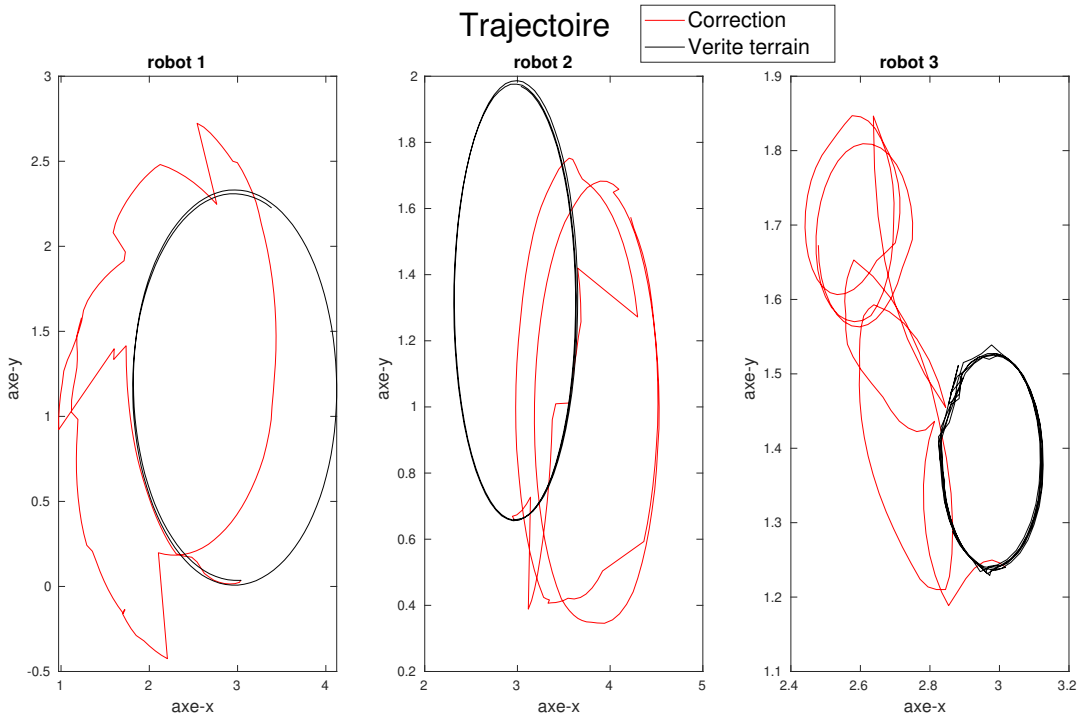


Figure 4.19 – Résultats de la fusion de données en présence de défauts sans étape de diagnostic

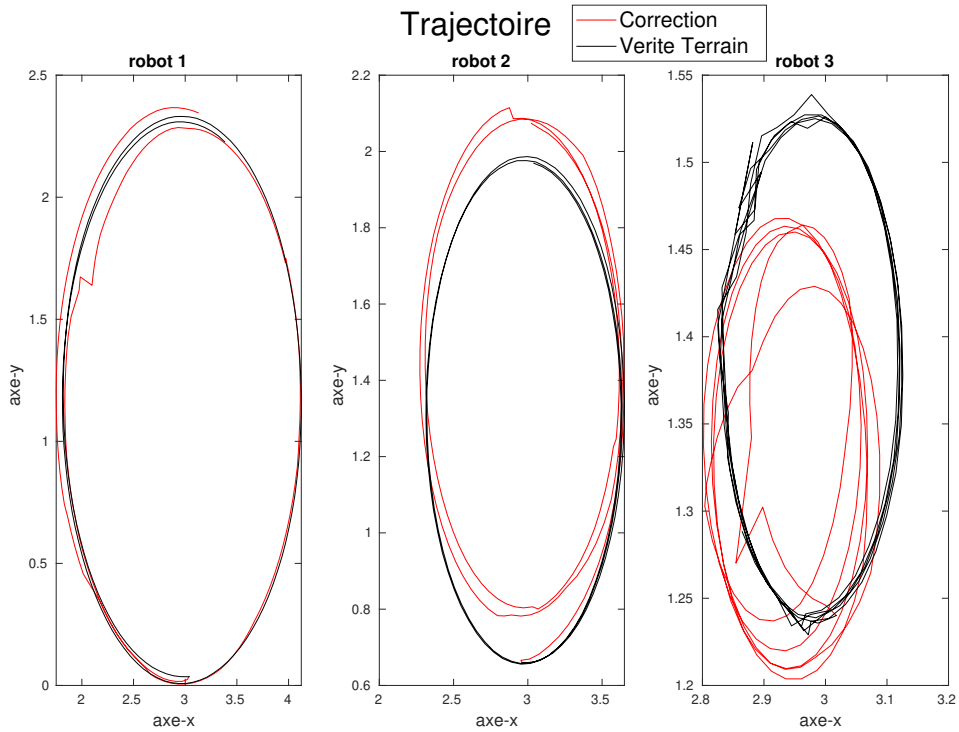


Figure 4.20 – Trajectoire après la détection, l'isolation et l'exclusion des défauts

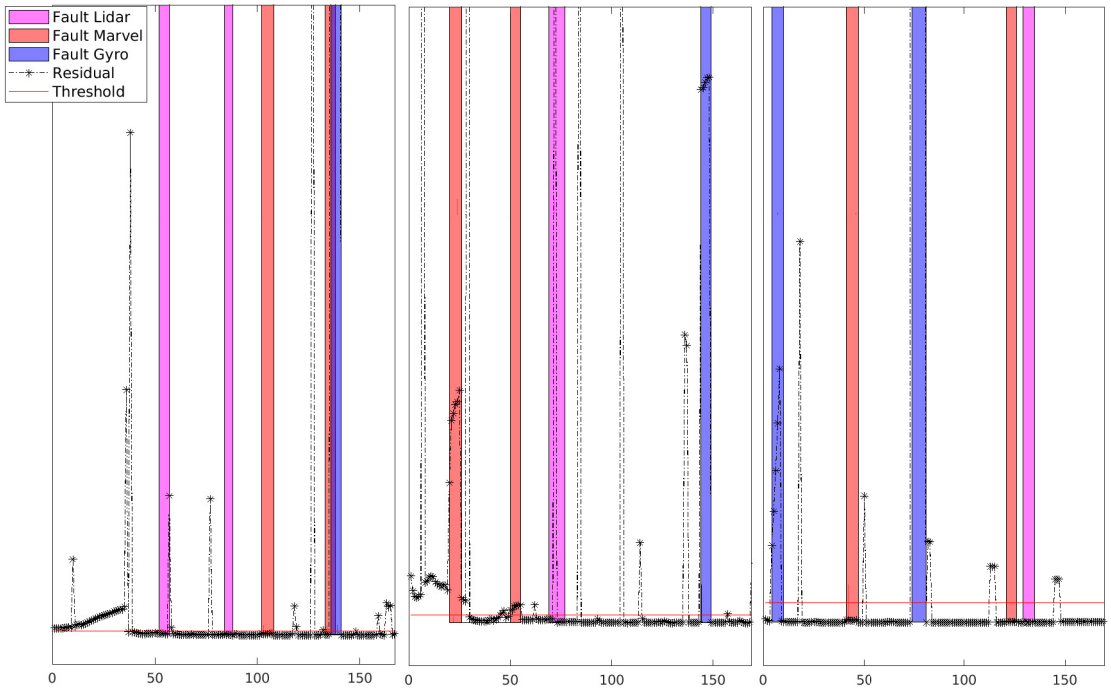


Figure 4.21 – Résidus de détection pour les trois robots, indiquant le seuil constant calculé.

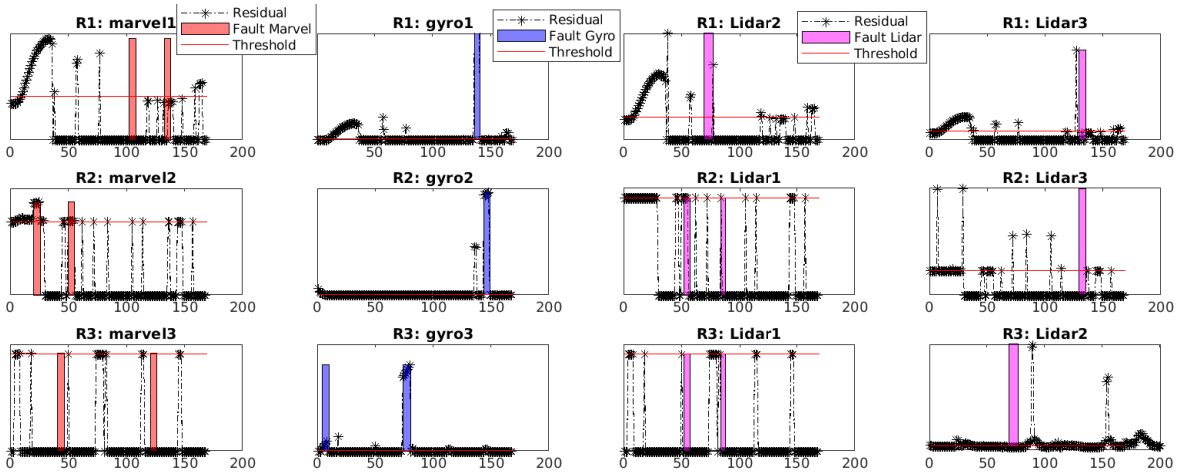
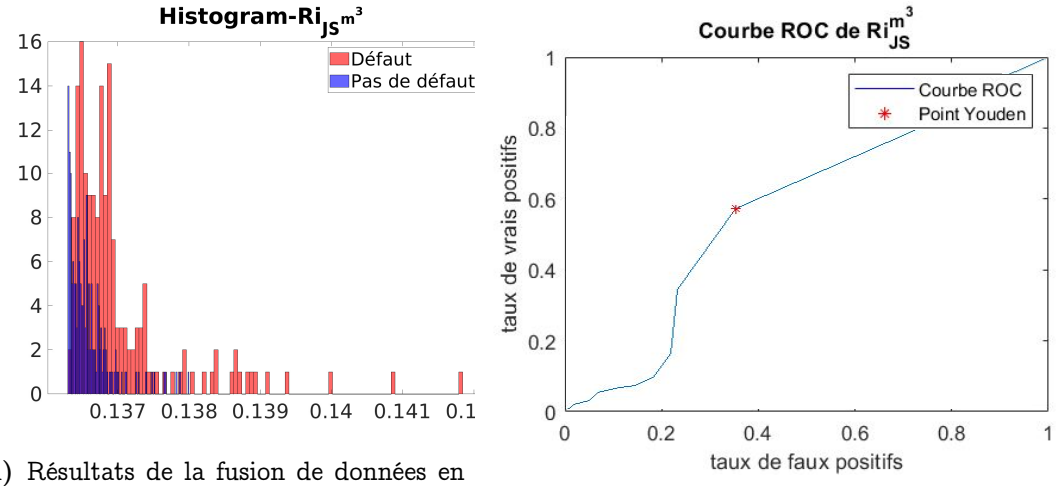


Figure 4.22 – Résidus d'isolation DOS pour les trois robots, avec leurs seuils.

injectées : ce sont les erreurs réelles qui ont affecté le capteur lors de l'étape d'acquisition. Nous remarquons qu'au début de la trajectoire, il y a une isolation pour presque tous les capteurs. Ensuite, une fois que les robots sont calibrés après l'évolution de la trajectoire, les données injectées et réelles sont isolées au fur et à mesure de leur détection. Les défauts du $LiDAR$ indiqués par les résidus sont des erreurs dans l'interdistance obtenue et dans l'angle de gisement obtenus à partir du $LiDAR$. Cela peut être dû à un nuage de points très clairsemé autour du robot obtenu par le balayage du $LiDAR$, ce qui si-



(a) Résultats de la fusion de données en présence de défauts sans étape de diagnostic

(b) Courbe ROC présentant le point de Youden utilisé

Figure 4.23 – Étude statistique du résidu $Ri_{JS^{m^3}}$, le résidu d'isolation basé sur la divergence de Jensen-Shannon pour le *marvelmind* du robot 3

gnifie qu'il n'a pas convergé vers la position réelle du robot, et ce faisant, la position est biaisée, entraînant une erreur.

En fin de compte, l'étape d'exclusion produit la trajectoire présentée dans la Fig.4.20, aboutissant à l'isolation de :

1. Sur le *gyroscope* :
   — **robot 1** à {[9 :39], 44, 59, 78, 90, 91, [118 :120], 128, [137 :141], 149, 160, 164}.
   — **robot 2** à {[2 :8], 137, 138}.
   — **robot 3** à {[5 :9], 19, 51, [75 :81], 83, 84, [114 :116], [146 :148]}.
2. Sur le *marvelmind* :
   — **robot 1** à {[9 :39], 44, 59, 78, 85, 90, 91, 149, 160, 164}.
   — **robot 2** à {[5 :8], [21 :26], 30}.
   — **robot 3** à {[5 :9], 19}.
3. Sur le *LiDAR* :
   — **robot 1** détecté par le robot 3 à {9}.
   — **robot 2** détecté par le robot 1 à {[6 :30], 137, 138}.
   — **robot 3** détecté par le robot 1 à {[6 :9], 19, 78} et par le robot 2 à {8,115}.

En comparant les défauts détectés aux défauts injectés, on remarque qu'on a une détection sur tous les capteurs pour les premières itérations. Durant ces itérations, le résidu n'a pas encore convergé, et sa valeur est supérieure au niveau de la suite de ses valeurs. De plus, les données du scénario nominal n'étaient pas nettoyées à ce stade, et des défauts réels sur les mesures des capteurs ont eu lieu durant l'acquisition. C'est pour cette raison qu'on trouve une détection des défauts injectés et d'autres défauts qui peuvent être des défauts d'acquisition.

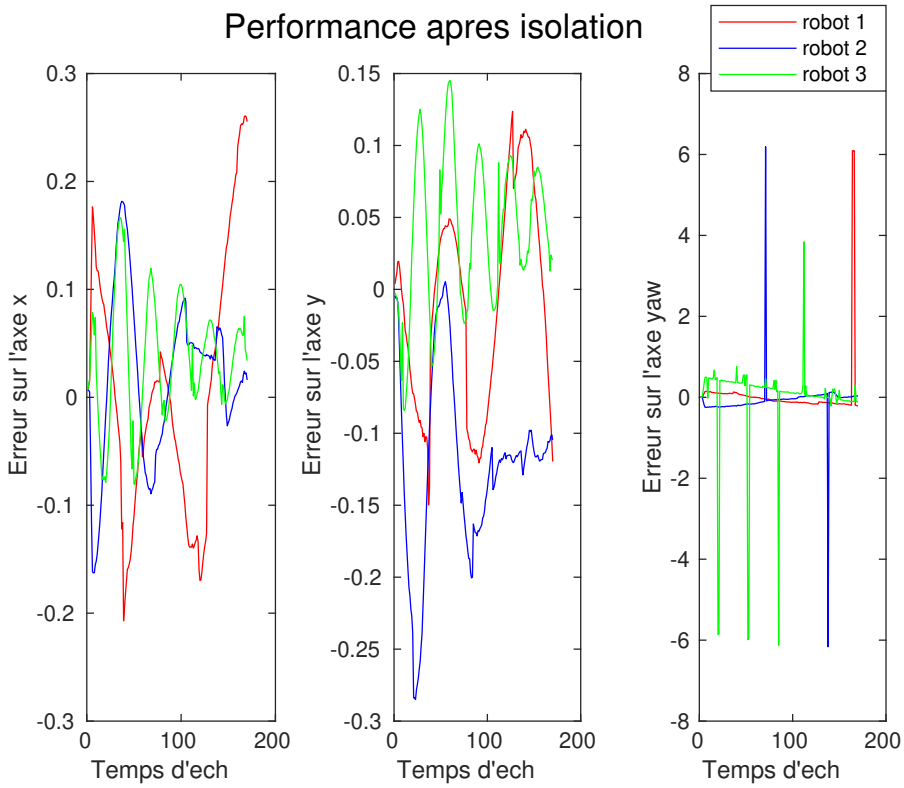


Figure 4.24 – Performance par rapport à la vérité terrain, en mètres pour les coordonnées x et y, radiant pour la composante yaw

La comparaison de la position obtenue par l'approche de fusion tolérante aux défauts avec le système de vérité terrain est présentée dans la Fig.4.24, où la déviation quadratique moyenne racine ($RMSD$) de l'erreur avant et après l'isolation est présentée dans le tableau 4.8.

	avant isolation		*après isolation*	
$Robot_1$	x	0.8290	x	0.0198
	y	0.7520	y	0.0071
	θ	3.7844	θ	0.6680
$Robot_2$	x	0.5648	x	0.0056
	y	0.1110	y	0.0223
	θ	2.6594	θ	0.4664
$Robot_3$	x	0.1316	x	0.0043
	y	0.0728	y	0.0046
	θ	6.3933	θ	1.2030

Table 4.8 – Déviation quadratique moyenne racine (RMSD) (en mètres pour x et y et en radians pour θ)

4.3.2 Performance des algorithmes d'apprentissage

Lorsqu'on passe vers les méthodes de diagnostic guidées par les données, il est crucial de distinguer deux aspects essentiels : d'une part, l'évaluation des performances par rapport aux données elles-mêmes, et d'autre part, l'évaluation des performances en ce qui concerne la détection de défauts. Cette distinction est fondamentale dans le cadre de la recherche en diagnostic assisté par l'intelligence artificielle, car elle permet de mesurer à la fois l'efficacité des méthodes dans la compréhension des données et leur capacité à identifier de manière fiable les anomalies.

Comme présenté dans la section 3.7.2, trois algorithmes ont été étudiés dans ce contexte. La machine utilisée pour effectuer l'apprentissage est un PC *16 RAM core-i5* sous le système d'exploitation *Ubuntu 20.04 LTS*. Parmi les scénarios générés, 70% ont été utilisés pour l'entraînement de l'algorithme, tandis que 30% ont été utilisés pour le test.

4.3.2.1 Résultat pour des modèles basés sur des perceptrons multicouches

Dans une première application, les MLP sont appliqués dans la phase de diagnostic, un pour la détection, l'autre pour l'isolation. Leurs hyperparamètres étaient optimisés grâce à la fonction "GridCVSearch" de Python, cherchant à maximiser l'exactitude (accuracy) (voir section 3.7.2.5).

Pour le D-MLP de détection, il possède 4 couches avec respectivement 150, 100, 50 et 50 neurones, utilise la fonction d'activation 'relu', un taux d'apprentissage constant de 1e-06, pour un nombre d'itérations maximal de 100.

Quant au I-MLP d'isolation, il possède 5 couches avec respectivement 200, 100, 100, 50 et 25 neurones, utilisant la fonction d'activation "tanh", un taux d'apprentissage constant de 1e-06, pour un nombre d'itérations maximal de 300.

La trajectoire circulaire ($trajectoire_2$) choisie pour illustrer les résultats est visualisée dans la Fig.4.25. Cette figure montre l'évolution de la trajectoire après détection de défauts pour chaque robot ainsi que le résidu de détection correspondant. La couleur rouge indique la détection d'un défaut dans le système. Ces défauts sont également notifiés sur le résidu de détection par une ligne verticale indiquant leur apparition pour chaque robot, certains ayant été détectés et d'autres ayant été manqués. L'exactitude (accuracy) de la détection pour cette trajectoire ($trajectoire_2$), comme le montre le tableau 4.10, est de 83,52 % pour le Robot 1, avec 12,9 % de vrais négatifs (détection manquée (md)) et 3,52 % de faux positifs (fausses alarmes (fa)), qui pourraient provenir de la véritable défaillance survenue lors de l'acquisition des données. Ces deux valeurs ne dépassent pas 1/5 de l'ensemble de la trajectoire. Dans ce tableau, le cas "$traj_i detect$" désigne le MLP de détection pour "$traj_i$". "$traj_i isol$" désigne le MLP d'isolation.

L'exactitude change en fonction de la forme de la trajectoire. Pour le MLP de détection, l'entraînement a donné une exactitude de 73,15 %. Quant au MLP d'isolation, l'entraînement a donné une exactitude de 66,524 %. Cela signifie

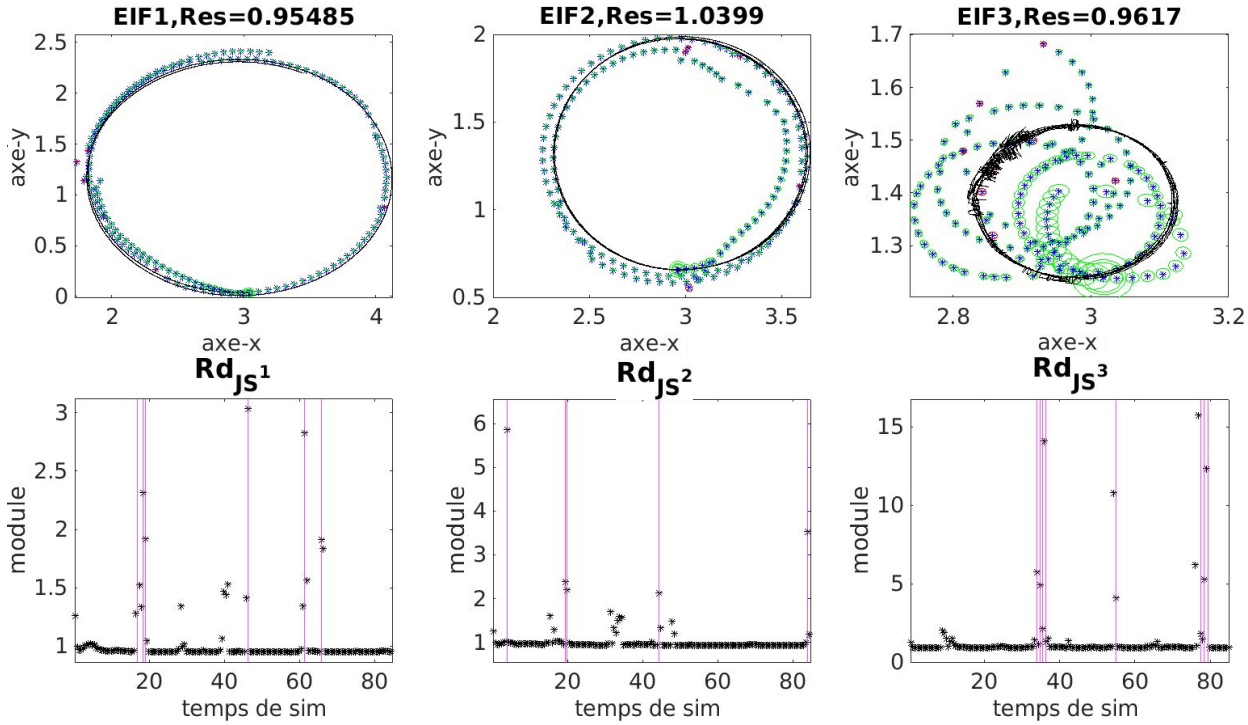


Figure 4.25 – Défauts détectés par le D-MLP de détection

que tous les défauts détectés ne sont pas isolés.

Pour les trajectoires circulaires {1,2}, les performances sont meilleures que pour les trajectoires transversales {4,5}. La troisième trajectoire est transversale puis circulaire (scénario carrefour), et elle présente la meilleure exactitude globale pour la détection et l'isolation. Dans les cas transversaux, certains robots ne sont pas en mesure de détecter ou d'isoler une grande majorité des défauts.

En ce qui concerne l'isolation, pour le capteur marvelmind, on observe 0,58 % (md), 8,23 % (fa) et une exactitude de 91,17 %. Pour le $LiDAR$, on observe 2,941 % (md), aucun (fa) et une exactitude de 97,05 %. Pour le gyroscope, on observe 7,05 % (md), 1,17 % (fa) et une exactitude de 91,76 %. Enfin, pour l'odométrie, on observe 3,5 % (md), aucun (fa) et une exactitude de 96,47 %, regroupés dans le tableau 4.9.

Table 4.9 – Résultats de l'isolation pour différents capteurs

Capteur	MD (%)	FA (%)	Exactitude (%)
Marvelmind	0,58	8,23	91,17
LiDAR	2,941	0	97,05
Gyroscope	7,05	1,17	91,76
Odométrie	3,5	0	96,47

Table 4.10 – Exactitude (*accuracy*) du MLP pour la détection et l'isolation

	$Robot_1$	$Robot_2$	$Robot_3$
Trajectoire	Exactitude	Exactitude	Exactitude
$traj_1$ D-MLP	0.6829	0.5975	0.6951
$traj_1$ I-MLP	0.5975	0.5121	0.6097
$traj_2$ D-MLP	0.8352	0.6882	0.7529
$traj_2$ I-MLP	0.7823	0.6764	0.7529
$traj_3$ D-MLP	0.8078	0.8374	0.7635
$traj_3$ I-MLP	0.7487	0.8128	0.7635
$traj_4$ D-MLP	0.5212	0.6063	0.7553
$traj_4$ I-MLP	0.4893	0.7340	0.6914
$traj_5$ D-MLP	0.7319	0.7010	0.6082
$traj_5$ I-MLP	0.4329	0.6288	0.5154

4.3.2.2 Résultat pour des modèles basés sur des arbres de décision

Une seconde implémentation est celle de l'arbre de décision et de forêts aléatoires. Ils partitionnent récursivement les données en sous-ensembles à chaque nœud, sur la base de la caractéristique qui sépare au mieux les classes selon le critère de division. Ce type de modèle permet d'expliquer pourquoi une certaine prédiction a été faite, en suivant les règles qui conduisent à une décision de classification particulière.

Compte tenus des hyper-paramètres, nous avons discuté dans la section 3.7.2.6 leur signification, et leur optimisation a produit les modèles suivants :

— L'optimisation de l'arbre de décision (D-DT) a abouti à une profondeur de 5, ce qui signifie qu'il a été configuré pour avoir 5 niveaux de partitionnement. Le critère utilisé pour séparer les différentes classes était le critère "gini". Lors de l'entraînement du modèle, il a atteint une exactitude (accuracy) de 72,612 %. Une visualisation illustrative de cet arbre peut être trouvée à la Fig.4.26, où la couleur orange indique la classe 0 sans défaut et la couleur bleue indique la classe 1 avec défaut. L'intensité de la couleur dans l'arbre indique la certitude de la classification par rapport aux étiquettes des échantillons de l'ensemble de données d'entraînement.
— Quant à l'isolation I-RF, elle utilise 8 estimateurs (arbres de décision) d'une profondeur maximale de 7, en utilisant également le critère *gini*, avec une exactitude (accuracy) d'entraînement de 65,657 %.

Ces D-DT/I-RF entraînés sont ensuite testés sur un scénario de défaillance pour chacune des trajectoires générées, et leur exactitude (accuracy) est présentée dans le Tableau 4.11.

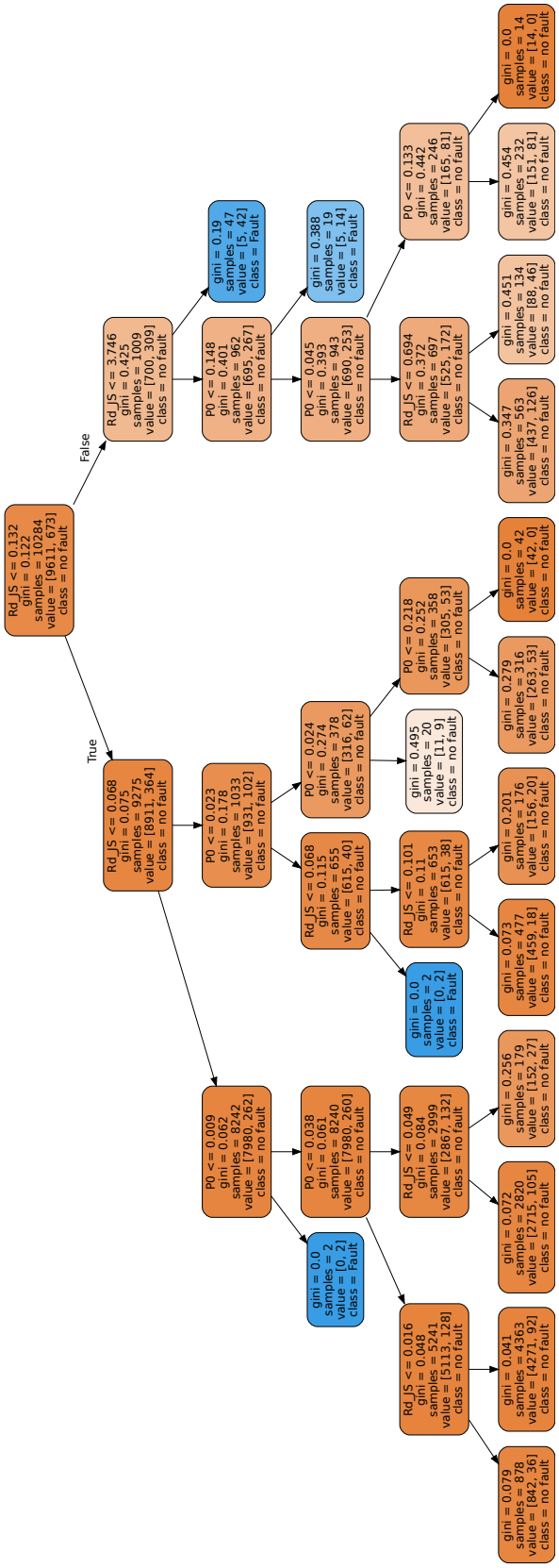


Figure 4.26 – Visualisation de l'arbre de décision de détection

Pour les trajectoires circulaires (c'est-à-dire traj{1,2}), la performance des modèles de classification est meilleure que pour les trajectoires transversales (traj{4,5}). La troisième trajectoire avec une partie transversale suivie d'une trajectoire circulaire (scénario du carrefour) présente la meilleure exactitude globale pour la détection et l'isolation par rapport aux autres. Dans les cas transversaux, certains robots sont capables de détecter mais pas d'isoler la grande majorité des défauts qui se produisent.

La trajectoire ($traj_2$) est visualisée dans la Fig.4.27. Cette figure montre l'évolution de la trajectoire pour chaque robot et le résidu de détection correspondant. La détection d'un défaut est notifiée en rouge, quant à la détection manquée entre la détection et l'isolation, elle est illustrée en magenta. Cela indique qu'un défaut a été détecté par l'I-RF en identifiant qu'au moins un capteur est en défaut, mais pas par le D-DT. Ces défauts sont également marqués sur le résidu de détection par une ligne verticale indiquant leur apparition et leur détection manquée pour chaque robot, en utilisant le même code couleur magenta.

Table 4.11 – Exactitude (*accuracy*) de l'arbre de décision et de la forêt aléatoire pour la détection et l'isolation

	Robot 1	**Robot 2**	**Robot 3**
Trajectoire	Exactitude	Exactitude	Exactitude
$traj_1$ D-DT	0.6585	0.7195	0.6341
$traj_1$ I-RF	0.4634	0.3658	0.5365
$traj_2$ D-DT	0.6588	0.7294	0.7470
$traj_2$ I-RF	0.6294	0.7235	0.7
$traj_3$ D-DT	0.8177	0.7192	0.7635
$traj_3$ I-RF	0.8177	0.7438	0.7586
$traj_4$ D-DT	0.6595	0.7340	0.6489
$traj_4$ I-RF	0.2446	0.5532	0.4361
$traj_5$ D-DT	0.6907	0.5154	0.5567
$traj_5$ I-RF	0.6391	0.4845	0.5876

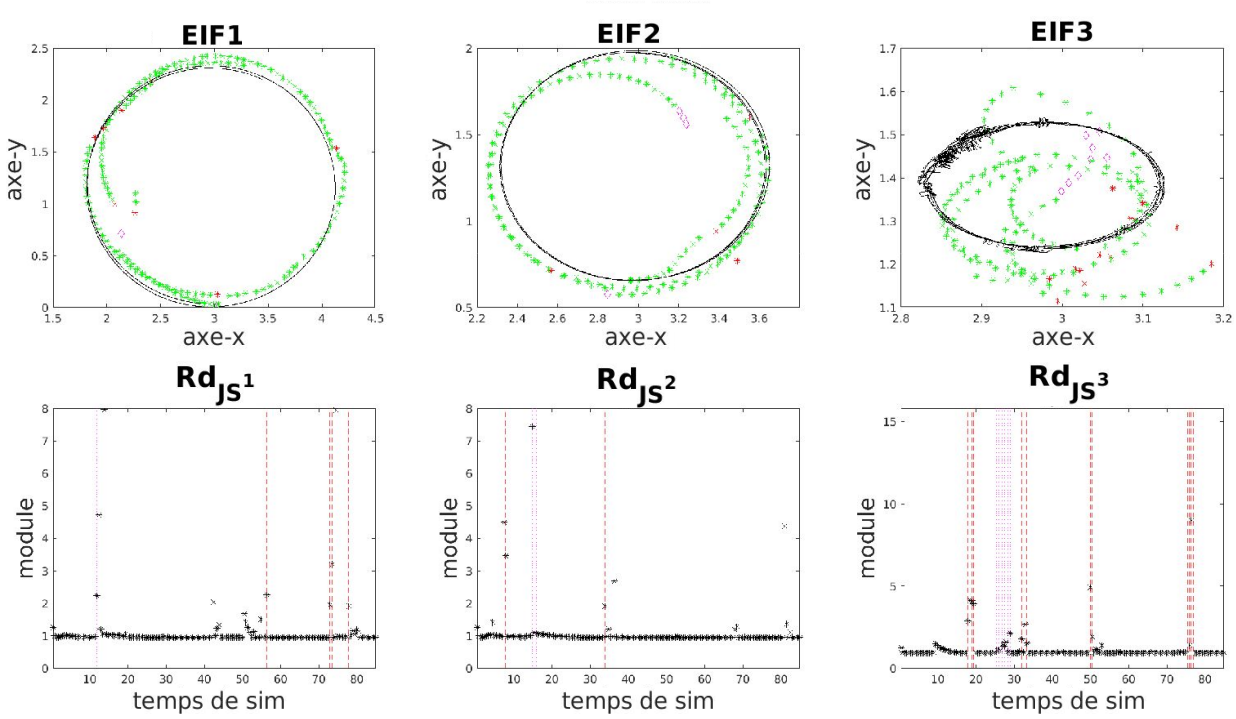


Figure 4.27 – Trajectoire montrant les défauts détectés à partir de la détection D-DT

Par exemple, pour le robot 1, ayant une exactitude de 65,88 % pour la détection, il n'y a pas eu de fausse alarme mais une détection manquée du défaut entre les deux modèles de détection et d'isolation. En ce qui concerne les capteurs, l'exactitude (accuracy) de l'I-RF pour l'odométrie est de 91,22 %, *marvelmind* 89,47 %, $LiDAR$ 2 est de 86,54 %, $LiDAR$ 3 est de 92,40 % et le *gyroscope* est de 95,90 %. Aucune fausse alarme n'a eu lieu sur ce robot. En ce qui concerne le robot 2, l'exactitude (accuracy) est de 72,94 %, avec une détection manquée de 25,88 % et une fausse alarme de 1,17 %. Pour le robot 3, l'exactitude (accuracy) est de 74,70 %, avec une détection manquée de 24,70 % et une fausse alarme de 0,58 %.

Ce modèle est plus performant que le modèle MLP. En effet, les arbres de décision sont intrinsèquement plus explicites, offrant une compréhension claire des règles de classification. De plus, les arbres de décision sont moins susceptibles de souffrir de sur-apprentissage lorsque les données sont limitées, contrairement aux MLP qui ont tendance à être plus complexes et à exiger davantage de données pour généraliser correctement.

4.3.2.3 Résultat pour des modèles basés sur la régression logistique

Une troisième implémentation est celle de la régression logistique. Comme montré dans la section 3.7.2.7, ce modèle cherche à ajuster une droite dans l'espace des logarithmes afin de classifier les données.

Pour le modèle de détection, ayant 2 variables d'entrée, le paramètre $\beta_1 = 0,1247$ est associé au résidu de détection Rd_{JS} et le paramètre $\beta_2 = -0,2036$

correspond à la probabilité a priori de l'hypothèse sans défaut P_0, ce qui signifie qu'elle est inversement proportionnelle à son évolution. Cependant, les paramètres de l'apprentissage fédéré sont $\beta_1 = 0,1355$ et $\beta_2 = -0,0804$, avec une valeur similaire de β_1 à celle de l'apprentissage centralisé, mais une valeur plus faible pour β_2, ce qui signifie que la relation entre P_0 et l'occurrence d'un défaut est moins forte dans le modèle d'apprentissage fédéré.

Pour l'isolation, comme montré précédemment, chaque capteur possède sa régression logistique et son modèle propre à lui. Pour le marvelmind, le modèle centralisé a une exactitude (accuracy) de $0,9611$ et un score F1 de $0,6680$. Pour le $robot_1$, l'exactitude (accuracy) est de $0,9606$ et le score F1 est de $0,6262$, le $robot_2$ a une précision de $0,9612$ et un score F1 de $0,6219$, et le $robot_3$ a une exactitude (accuracy) de $0,9619$ et un score F1 de $0,6418$. Les modèles ajustés localement ont des performances très similaires à celui du modèle centralisé, avec moins de temps consommé.

Concernant le *gyroscope*, le modèle centralisé ajusté possède une exactitude de $0,9770$ et un score F1 de $0,7554$. Pour le $robot_1$, la précision est de $0,9715$ et le score F1 est de $0,7154$, le $robot_2$ a une exactitude de $0,9793$ et un score F1 de $0,7837$, et pour le $robot_3$ l'exactitude est de $0,9836$ et le score F1 de $0,7896$. Les performances des modèles locaux sont également très proches de celles du modèle d'apprentissage centralisé.

Enfin, pour le $LiDAR$, le modèle centralisé ajusté possède une exactitude de $0,9834$ et un score F1 de $0,8010$. Pour le $robot_1$, l'exactitude est de $0,9805$ et le score F1 est de $0,7916$, le $robot_2$ a une exactitude de $0,9847$ et un score F1 de $0,8050$, et le $robot_3$ a une exactitude de $0,9825$ et un score F1 de $0,7835$. Les performances des modèles locaux sont également très proches de celles du modèle d'apprentissage centralisé.

L'ajustement des paramètres a utilisé une régularisation L2, en conservant toutes les entrées pour appliquer l'algorithme d'apprentissage fédéré. Leurs valeurs reflètent l'intérêt de l'entrée étant donné le capteur étudié : en cas de défaut sur le capteur d'encodeur, les 10 résidus d'isolation ont des valeurs proches et équilibrées, ce qui est attendu en raison du concept du schéma GOS sur lequel ces résidus ont été conçus. Le modèle odométrique, utilisé comme modèle de prédiction, apparaît dans chaque correction, ce qui explique que tous les indicateurs réagissent si une erreur se produit dans cette mesure.

En cas de défaut sur les autres capteurs utilisés pour la correction, on remarque que les variables concernant le capteur obtiennent des valeurs plus élevées que les autres. Ces variables sont la probabilité à priori d'hypothèse sans défaut P_0^{obs} et les résidus d'isolation Ri_{JS} suivant le schéma d'observateurs GOS.

Ce comportement est attendu, car les résidus ont été construits pour refléter et permettre la séparation et la distinction des défauts, dans le cadre d'observateurs généralisés GOS.

Par exemple, en cas de défaut sur le capteur marvelmind, le paramètre β_1 qui reflète le résidu met en œuvre toutes les corrections sauf celle du marvelmind. Ce paramètre a la valeur la plus basse parmi toutes les autres premières couches de résidus, en excluant un capteur à la fois. En revanche, le paramètre

β_{12} qui concerne la probabilité a priori d'hypothèse sans défaut du capteur lui-même a la valeur la plus élevée parmi tous les autres paramètres du même type pour les autres capteurs. Cela prouve que les défauts sont indépendants d'un capteur à l'autre. La même analogie est trouvée pour les autres capteurs également.

Pour l'un des scénarios testés, un tableau comparant le comportement des types de classificateurs pour la détection et l'isolation sous les deux techniques d'apprentissage proposées (centralisé/fédéré) est présenté dans le Tableau 4.12, ainsi que les modèles ajustés localement sur chaque robot. Ces performances sont présentées pour les 5 trajectoires utilisées, et pour le scénario de test 8.

L'indication "$traj_1D - Centr$" désigne la première trajectoire, "D" pour la détection, et "$Centr$" pour indiquer le modèle d'apprentissage centralisé. D'autre part, "$traj_1D - Fed$" désigne celui du modèle d'apprentissage fédéré. L'indication "$traj_1D - Robot$" signifie les modèles ajustés localement et testés localement sur les données de test du robot lui-même. Pour l'isolation, une moyenne des performances des modèles de capteurs est présentée. Ce tableau présente, comme discuté concernant les performances de l'ajustement, que les modèles fédérés et les modèles ajustés localement ont des performances égales ou meilleures que celles du modèle d'apprentissage centralisé. Nous remarquons que les performances sont similaires, voire meilleures dans l'apprentissage fédéré que dans le modèle local sur le robot ou dans le modèle centralisé.

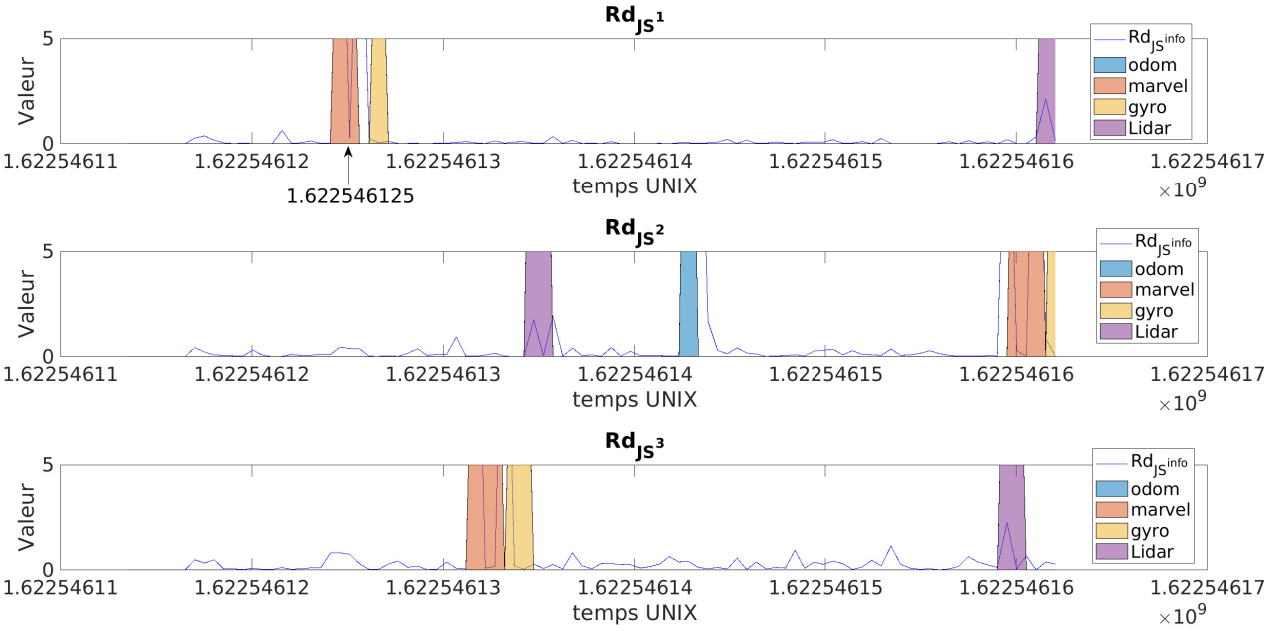


Figure 4.28 – Résidus de détection pour la $traj_5$ scénario = 8

En ce qui concerne l'étape de diagnostic, les résidus de détection pour chaque robot sont présentés dans la Fig.4.28. Nous avons choisi de montrer le scénario 8 pour $traj_5$, qui comporte des défauts simultanés et où des défauts d'odométrie ont été injectés. Pour $robot_1$, les défauts injectés sur le *marvel-*

mind et le *gyroscope* à l'instant Unix $1,622546125$ sont très proches. Cela se traduit par une augmentation de la valeur du résidu de détection qui réagit à tous les défauts. Pour le défaut de marvelmind, le défaut provoque un premier saut sur le résidu de détection Rd_{JS} lors de son application, diminue puis, puis fait un deuxième saut lorsque son défaut est désactivé. Ensuite, un défaut sur le *gyroscope* est appliqué, ce qui a moins d'impact sur les résidus de détection dans ce cas de trajectoire. La détection d'un défaut active l'étape d'isolation. Les résidus d'isolation Ri_{JS} sont conçus de manière à pouvoir isoler les défauts détectés. La première couche des résidus d'isolation (avec un capteur enlevé à chaque fois) est présentée dans la Fig.4.29, où leur nom contenant celui du capteur indique que toute observation sauf celle de ce capteur a été utilisée, en utilisant le schéma généralisé GOS d'observateurs.

Pour le résidu *marvelmind* pour le $robot_1$ (Fig.4.29 première ligne première colonne), il est sensible aux défauts de l'encodeur, des gyroscopes et du $LiDAR$. Le résidu du *gyroscope* (deuxième colonne) est sensible aux défauts de l'encodeur, de *marvelmind* et du $LiDAR$. Enfin, les résidus du $LiDAR$ (troisième et quatrième colonnes) sont sensibles aux défauts de l'encodeur, de *marvelmind* et du gyroscope. La même analogie peut être faite pour les deux autres robots.

Nous pouvons observer l'amélioration de la trajectoire dans la Figure 4.30, ainsi que l'impact de l'exclusion des défauts sur l'amélioration de la trajectoire. En comparant les capteurs exclus un par un entre l'apprentissage centralisé et l'apprentissage fédéré, nous avons obtenu les mêmes capteurs défectueux détectés et exclus exactement aux mêmes moments. Cela est attendu en raison de la performance très similaire entre l'apprentissage centralisé et l'apprentissage fédéré, comme discuté dans les résultats du Tableau 4.12.

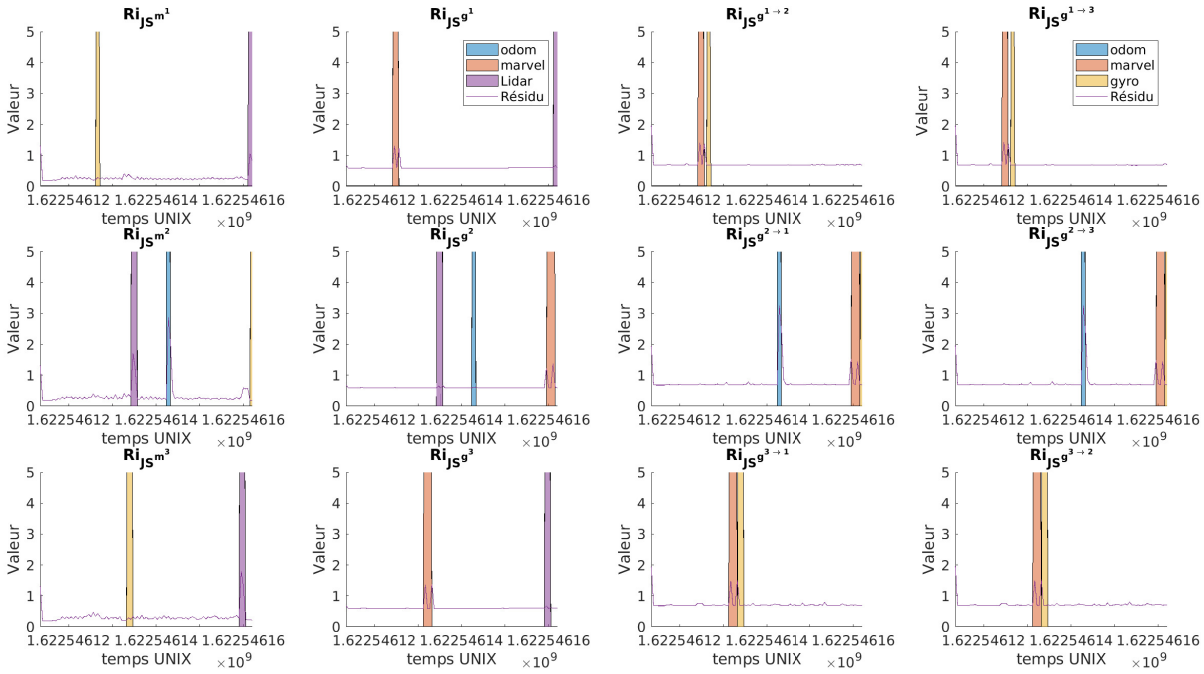
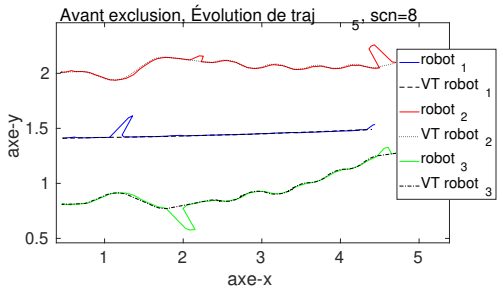
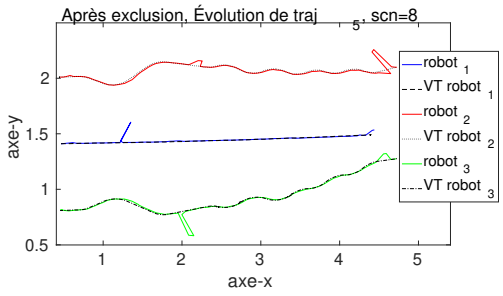


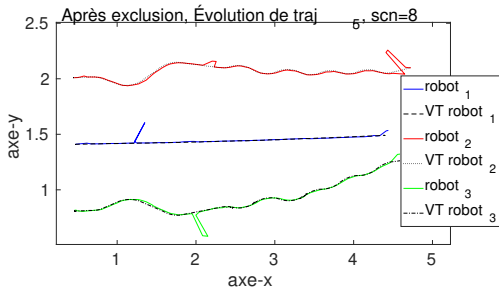
Figure 4.29 – Résidus d'isolation pour la $traj_5$ scénario = 8



(a) Trajectoire avant exclusion



(b) Trajectoire après exclusion avec le modèle centralisé



(c) Trajectoire après exclusion avec le modèle fédéré

Figure 4.30 – Évolution $Trajectoire_5$, présentant des performances identiques en exclusion entre l'apprentissage centralisé et fédéré pour cette trajectoire

Table 4.12 – Ratio de performance des modèles de détection et d'isolation en régression logistique du scénario 8

	Robot 1		Robot 2		Robot 3	
Trajectoire	Exactitude	f1-score	Exactitude	f1-score	Exactitude	f1-score
$traj_1D - Centr$	0.9250	0.6800	0.9500	0.6536	0.9250	0.6800
$traj_1D - Robot$	0.9125	0.6583	0.9500	0.6536	0.9375	0.7056
$traj_1D - Fed$	0.9250	0.6800	0.9500	0.6536	0.9250	0.6800
$traj_1I - Centr$	0.9625	0.7054	0.9843	0.7876	0.9624	0.6751
$traj_1I - Robot$	0.9593	0.6707	0.990	0.81	0.9718	0.7177
$traj_1I - Fed$	0.968	0.685	0.99	0.922	0.968	0.694
$traj_2D - Centr$	0.9638	0.7762	0.9457	0.6858	0.9457	0.7210
$traj_2D - Robot$	0.9698	0.8254	0.9457	0.6858	0.9457	0.7210
$traj_2D - Fed$	0.9638	0.7762	0.9457	0.6858	0.9457	0.7210
$traj_2I - Centr$	0.97138	0.6829	0.983	0.717	0.9713	0.691
$traj_2I - Robot$	0.9728	0.67	0.981	0.628	0.966	0.698
$traj_2I - Fed$	0.974	0.658	0.975	0.618	0.9743	0.6913
$traj_3D - Centr$	0.9504	0.7092	0.9504	0.7369	0.9257	0.5857
$traj_3D - Robot$	0.9455	0.6962	0.9504	0.7369	0.9356	0.6008
$traj_3D - Fed$	0.9504	0.7092	0.9504	0.7369	0.9207	0.5791
$traj_3I - Centr$	0.9789	0.619	0.974	0.607	0.9678	0.554
$traj_3I - Robot$	0.975	0.60	0.974	0.597	0.9777	0.577
$traj_3I - Fed$	0.977	0.619	0.97	0.606	0.971	0.576
$traj_4D - Centr$	0.9032	0.6278	0.9569	0.8218	0.9462	0.7081
$traj_4D - Robot$	0.9032	0.6278	0.9569	0.8218	0.9462	0.7080
$traj_4D - Fed$	0.9032	0.6278	0.9569	0.8218	0.9462	0.7080
$traj_4I - Centr$	0.954	0.687	0.973	0.76388	0.9677	0.6964
$traj_4I - Robot$	0.9596	0.7817	0.9811	0.7826	0.975	0.722
$traj_4I - Fed$	0.9596	0.739	0.9704	0.709	0.973	0.69
$traj_5D - Centr$	0.9484	0.63	0.9278	0.6626	0.9278	0.5922
$traj_5D - Robot$	0.9485	0.6295	0.9278	0.6626	0.9278	0.5922
$traj_5D - Fed$	0.9485	0.6294	0.9278	0.6626	0.9278	0.5921
$traj_5I - Centr$	0.987	0.788	0.966	0.708	0.971	0.653
$traj_5I - Robot$	0.987	0.796	0.966	0.605	0.9742	0.6541
$traj_5I - Fed$	0.984	0.7877	0.961	0.676	0.976	0.779

4.3.2.4 Comparaison de temps des algorithmes et taille des modèles

Dans cette section, nous faisons une comparaison des performances des modèles développés en termes de temps d'apprentissage et de la capacité mémoire allouée, ainsi que leur exactitude vis-à-vis des données. Pour cela, une base de données nettoyée de 10 scénarios a été générée, dont 8 sont utilisés pour l'entraînement et le reste pour le test. Cette base de données a subi un phase de pré-traitement, où les défauts sur les mesures lors de l'acquisition ont été récompensés.

Pour la détection, l'entraînement de l'arbre de décision prend $0,0316$ se-

condes pour l'arbre de décision de détection D-DT, avec une précision de $0,5292$ et une exactitude (accuracy) de $0,91316$. D'autre part, le D-MLP de détection prend $23,031$ secondes pour être entraîné, avec une précision de $0,658$ et une exactitude (accuracy) de $0,6988$.

Quant à la régression logistique, l'ajustement du modèle de détection prend $0,0762$ seconde pour l'apprentissage centralisé, alors que pour chaque robot seul, il prend environ $0,01$ seconde. Ces calculs sont effectués en parallèle, ce qui signifie que l'ajustement dans l'apprentissage fédéré est plus rapide que dans l'apprentissage centralisé. L'exactitude (accuracy) du modèle d'apprentissage centralisé est de $0,9463$, et le score f1 est de $0,7035$. Pour les modèles appris localement sur les robots, le modèle du $robot_1$ a une exactitude (accuracy) de $0,9489$ et un score f1 de $0,7189$, l'exactitude (accuracy) du $robot_2$ est de $0,9462$ et le score f1 est de $0,6144$, et le $robot_3$ a une exactitude (accuracy) de $0,9482$ et un score f1 de $0,6990$. On peut remarquer que l'exactitude (accuracy) de chaque modèle appris localement sur les robots est égale ou supérieure à celle du modèle d'apprentissage centralisé. En ce qui concerne le score f1, il est plus élevé dans le cas de l'apprentissage centralisé que dans la plupart des robots. Cela est dû au fait qu'on a accès à un ensemble de données plus vaste et plus diversifié, puisque les données de tous les clients sont combinées. Cela peut entraîner la création d'un ensemble de données d'entraînement plus représentatif et complet, permettant au modèle de classifier de manière plus efficace les cas avec et sans défaut. En combinant les modèles de chaque robot vers un seul en appliquant un apprentissage fédéré, l'exactitude (accuracy) du modèle de détection agrégé par rapport aux données d'entraînement est de $0,9830$, ce qui est plus élevé que celui de tout autre modèle entraîné localement sur les robots. Cela signifie que le modèle fédéré est plus précis pour classifier les instances dans l'ensemble de données. Cependant, malgré sa meilleure exactitude (accuracy), le score F1 du modèle fédéré est nettement inférieur au score F1 du modèle centralisé. Le score F1 prend en compte à la fois la précision et le rappel (recall), et un score F1 plus bas indique que le modèle fédéré n'équilibre pas bien les faux positifs et les faux négatifs.

Pour l'isolation, l'apprentissage de la forêt aléatoire I-RF prend $0,031604$ secondes, avec une précision de $0,82352$ et une exactitude de $0,95954$. Le I-MLP d'isolation prend $28,10667$ secondes pour être entraîné, avec une précision de $0,16938$ et une exactitude de $0,843737$. Les faibles valeurs de précision sont attendues en raison du déséquilibre des données où les cas de présence de défaut sont considérés comme rares. Quant à la régression logistique, l'étape d'isolation comporte un modèle pour chaque capteur, ayant tous la même entrée, avec une durée d'ajustement de $0,08$ seconde dans l'apprentissage centralisé et en moyenne de $0,03$ seconde sur chaque robot. Quant à l'exactitude, le modèle du capteur *marvelmind* centralisé possède une exactitude de 0.9607, fédéré de 0.9596, du $robot_1$ de 0.9608, $robot_2$ de 0.9614, du $robot_3$ de 0.9614 aussi. Pour le gyroscope, le modèle centralisé possède une exactitude de 0.97708, fédéré de 0.9822, du $robot_1$ de 0.9715, $robot_2$ de 0.9796, du $robot_3$ de 0.9834. Pour le $LiDAR$, le modèle centralisé possède une exactitude de 0.9842, fédéré de 0.9791, du $robot_1$ de 0.9805, $robot_2$ de 0.9841, du $robot_3$ de 0.9841 aussi. Enfin

pour l'odométrie, le modèle d'isolation centralisé possède une exactitude de 0.9690, fédéré de 0.9837, du $robot_1$ de 0.9814, $robot_2$ de 0.9834, du $robot_3$ de 0.9782. Ces valeurs sont regroupés dans le tableau 4.13.

Table 4.13 – Exactitude des capteurs et des modèles centralisés et fédérés

Capteur/Modèle	**Centralisé**	**Fédéré**	**Robot 1**	**Robot 2**	**Robot 3**
Marvelmind	0.9607	0.9596	0.9608	0.9614	0.9614
Gyroscope	0.97708	0.9822	0.9715	0.9796	0.9834
LiDAR	0.9842	0.9791	0.9805	0.9841	0.9841
Odométrie	0.9690	0.9837	0.9814	0.9834	0.9782

Table 4.14 – Performance d'entraînement des modèles d'apprentissage

Modèle	Tps d'entraîn.	Mémoire	Exactitude
D-DT	0.031604 sec	118.6 ko	0.91316
D-MLP	23.03099 sec	136.9 ko	0.6988
D-LR-center	0.0762 sec	849 o	0.9463
D-LR-Feder	0.01 sec	508 o	0.983
I-RF	0.0316 sec	237.7 ko	0.9595
I-MLP	28.1066 sec	962.7 ko	0.8437
I-LR-center	0.08 sec	2.2 ko	0.9727
I-LR-Feder	0.03 sec	1.4 ko	0.9761

En ce qui concerne l'allocation de mémoire, le D-DT occupe $118,6$ ko et le D-MLP occupe $562,9$ ko, tandis que le I-RF occupe $237,7$ ko et le I-MLP occupe $962,7$ ko. Cela est dû au fait qu'un arbre de décision/forêt aléatoire est un ensemble de règles avec des opérandes et des seuils, et ne nécessite pas autant d'espace que le MLP où les poids et les biais de chaque neurone sont enregistrés.

Quant à la régression logistique, les deux modèles occupent un espace minimal par rapport aux autres types d'apprentissages, le modèle centralisé nécessite 849 octets et le modèle fédéré n'utilise que 508 octets. D'autre part, les modèles d'isolation occupent ensemble un total de $2,2$ ko et $1,4$ ko pour l'apprentissage centralisé et l'apprentissage fédéré, respectivement.

Ces performances sont regroupées dans le Tableau 4.14.

4.3.3 Conclusion du chapitre

Dans ce chapitre, nous avons présenté les résultats de la méthode selon son évolution. Nous avons commencé par la méthode classique complètement à base de modèles, dont nous avons présenté les points fort et faible, et la limitation qu'elle avait qui a motivé le changement du paradigme de diagnostic. Ensuite, nous avons montré la première implémentation des méthodes guidées

par les données avec un perceptron multi-couche, où la problématique était d'essayer de modéliser des relations non linéaires complexes dans les données. Étant donné que les résultats n'étaient pas assez satisfaisants, que le modèle était lourd, et que le temps d'apprentissage du modèle était long, nous l'avons changé vers des arbres de décisions, en se fondant sur l'état de l'art des outils utilisés dans le diagnostic. Ces modèles sont explicables pour le problème étudié, avec un apprentissage par partitionnement récursif. Ces modèles avaient des performances meilleures que celles des MLP.

En étudiant l'approche d'apprentissage, et vu que nous cherchons à augmenter la sécurité du système, nous avons développé un apprentissage fédéré. Dans ce type d'apprentissage, les clients gardent leurs informations en local, appliquent l'apprentissage localement, et ne partagent que les derniers paramètres affinés. Dans une telle hypothèse, le partage des paramètres d'un MLP dense (celui qu'on a obtenu dans la section 3.7.2.5), il n'est pas possible d'utiliser ce type de modèle sous l'apprentissage fédéré, vu le grand nombre de paramètres à partager. D'autre part, la fédération de l'apprentissage par arbre de décision ne peut être faite qu'avec un système de vote, qui complique la classification et rend cette étape plus lourde. Donc, nous avons convergé vers une méthode, dont les paramètres sont connus à partir de la dimension de l'entrée, qui est la régression logistique. Ce type d'apprentissage avait un temps d'apprentissage plus rapide, avec une meilleure exactitude.

Chapitre 5

Conclusion et perspectives

5.1 Conclusion

Dans cette thèse, nous avons abordé le domaine de la fusion de données tout en tenant compte des éventuels défauts des capteurs. Dans un premier temps, nous avons présenté un état de l'art sur la localisation des véhicules autonomes, les systèmes multi-véhicules et la localisation coopérative. Puis nous avons présenté les méthodes de diagnostic à base de modèles et guidées par les données, puis la combinaison de ces deux méthodes (Chapitre 2).

Dans le chapitre 3, nous avons commencé par nous situer par rapport à l'état de l'art, puis nous avons formulé la problématique et les hypothèses de travail : création d'une solution de localisation coopérative dans le cadre d'une mobilité autonome et communicante, qui sera tolérante aux défauts capteurs. Nous avons ainsi commencé par décrire la conception des protocoles de communication de ce système, sa modélisation en évolution et en observation, et la manière avec laquelle l'estimation de la pose est réalisée. Dans ce travail, cette estimation de la pose nécessite l'usage d'un filtre capable de réaliser une fusion de données multi-capteurs. C'est pour cela que nous avons introduit l'usage de l'un des deux filtres : le filtre informationnel étendu EIF et le filtre d'intersection de covariances par lots B-CIF. Une fois qu'on a la prédiction et la correction, l'étape qui suit est celle de diagnostic, qui cherche à détecter et à localiser les défauts capteurs dans le système afin de les exclure de la phase de fusion de données. Étant donnée la nature stochastique des données, les indicateurs de défauts sont des divergences, donnant une mesure de dissimilarité entre les distributions de probabilités de la prédiction et la/les corrections. Ces indicateurs sont générés en deux étapes : une première étape de détection de défauts, avec un résidu de détection Rd_{JS} mesurant la dissimilarité entre la prédiction et la correction obtenue par la fusion de données multi-capteur. La seconde étape est celle d'isolation de défauts, où des observateurs suivant un schéma généralisé GOS sont conçus pour calculer des corrections qui fusionnent les données de tous les capteurs sauf un à la fois puis deux. La dissimilarité de ces corrections à la prédiction est calculée en générant des résidus d'isolation Ri_{JS}^{obs}. La décision de la détection de défaut est réalisée en évaluant les valeurs des résidus générés. Dans notre cas, nous avons présenté

deux approches différentes : la première basée sur des critères de la théorie d'information pour générer des seuils, tandis que la deuxième cherche à déduire à partir des données d'évolution des divergences les cas où ils indiquent la présence d'un défaut. Pour chaque modèle proposé, nous avons présenté ses étapes, ses hyper-paramètres et leur optimisation dans notre cas.

La plateforme d'expérimentation et les résultats sont ensuite présentés dans le chapitre 4. Nous avons présenté dans un premier temps la plateforme robotique sous le système d'exploitation ROS, les trois robots turtlebot3 avec les capteurs et le montage de la plateforme d'acquisition avec l'OptiTrack considéré comme une vérité terrain. Après avoir acquis les données des robots pour plusieurs trajectoires, nous passons à la création de la base de données de défauts capteurs, où nous avons présenté comment les scénarios de défauts sont créés et leurs injections sont réalisées. Par la suite, nous avons présenté les résultats de l'utilisation de cette base de données sur les approches proposées. Premièrement, l'approche classique complètement à base de modèles, ensuite pour les trois modèles : par perceptron multi-couche, arbre de décision/forêt aléatoire et régression logistique en deux types d'apprentissage : centralisé et décentralisé (fédéré). Nous présentons pour chacune des tableaux comparant l'exactitude des modèles pour un scénario de chaque trajectoire et pour chaque robot. Enfin, nous regroupons les performances des modèles dans un tableau comparant le temps d'entraînement, la mémoire occupée et l'exactitude (accuracy) du modèle.

5.2 Perspectives

Les performances obtenues avec trois robots communicants équipés de capteurs hétérogènes sont prometteuses.

Encouragées par ces résultats, plusieurs perspectives sont envisagées.

5.2.1 Aspect méthodologique

— **Extension de l'usage des capteurs :** Une piste potentielle consiste à incorporer des sources d'information supplémentaires pour la localisation, telles que la cartographie en 2D/3D, les systèmes de navigation inertielle (INS) et les données visuelles de caméra. L'intégration de ces diverses sources augmente la complexité du système. L'étude des limites de cette méthode et des types de capteurs avec lesquels elle pourrait être mise en œuvre constitue une autre orientation future.

— **Prise en considération de l'aspect évolutionnel des données avec des modèles récurrents :** Dans la méthode proposée, l'entrée du modèle d'apprentissage est les indicateurs de défauts basés sur des divergences informationnels, avec la probabilité préalable de l'hypothèse de non-défaut P_0. Le choix de ces entrées était pour permettre au modèle de classifier les cas de défaut tenant compte des valeurs de ces résidus de caractère évolutionnel, et qui dépendent de l'historique de fonctionnement

du système. Cette évolution et dépendance de l'historique peut être traitée autrement par l'établissement d'une mémoire dans l'apprentissage, c'est-à-dire en utilisant des modèles récurrents. Ces modèles peuvent diminuer le nombre de variables d'entrées et se limiter aux résidus seuls, en étudiant l'évolution de leurs valeurs au cours du temps.

— **Conception d'une méthode hybride de localisation multi-véhicules tolérante aux fautes :** La méthode de diagnostic qu'on propose dans ce travail combine le diagnostic à base de modèle et guidé par les données, pour décider la présence de défaut. Cette combinaison remplace l'étape de seuillage des indicateurs de défauts dans la méthode classique. L'étape suivante consiste d'étudier l'utilisation de l'apprentissage dans les différentes phases en amont de l'approche de localisation multi-véhicules tolérante aux fautes.

— **Évaluation des performances de la localisation coopérative :** Dans ce travail, nous nous sommes intéressés à la création de modèles de diagnostic afin de détecter et d'isoler les défauts des capteurs. Après avoir créée ces modèles, une étude plus exhaustive sur sa performance de en termes d'intégrité, de disponibilité, et de précision constitue l'étape suivante. Ceci est l'objectif du lot de travail (WP) 5 du projet LOCSP, lancé à la fin du deuxième semestre de l'année 2023. Son objectif étant d'identifier des indicateurs principaux de performance à analyser et à comparer lors de la mise au banc d'essai des algorithmes de positionnement.

5.2.2 Aspect applicatif

— **Tester sur des véhicules urbains :** L'étude a été réalisée sur des robots en milieu interne. L'équipe ToSyMA dispose de deux véhicules Renault Zoé robotisés fournissant des données capteurs sous format ROS. Une application de la méthode développée sur ces véhicules est alors réalisable à court terme (voir Fig.5.1). Dans ce cas, l'étude de coopération sera opportuniste.

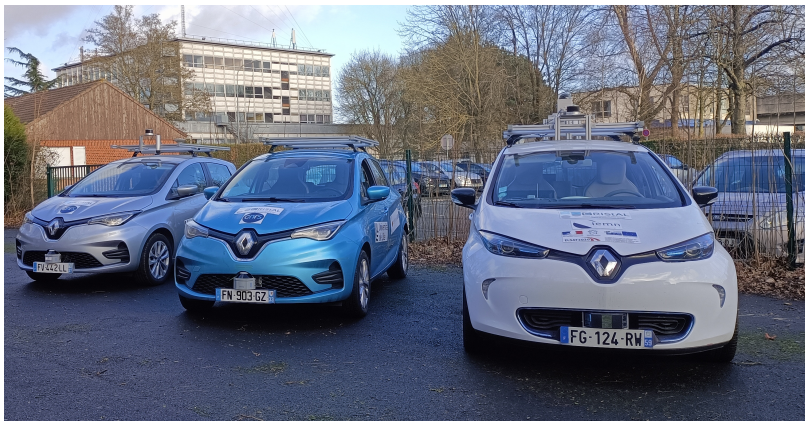


Figure 5.1 – Véhicules zoé de la plateforme PRETIL de CRIStAL

— **Jumeaux numériques des véhicules autonomes connectés :** La validation et le développement des algorithmes de navigation autonome nécessitent le parcours de plusieurs millions de kilomètres selon MATTHEW WOOD et al., 2019, ce qui n'est pas faisable avec des véhicules réels. Afin de parcourir cette distance, l'équipe ToSyMA développe un jumeau numérique de mobilités autonomes connectés (voir Fig.5.2), intégrés dans un environnement de simulation CARLA et associés à un émulateur de signaux GNSS StellaNGC de M3Systems et Skydel de Safran avec une interface matérielle en boucle fermée (HIL). Ce jumeau numérique permet de modéliser des variations météorologiques et des conditions routières changeantes, créant ainsi un environnement réaliste pour les expérimentations. De plus, il prend en charge la simulation de scénarios multi-véhicules, ce qui étend la portée des études et permet d'évaluer des systèmes de localisation dans des contextes plus complexes. De plus, le simulateur d'environnement offre une flexibilité considérable en termes de tests de divers capteurs, facilitant ainsi le développement et la validation de solutions de localisation basées sur une variété de capteurs. Enfin, l'aptitude à injecter des défauts réels sur les signaux GNSS et sur les capteurs simulés sans risquer d'endommager le vrai véhicule est d'une grande utilité pour l'extension de la méthode de localisation développée, permettant des tests rigoureux tout en préservant l'intégrité des équipements.

Figure 5.2 – Jumeau numérique avec Stella NGC HIL de M3Systems

Bibliographie

Abbas, T., K. Sjöberg, J. Karedal et F. Tufvesson (2015). « A Measurement Based Shadow Fading Model for Vehicle-to-Vehicle Network Simulations ». en. In : *International Journal of Antennas and Propagation* 2015, p. 1-12. doi : 10.1155/2015/190607.

Afshari, H., S. Gadsden et S. Habibi (2017). « Gaussian filters for parameter and state estimation : A general review of theory and recent trends ». In : *Signal Processing* 135, p. 218-238. doi : https://doi.org/10.1016/j.sigpro.2017.01.001.

Aït Tmazirte, N., M. El Badaoui El Najjar, C. Smaili et D. Pomorski (2012). « Multi-sensor data fusion based on information theory. Application to GNSS positionning and integrity monitoring ». In : *2012 15th International Conference on Information Fusion*, p. 743-749.

Al Hage, J. (oct. 2016). « Fusion de données tolérante aux défaillances : Application à la surveillance de l'intégrité d'un système de localisation ». Thèse de doctorat. Université de Lille.

Alsaade, F. W. et M. H. Al-Adhaileh (2023). « Cyber Attack Detection for Self-Driving Vehicle Networks Using Deep Autoencoder Algorithms ». In : *Sensors* 23.8. doi : 10.3390/s23084086.

Bahr, A., M. R. Walter et J. J. Leonard (2009). « Consistent cooperative localization ». In : *2009 IEEE International Conference on Robotics and Automation*, p. 3415-3422. doi : 10.1109/ROBOT.2009.5152859.

Balakrishnan, R. et K. Ranganathan (2012). *A textbook of graph theory*. eng. 2nd ed. 2012. Universitext. Springer New York Springer e-books Imprint : Springer Springer e-books. doi : 10.1007/978-1-4614-4529-6.

Basseville, M., I. V. Nikiforov et al. (1993). « Detection of abrupt changes : theory and application ». In : 104.

Bauer, S., Y. Alkhorshid et G. Wanielik (nov. 2016). « Using High-Definition maps for precise urban vehicle localization ». In : *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, p. 492-497. doi : 10.1109/ITSC.2016.7795600.

Bender, P., J. Ziegler et C. Stiller (juin 2014). « Lanelets : Efficient map representation for autonomous driving ». In : *2014 IEEE Intelligent Vehicles Symposium Proceedings*. MI, USA : IEEE, p. 420-425. doi : 10.1109/IVS.2014.6856487.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. 1re éd. Information Science and Statistics. Springer-Verlag New York 2006. Springer New York, NY.

Boussad, A. (déc. 2019). « Approche informationnelle pour la navigation autonome tolérante aux défauts : application aux systèmes robotiques mobiles ». Thèse de doctorat. Université de Lille.

Brambilla, M., E. Ferrante, M. Birattari et M. Dorigo (mars 2013). « Swarm robotics : a review from the swarm engineering perspective ». en. In : *Swarm Intelligence* 7.1, p. 1-41. doi : 10.1007/s11721-012-0075-2.

Brooks, R. A. (1991). In : *Artificial Intelligence* 47.1, p. 139-159. doi : https://doi.org/10.1016/0004-3702(91)90053-M.

Buckman, N., A. Hansen, S. Karaman et D. Rus (sept. 2022). « Evaluating Autonomous Urban Perception and Planning in a 1/10th Scale MiniCity ». en. In : *Sensors* 22.18, p. 6793. doi : 10.3390/s22186793.

Buehler, M., K. Iagnemma, S. Singh, B. Siciliano, O. Khatib et F. Groen, éd. (2007). *The 2005 DARPA Grand Challenge : The Great Robot Race*. en. T. 36. Springer Tracts in Advanced Robotics. Berlin, Heidelberg : Springer Berlin Heidelberg. doi : 10.1007/978-3-540-73429-1.

Cai, B., L. Huang et M. Xie (oct. 2017). « Bayesian Networks in Fault Diagnosis ». In : *IEEE Transactions on Industrial Informatics* 13.5, p. 2227-2240. doi : 10.1109/TII.2017.2695583.

Cai, B., Y. Liu, Z. Liu, Y. Chang et L. Jiang (2020). *Bayesian Networks for Reliability Engineering*. 1re éd. Springer Nature Singapore Pte Ltd. 2020. Springer Singapore, p. IX, 257. doi : https://doi.org/10.1007/978-981-13-6516-4.

Cao, Y. U., A. S. Fukunaga et A. Kahng (mars 1997). « Cooperative Mobile Robotics : Antecedents and Directions ». In : *Autonomous Robots* 4.1, p. 7-27. doi : 10.1023/A:1008855018923.

Cappelle, C., M. El Badaoui El Najjar, D. Pomorski et F. Charpillet (2007). « Localisation in urban environment using GPS and INS aided by monocular vision system and 3D geographical model ». In : *2007 IEEE Intelligent Vehicles Symposium*, p. 811-816. doi : 10.1109/IVS.2007.4290216.

Chen, C., B. Wang, C. X. Lu, N. Trigoni et A. Markham (2020). *A Survey on Deep Learning for Localization and Mapping : Towards the Age of Spatial Machine Intelligence*.

Chen, J. et R. J. Patton (1999). *Robust Model-Based Fault Diagnosis for Dynamic Systems*. 1re éd. The International Series on Asian Studies in Computer and Information Science. Springer Science+Business Media New York 1999. Springer New York, NY, p. XIX, 356. doi : https://doi.org/10.1007/978-1-4615-5149-2.

Chen, S., J. Yu et S. Wang (mars 2022). « One-dimensional convolutional neural network-based active feature extraction for fault detection and diagnosis of industrial processes and its understanding via visualization ». en. In : *ISA Transactions* 122, p. 424-443. doi : 10.1016/j.isatra.2021.04.042.

Chen, S. et V. Fremont (juill. 2021). « A Loosely Coupled Vision-LiDAR Odometry using Covariance Intersection Filtering ». In : *2021 IEEE Intelligent Vehicles Symposium (IV)*. Nagoya, Japan : IEEE, p. 1102-1107. doi : 10.1109/IV48863.2021.9575275.

Chen, Y. et R. Du (sept. 1998). « An Improved Artificial Neural Network Method for Monitoring and Diagnosis of Engineering Processes With Applications ». en. In : *Journal of Vibration and Control* 4.5, p. 635-650. doi : 10.1177/107754639800400506.

Christensen, A. L., R. O'Grady, M. Birattari et M. Dorigo (jan. 2008). « Fault detection in autonomous robots based on fault injection and learning ». en. In : *Autonomous Robots* 24.1, p. 49-67. doi : 10.1007/s10514-007-9060-9.

Christophe, C. (nov. 2001). « Surveillance des systèmes non linéaires : Application aux machines électriques ». Thèse de doctorat. Université de Lille.

Clark, R., S. Wang, H. Wen, A. Markham et N. Trigoni (2017). *VINet : Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem*.

Cox, D. R. (juill. 1958). « The Regression Analysis of Binary Sequences ». en. In : *Journal of the Royal Statistical Society : Series B (Methodological)* 20.2, p. 215-232. doi : 10.1111/j.2517-6161.1958.tb00292.x.

D'Addona, D. M. (2014). « Neural Network ». en. In : *CIRP Encyclopedia of Production Engineering*. Sous la dir. de The International Academy for Production Engineering, L. Laperrière et G. Reinhart. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 911-918. doi : 10.1007/978-3-642-20617-7_6563.

Dictionnaire de l'Académie Française (1992). Paris : Imprimerie Nationale.

Ding, S. X. (2013). *Model-Based Fault Diagnosis Techniques : Design Schemes, Algorithms and Tools*. en. Advances in Industrial Control. London : Springer London. doi : 10.1007/978-1-4471-4799-2.

Durrant-Whyte, H. et T. Bailey (2006). « Simultaneous localization and mapping : part I ». In : *IEEE Robotics & Automation Magazine* 13.2, p. 99-110. doi : 10.1109/MRA.2006.1638022.

Durrant-Whyte, H. (fév. 1988). « Uncertain geometry in robotics ». In : *IEEE Journal on Robotics and Automation* 4.1, p. 23-31. doi : 10.1109/56.768.

Durrant-Whyte, H. et T. C. Henderson (2008). « Multisensor Data Fusion ». In : *Springer Handbook of Robotics*. Sous la dir. de B. Siciliano et O. Khatib. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 585-610. doi : 10.1007/978-3-540-30301-5_26.

Durrant-Whyte, H., D. Rye et E. Nebot (1996). « Localization of Autonomous Guided Vehicles ». In : *Robotics Research*. Sous la dir. de G. Giralt et G. Hirzinger. London : Springer London, p. 613-625.

Fallah, A., A. Mokhtari et A. Ozdaglar (2020). « Personalized Federated Learning : A Meta-Learning Approach ». In.

Fang, Y., H. Min, W. Wang, Z. Xu et X. Zhao (2020). « A Fault Detection and Diagnosis System for Autonomous Vehicles Based on Hybrid Approaches ». In : *IEEE Sensors Journal*, p. 1-1. doi : 10.1109/JSEN.2020.2987841.

Ferber, J. (1995). *Les systèmes multi-agents : Vers une intelligence collective*. Elsevier-Masson, p. 544.

Fox, D., W. Burgard, H. Kruppa et S. Thrun (1999). *Collaborative Multi-Robot Localization*. Sous la dir. de W. Förstner, J. M. Buhmann, A. Faber et P. Faber. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 15-26.

Frank, P. M. (mai 1990). « Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy ». en. In : *Automatica* 26.3, p. 459-474. doi : 10.1016/0005-1098(90)90018-D.

Füssel, D. (2002). « Fault diagnosis with tree structured neuro-fuzzy systems ». In.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Second edition. Beijing [China] ; Sebastopol, CA : O'Reilly Media, Inc.

Goel, P., G. Dedeoglu, S. Roumeliotis et G. Sukhatme (2000). « Fault detection and identification in a mobile robot using multiple model estimation and neural network ». In : *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. T. 3. San Francisco, CA, USA : IEEE, p. 2302-2309. doi : 10.1109/ROBOT.2000.846370.

Gonzalez-de-Santos, P., A. Ribeiro, C. Fernandez-Quintanilla, F. Lopez-Granados, M. Brandstoetter, S. Tomic et al. (août 2017). « Fleets of robots for environmentally-safe pest control in agriculture ». en. In : *Precision Agriculture* 18.4, p. 574-614. doi : 10.1007/s11119-016-9476-3.

Goodfellow, I., Y. Bengio et A. Courville (2016). *Deep learning*. Adaptive computation and machine learning. Cambridge, Massachusetts : The MIT Press.

Gordon, D., D. Fox et A. Farhadi (2019). *What Should I Do Now ? Marrying Reinforcement Learning and Symbolic Planning*.

Gordon, N., D. Salmond et A. Smith (1993). « Novel approach to nonlinear/non-Gaussian Bayesian state estimation ». en. In : *IEE Proceedings F Radar and Signal Processing* 140.2, p. 107. doi : 10.1049/ip-f-2.1993.0015.

Grabowski, R., L. E. Navarro-Serment, C. J. Paredis et P. K. Khosla (juin 2000). « Heterogeneous Teams of Modular Robots for Mapping and Exploration ». In : t. 8. 3, p. 293-308. doi : 10.1023/A:1008933826411.

Grime, S. et H. Durrant-Whyte (1994). « Data fusion in decentralized sensor networks ». In : *Control Engineering Practice* 2.5, p. 849-863. doi : https://doi.org/10.1016/0967-0661(94)90349-2.

Guizzo, E. (juill. 2008). « Three Engineers, Hundreds of Robots, One Warehouse ». In : *IEEE Spectrum* 45.7. Conference Name : IEEE Spectrum, p. 26-34. doi : 10.1109/MSPEC.2008.4547508.

Guo, Q., Y. Li, Y. Song, D. Wang et W. Chen (mars 2020). « Intelligent Fault Diagnosis Method Based on Full 1-D Convolutional Generative Adversarial Network ». In : *IEEE Transactions on Industrial Informatics* 16.3, p. 2044-2053. doi : 10.1109/TII.2019.2934901.

HAARNOJA, T., A. AJAY, S. LEVINE et P. ABBEEL (2017). *Backprop KF : Learning Discriminative Deterministic State Estimators*.

HANDSCHIN, J. E. et D. Q. MAYNE (1969). « Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering† ». In : *International Journal of Control* 9.5, p. 547-559. DOI : 10.1080/00207176908905777.

HARBAOUI, N. (déc. 2022). « Diagnostic adaptatif à l'environnement de navigation : Apport de l'apprentissage profond pour une localisation sûre et précise ». Thèse de doctorat. Université de Lille.

HÉRY, E., XU, P. AND BONNIFAIT, P. (fév. 2021). « Consistent Decentralized Cooperative Localization for Autonomous Vehicles using LiDAR, GNSS and HD maps ». In : DOI : 10.1002/rob.22004.

HOWARD, A., M. MATARIC et G. SUKHATME (2003). « Putting the 'I' in 'team' : an ego-centric approach to cooperative localization ». In : *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. T. 1, 868-874 vol.1. DOI : 10.1109/ROBOT.2003.1241702.

HOWARD, A., M. MATARK et G. SUKHATME (2002). « Localization for mobile robot teams using maximum likelihood estimation ». In : *IEEE/RSJ International Conference on Intelligent Robots and Systems*. T. 1, 434-439 vol.1. DOI : 10.1109/IRDS.2002.1041428.

HUANG, Y., G. SUN, J. TAO, Y. HU et L. YUAN (août 2022). « A modified fusion model-based/data-driven model for sensor fault diagnosis and performance degradation estimation of aero-engine ». In : *Measurement Science and Technology* 33.8, p. 085105. DOI : 10.1088/1361-6501/ac6081.

IFAC TECHNICAL COMMITTEE ON CONTROL EDUCATION (2006). *Terminology in the area of fault management*. https://tc.ifac-control.org/6/4/terminology/terminology-in-the-area-of-fault-management. Accessed : 27 July 2023.

ISERMANN, R. (2006). *Fault-Diagnosis Systems. An Introduction from Fault Detection to Fault Tolerance*. 1re éd. Springer-Verlag Berlin Heidelberg 2006. Springer Berlin, Heidelberg, p. XVIII, 475. DOI : https://doi.org/10.1007/3-540-30368-5.

Dedicated Short Range Communications (DSRC) Performance Requirements for V2V Safety Awareness (oct. 2018). Standard. Society of Automotive Engineering.

JIN, D., F. YIN, C. FRITSCHE, A. M. ZOUBIR et F. GUSTAFSSON (mars 2016). « Cooperative localization based on severely quantized RSS measurements in wireless sensor network ». In : *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN : 2379-190X, p. 4214-4218. DOI : 10.1109/ICASSP.2016.7472471.

JIN, H., Z. ZUO, Y. WANG, L. CUI et L. LI (déc. 2022). « An Integrated Model-Based and Data-Driven Gap Metric Method for Fault Detection and Isolation ». In : *IEEE Transactions on Cybernetics* 52.12, p. 12687-12697. DOI : 10.1109/TCYB.2021.3086193.

JING W. (1994). « On sign-board based inter-robot communication in distributed robotic systems ». en. In : *Proceedings of the 1994 IEEE International*

Conference on Robotics and Automation. San Diego, CA, USA : IEEE Comput. Soc. Press, p. 1045-1050. DOI : 10.1109/ROBOT.1994.351219.

JONSCHKOWSKI, R., D. RASTOGI et O. BROCK (2018). *Differentiable Particle Filters : End-to-End Learning with Algorithmic Priors*.

JULIER, S. J., H. F. DURRANT-WHYTE et S. B. COOPER (1998). « Localization system for a high-speed land vehicle ». eng. In : *Proceedings of SPIE*. T. 3210. ISSN : 0277-786X Issue : 1. SPIE, p. 54-65.

JULIER, S. et UHLMANN, J. K. (2009). « General Decentralized Data Fusion with Covariance Intersection (CI) ». In : *Multisensor Data Fusion*. Sous la dir. de DAVID HALL et JAMES LLINAS. T. 3. CRC Press, p. 319-344.

JUTTEN, C. (2007). *Détection, Estimation, Information - Polytech' Grenoble : Université Joseph Fourier*.

JWO, D.-J. et H.-C. HUANG (sept. 2004). « Neural Network Aided Adaptive Extended Kalman Filtering Approach for DGPS Positioning ». en. In : *Journal of Navigation* 57.3, p. 449-463. DOI : 10.1017/S0373463304002814.

KALMAN, R. (1960). « A new approach to linear filtering and prediction problems ». In : *Transaction of the ASME journal of basic*.

KALMAN, R. E. et R. S. BUCY (mars 1961). « New Results in Linear Filtering and Prediction Theory ». en. In : *Journal of Basic Engineering* 83.1, p. 95-108. DOI : 10.1115/1.3658902.

KARAM, N. (déc. 2009). « Agrégation de données décentralisées pour la localisation multi-véhicules ». Issue : 62T1071932HJQ. Thèse de doctorat. Université Blaise Pascal - Clermont-Ferrand II.

KARAM, N., F. CHAUSSE, R. AUFRERE et R. CHAPUIS (juin 2006). « Cooperative Multi-Vehicle Localization ». In : *IEEE Intelligent Vehicles Symposium*, p. 564-570. DOI : 10.1109/IVS.2006.1689688.

KATO, S., S. TSUGAWA, K. TOKUDA, T. MATSUI et H. FUJII (sept. 2002). « Vehicle control algorithms for cooperative driving with automated vehicles and intervehicle communications ». en. In : *IEEE Transactions on Intelligent Transportation Systems* 3.3, p. 155-161. DOI : 10.1109/TITS.2002.802929.

KENDALL, A., M. GRIMES et R. CIPOLLA (déc. 2015). « PoseNet : A Convolutional Network for Real-Time 6-DOF Camera Relocalization ». In : *2015 IEEE International Conference on Computer Vision (ICCV)*. ISSN : 2380-7504, p. 2938-2946. DOI : 10.1109/ICCV.2015.336.

KURAZUME, R., S. NAGATA et S. HIROSE (1994). « Cooperative positioning with multiple robots ». In : *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1250-1257 vol.2. DOI : 10.1109/ROBOT.1994.351315.

LASSOUED, K. (juill. 2016). « Localisation de robots mobiles en coopération mutuelle par observation d'état distribuée ». Thèse de doctorat. Université de Technologie de Compiègne.

LASSOUED, K., P. BONNIFAIT et I. FANTONI (juin 2016). « Cooperative localization of vehicles sharing GNSS pseudoranges corrections with no base station using set inversion ». In : *2016 IEEE Intelligent Vehicles Sym-*

posium (IV). Gotenburg, Sweden : IEEE, p. 496-501. DOI : 10.1109/IVS.2016.7535432.

LE PAPE, C. (mai 1990). « A combination of centralized and distributed methods for multi-agent planning and scheduling ». In : *, IEEE International Conference on Robotics and Automation Proceedings*, 488-493 vol.1. DOI : 10.1109/ROBOT.1990.126026.

LEE, D. (2008). « Nonlinear Estimation and Multiple Sensor Fusion Using Unscented Information Filtering ». In : *Signal Processing Letters, IEEE* 15, p. 861-864. DOI : 10.1109/LSP.2008.2005447.

LEE, T. B. (août 2023). « Driverless Cars May Already Be Safer Than Human Drivers ». In : *Understanding AI,* DOI : https://www.understandingai.org/p/driverless-cars-may-already-be-safer?utm_campaign=post&utm_medium=web.

LEONARD, J. J. et H. F. DURRANT-WHYTE (1992). « Model-based Localization ». eng. In : *Directed Sonar Sensing for Mobile Robot Navigation*. The Springer International Series in Engineering and Computer Science. ISSN : 0893-3405. Boston, MA : Springer US, p. 51-95.

LI, L., Y. MA, K. TANG, X. ZHAO, C. CHEN, J. HUANG et al. (août 2023). « Geo-Localization With Transformer-Based 2D-3D Match Network ». In : *IEEE Robotics and Automation Letters* 8.8, p. 4855-4862. DOI : 10.1109/LRA.2023.3290526.

LI, Q., S. CHEN, C. WANG, X. LI, C. WEN, M. CHENG et al. (2020). *LO-Net : Deep Real-time Lidar Odometry*. en. DOI : http://arxiv.org/abs/1904.08242.

LI, T., A. K. SAHU, M. ZAHEER, M. SANJABI, A. TALWALKAR et V. SMITH (2020). « Federated Optimization in Heterogeneous Networks ». In : DOI : 10.48550/ARXIV.1812.06127.

LI, T., M. SANJABI, A. BEIRAMI et V. SMITH (2019). « Fair Resource Allocation in Federated Learning ». In : Publisher : arXiv Version Number : 2. DOI : 10.48550/ARXIV.1905.10497.

LI, H. et NASHASHIBI, F. (avr. 2013). « Cooperative Multi-Vehicle Localization Using Split Covariance Intersection Filter ». In : *IEEE Intelligent Transportation Systems Magazine* 5.2, p. 33-44. DOI : 10.1109/MITS.2012.2232967.

LIM, K. et K. M. TULADHAR (jan. 2019). « LIDAR : Lidar Information based Dynamic V2V Authentication for Roadside Infrastructure-less Vehicular Networks ». en. In : *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. Las Vegas, NV, USA : IEEE, p. 1-6. DOI : 10.1109/CCNC.2019.8651684.

MADHAVAN, R., K. FREGENE et L. PARKER (mai 2002). « Distributed heterogeneous outdoor multi-robot localization ». In : *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. T. 1, 374-381 vol.1. DOI : 10.1109/ROBOT.2002.1013389.

MAKKAWI, K. (déc. 2020). « An adaptive fault tolerant fusion framework for a precise, available and fail-safe localization ». Thèse de doctorat. Université de Lille.

MARTINELLI, A., F. PONT et R. SIEGWART (2005). « Multi-Robot Localization Using Relative Observations ». In : *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, p. 2797-2802. DOI : 10.1109/ROBOT.2005.1570537.

MATHEWS, N., A. L. CHRISTENSEN, R. O'GRADY, F. MONDADA et M. DORIGO (sept. 2017). « Mergeable nervous systems for robots ». en. In : *Nature Communications* 8.1, p. 439. DOI : 10.1038/s41467-017-00109-2.

MATSUMOTO, A., H. ASAMA, Y. ISHIDA, K. OZAKI et I. ENDO (juill. 1990). « Communication in the autonomous and decentralized robot system ACTRESS ». In : *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, 835-840 vol.2. DOI : 10.1109/IROS.1990.262503.

MATTHEW WOOD, PHILIPP ROBBEL, MICHEAL MAASS et RADBOUD DUINTJER TEBBENS (juill. 2019). « Safety First for Automated driving, »

MCFARLAND, D. (1994). « Towards Robot Cooperation ». In : *Proceedings of the Simulation of Adaptive Behavior*.

MCLAUGHLIN, S., R. EVANS et V. KRISHNAMURTHY (2003). « Data incest removal in a survivable estimation fusion architecture ». In : *Sixth International Conference of Information Fusion, 2003. Proceedings of the*. Cairns, Queensland, Australia : IEEE, p. 229-236. DOI : 10.1109/ICIF.2003.177451.

MCMAHAN, H. B., E. MOORE, D. RAMAGE, S. HAMPSON et B. A. y. ARCAS (2016). « Communication-Efficient Learning of Deep Networks from Decentralized Data ». In : Publisher : arXiv Version Number : 3. DOI : 10.48550/ARXIV.1602.05629.

NATIONAL HIGHWAY TRAFFIC SAFETY ADMINISTRATION, U.S. DEPARTMENT OF TRANSPORTATION (2020). *Traffic Safety Facts 2020*. Rapp. tech.

PATTON, R., C. LOPEZ-TORIBIO et F. UPPAL (avr. 1999). « Artificial intelligence approaches to fault diagnosis ». In : *IEE Colloquium on Condition Monitoring : Machinery, External Structures and Health (Ref. No. 1999/034)*, p. 5/1-518. DOI : 10.1049/ic:19990188.

PAULOS, J., S. W. CHEN, D. SHISHIKA et V. KUMAR (2019). *Decentralization of Multiagent Policies by Learning What to Communicate*. DOI : 10.1109/ICRA.2019.8793777.

PETIT, S. (oct. 2017). « World Vehicle Population Rose 4.6% in 2016 ». In : *Wards Intelligence*. DOI : https://wardsintelligence.informa.com/WI058630/World-Vehicle-Population-Rose-46-in-2016.

PIERRE, C., R. CHAPUIS, R. AUFRÈRE, J. LANEURIT et C. DEBAIN (juill. 2018). « Range-Only Based Cooperative Localization for Mobile Robots ». In : *2018 21st International Conference on Information Fusion (FUSION)*, p. 1933-1939. DOI : 10.23919/ICIF.2018.8455692.

PONTE MÜLLER, F. de, E. M. DIAZ et I. RASHDAN (juin 2016). « Cooperative positioning and radar sensor fusion for relative localization of vehicles ». In : *2016 IEEE Intelligent Vehicles Symposium (IV)*, p. 1060-1065. DOI : 10.1109/IVS.2016.7535520.

Reina, G., J. Underwood, G. Brooker et H. Durrant-Whyte (2011). « Radar-based perception for autonomous outdoor vehicles ». eng. In : *Journal of field robotics* 28. Place : Hoboken Publisher : Wiley Subscription Services, Inc., A Wiley Company, p. 894-913.

Rekleitis, Ioannis M., Dudek, G. et Milios, E.E. (oct. 2002). « Multi-robot cooperative localization : a study of trade-offs between efficiency and accuracy ». English. In : t. 3. Lausanne, Switzerland : IEEE, p. 2690-2695. doi : https://doi.org/10.1109/IRDS.2002.1041676.

Revach, G., N. Shlezinger, X. Ni, A. L. Escoriza, R. J. G. Van Sloun et Y. C. Eldar (2022). « KalmanNet : Neural Network Aided Kalman Filtering for Partially Known Dynamics ». In : *IEEE Transactions on Signal Processing* 70, p. 1532-1547. doi : 10.1109/TSP.2022.3158588.

Rohani, M., D. Gingras et D. Gruyer (nov. 2014). « Dynamic base station DGPS for cooperative vehicle localization ». In : *2014 International Conference on Connected Vehicles and Expo (ICCVE)*. ISSN : 2378-1297, p. 781-785. doi : 10.1109/ICCVE.2014.7297658.

Rossi, F., S. Bandyopadhyay, M. Wolf et M. Pavone (2018). « Review of Multi-Agent Algorithms for Collective Behavior : a Structural Taxonomy ». en. In : *IFAC-PapersOnLine* 51.12, p. 112-117. doi : 10.1016/j.ifacol.2018.07.097.

Rouček, T., M. Pecka, P. Čížek, T. Petříček, J. Bayer, V. Šalanský et al. (mars 2022). « System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA Subterranean Challenge ». en. In : *Field Robotics* 2.1, p. 1779-1818. doi : 10.55417/fr.2022055.

Roumeliotis, S. et G. Bekey (oct. 2002). « Distributed multirobot localization ». en. In : *IEEE Transactions on Robotics and Automation* 18.5, p. 781-795. doi : 10.1109/TRA.2002.803461.

Roumeliotis, S. I. et I. M. Rekleitis (juill. 2004). « Propagation of Uncertainty in Cooperative Multirobot Localization : Analysis and Experimental Results ». en. In : *Autonomous Robots* 17.1, p. 41-54. doi : 10.1023/B:AURO.0000032937.98087.91.

Russell, S. J. (J. (2010). *Artificial Intelligence : A Modern Approach*. Upper Saddle River, N.J. : Prentice Hall.

Sammut, C. et G. I. Webb, éd. (2010). *Encyclopedia of Machine Learning*. en. Boston, MA : Springer US. doi : 10.1007/978-0-387-30164-8.

Sanderson, A. C. (jan. 1997). « A distributed algorithm for cooperative navigation among multiple mobile robots ». en. In : *Advanced Robotics* 12.4, p. 335-349. doi : 10.1163/156855398X00235.

Shi, Q. et H. Zhang (juill. 2021). « Fault Diagnosis of an Autonomous Vehicle With an Improved SVM Algorithm Subject to Unbalanced Datasets ». In : *IEEE Transactions on Industrial Electronics* 68.7, p. 6248-6256. doi : 10.1109/TIE.2020.2994868.

Singh, K. et K. Fujimura (mai 1993). « Map making by cooperating mobile robots ». In : *[1993] Proceedings IEEE International Conference on Robotics and Automation*, 254-259 vol.2. doi : 10.1109/ROBOT.1993.292155.

Sobhani-Tehrani, E. et K. Khorasani (2009). *Fault Diagnosis of Nonlinear Systems Using a Hybrid Approach*. en. T. 383. Lecture Notes in Control and Information Sciences. Boston, MA : Springer US. doi : 10.1007/978-0-387-92907-1.

Soysal, O. et E. Sahin (2005). « Probabilistic aggregation strategies in swarm robotic systems ». In : *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.* Pasadena, CA, USA : IEEE, p. 325-332. doi : 10.1109/SIS.2005.1501639.

Sutton, R. S. et A. Barto (1998). *Reinforcement learning : an introduction*. eng. Second edition. Adaptive computation and machine learning. Cambridge, Massachusetts London, England : The MIT Press.

Szafarczyk, M., éd. (1994). *Automatic Supervision in Manufacturing*. Advanced Manufacturing. Springer-Verlag London Limited 1994. Springer London, p. XIV, 283. doi : https://doi.org/10.1007/978-1-4471-3458-9.

Thrun, S. (2001). « A Probabilistic On-Line Mapping Algorithm for Teams of Mobile Robots ». eng. In : *The International journal of robotics research* 20.5. Place : London Publisher : SAGE Publications, p. 335-363.

Thrun, S., W. Burgard et D. Fox (1998). « A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots ». eng. In : *Machine learning* 31.1-3, p. 29-53.

— (2005). *Probabilistic robotics*. Intelligent robotics and autonomous agents. OCLC : ocm58451645. Cambridge, Mass : MIT Press.

Tsugawa, S., S. Kato, T. Matsui, H. Naganawa et H. Fujii (2000). « An architecture for cooperative driving of automated vehicles ». In : *2000 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.00TH8493)*. Dearborn, MI, USA : IEEE, p. 422-427. doi : 10.1109/ITSC.2000.881102.

Turing, A. M. (1950). « Computing Machinery and Intelligence ». eng. In : *Mind* 59.236. Place : London, etc Publisher : Thomas Nelson and Son, Ltd, p. 433-460.

Upcroft, B., B. Douillard, T. Kaupp, M. Ridley, M. Ridley, L.-L. Ong et al. (2006). « Non-Gaussian State Estimation in an Outdoor Decentralised Sensor Network ». eng. In : *Proceedings of the 45th IEEE Conference on Decision and Control*. ISSN : 0191-2216. IEEE, p. 366-372.

Valada, A., N. Radwan et W. Burgard (mai 2018). « Deep Auxiliary Learning for Visual Localization and Odometry ». In : *2018 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN : 2577-087X, p. 6939-6946. doi : 10.1109/ICRA.2018.8462979.

Wan, E. et R. Van Der Merwe (2000). « The unscented Kalman filter for nonlinear estimation ». In : *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. Lake Louise, Alta., Canada : IEEE, p. 153-158. doi : 10.1109/ASSPCC.2000.882463.

Wang, C., Wen Shang et D. Sun (août 2009). « A neural network approach to monitoring robot malfunction in multirobot formation control tasks ». In : *2009 International Conference on Mechatronics and Automation*.

Changchun, China : IEEE, p. 2689-2694. DOI : 10.1109/ICMA.2009.5244944.

WANG, D., H. QI, B. LIAN, Y. LIU et H. SONG (2023). « Resilient Decentralized Cooperative Localization for Multisource Multirobot System ». en. In : *IEEE Transactions on Instrumentation and Measurement* 72, p. 1-13. DOI : 10.1109/TIM.2023.3291805.

WANG, S., R. CLARK, H. WEN et N. TRIGONI (mai 2017). « DeepVO : Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks ». en. In : *2017 IEEE International Conference on Robotics and Automation (ICRA)*. arXiv :1709.08429 [cs], p. 2043-2050. DOI : 10.1109/ICRA.2017.7989236.

WENG, K.-C., S.-T. LIN, C.-C. HU, R.-T. SOONG et M.-T. CHI (nov. 2022). « Multi-view approach for drone light show ». en. In : *The Visual Computer*. DOI : 10.1007/s00371-022-02696-8.

YIN, Z. et J. SHI (2018). « GeoNet : Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose ». eng. In : *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. ISSN : 2575-7075. IEEE, p. 1983-1992.

YOSHIDA, E., T. ARAI, J. OTA et T. MIKI (sept. 1994). « Effect of grouping in local communication system of multiple mobile robots ». In : *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*. T. 2, 808-815 vol.2. DOI : 10.1109/IROS.1994.407546.

YUTA, S. et S. PREMVUTI (juin 1991). « Consideration on cooperation of multiple autonomous mobile robots-introduction to modest cooperation ». In : *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*, 545-550 vol.1. DOI : 10.1109/ICAR.1991.240596.

Annexes

Annexe A

Régression Logistique

(*La définition de cet outil est présentée dans la section 2.4.4.1.*)

Transformation des Échantillons vers l'Espace Logarithmique (Logits)

Lorsque nous effectuons une Régression Logistique, nous souhaitons modéliser la probabilité qu'une observation appartienne à une classe particulière. Pour ce faire, nous utilisons une fonction logistique, également appelée fonction sigmoïde. Cependant, avant d'appliquer la fonction sigmoïde, nous effectuons une transformation des échantillons vers l'espace logarithmique, appelé espace logits.

La transformation logarithmique des échantillons X vers l'espace logits est effectuée en utilisant la fonction logistique inverse, également connue sous le nom de fonction logit :

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) \tag{A.1}$$

où p représente la probabilité de l'échantillon d'appartenir à la classe positive. Cette transformation permet de modéliser la relation linéaire entre les caractéristiques et les probabilités de classe.

La conversion inverse des logits vers les probabilités de classe nous permet d'obtenir des valeurs entre 0 et 1, représentant la probabilité que chaque échantillon appartienne à la classe positive. Ces probabilités sont ensuite utilisées pour effectuer des prédictions et prendre des décisions de classification.

Cela transforme le problème en un modèle de régression linéaire de la forme :

$$z = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n \tag{A.2}$$

où x_1 à x_n représentent les valeurs des n attributs ou variables d'entrée et β_0 à β_n représentent les poids ou l'interception.

β affecte le logit (logarithme des chances). Pour les variables d'entrée continues, le logarithme du rapport de vraisemblance doit être optimisé pour avoir une valeur maximale afin d'obtenir la courbe la mieux adaptée à la classification.

Conversion Inverse vers les Probabilités de Classe

Après avoir effectué les calculs dans l'espace logits, il faut le rapporter à l'espace d'origine. Pour ce faire, le modèle de (A.2) est transformé à chaque échantillon logits z vers une représentation sur l'intervalle [0,1] à l'aide de la fonction sigmoid pour obtenir la probabilité p de la classe positive définie comme suit :

$$P(c_0|x_1, x_2, ..., x_n) = \frac{1}{1 + e^{-z}} \tag{A.3}$$

où e est la base du logarithme naturel.

Utilisation Pratique

La transformation logarithmique dans la Régression Logistique joue un rôle clé dans la modélisation des problèmes de classification. Elle permet de capturer la relation linéaire entre les caractéristiques d'entrée et les probabilités de classe, ce qui en fait un outil puissant dans l'analyse de données et l'apprentissage automatique.

Hyper-paramètres d'optimisation

Pour ajuster au mieux un modèle de régression logistique, les paramètres d'ajustement sont les suivants :

— Pénalité : détermine le type de régularisation appliqué au modèle de régression logistique. La régularisation permet d'éviter l'ajustement excessif en ajoutant un terme de pénalité à la fonction de perte. Il existe deux méthodes : L1 et L2. Géron, 2019 les présente comme suit :

1. Régularisation L1 (Régression par moindre rétrécissement absolu et opérateur de sélection - Régularisation Lasso) : elle ajoute un terme de régularisation à la fonction de coût, mais elle utilise la norme l_1 du vecteur de poids.
2. Régularisation L2 (Régularisation Ridge) : un terme de régularisation égal à $\alpha \sum_{i=1}^{n} \theta_i^2$ est ajouté à la fonction de coût. Cela oblige l'algorithme d'apprentissage non seulement à s'adapter aux données, mais aussi à maintenir les poids du modèle aussi faibles que possible.

La principale différence intuitive entre les régularisations L1 et L2 est que la régularisation L1 tente d'estimer la médiane des données tandis que la régularisation L2 tente d'estimer la moyenne des données afin d'éviter l'ajustement excessif.

— C : est l'inverse de la force de régularisation. Il contrôle l'étendue de la régularisation appliquée au modèle de régression logistique. Une valeur plus petite de C augmente la force de régularisation, ce qui se traduit par un modèle plus contraint avec des coefficients plus petits.

— Solveur : spécifie l'algorithme utilisé pour l'optimisation pendant l'apprentissage du modèle. Différents facteurs sont pris en compte par les différents solveurs, tels que la taille de l'ensemble de données, la vitesse de convergence, la prise en charge de la régularisation, la linéarité des données, l'efficacité de la mémoire, les contraintes, la capacité de parallélisation, l'approximation du Hessien et l'utilisation de mises à jour stochastiques ou par lots.
— Weight class : Les données sont déséquilibrées et si le modèle est formé en utilisant les données telles quelles, il sera biaisé en faveur de la classe abondante. Pour éviter cela, l'apprentissage sensible aux coûts est déployé. Il prend en compte le coût d'une mauvaise classification des différentes classes dans un problème de classification multi-classes. Dans ce cas, un poids doit être accordé à chacune des étiquettes utilisées.

Annexe B

Arbres de décision

(*La définition de cet outil est présentée dans la section 2.4.4.1.*)

Structure de l'Arbre de Décision

Un arbre de décision est une structure arborescente composée de nœuds. Il est utilisé pour prendre des décisions en suivant un chemin depuis le nœud racine jusqu'à une feuille (nœud terminal). Chaque nœud interne représente une décision basée sur une caractéristique (ou attribut) particulière de l'ensemble de données, tandis que les feuilles représentent les étiquettes de classe ou les valeurs de régression.

Apprentissage de l'Arbre de Décision

L'apprentissage d'un arbre de décision consiste à sélectionner de manière itérative les caractéristiques qui partitionnent le mieux l'ensemble de données. L'objectif est de maximiser la pureté des nœuds fils par rapport au nœud parent. La mesure de pureté la plus couramment utilisée est l'entropie ou le critère de Gini.

Gain d'information

Bien qu'il existe de multiples façons de sélectionner le meilleur attribut à chaque nœud, deux méthodes, le *gain d'information* et l'*impureté de Gini*, servent de critères de division populaires pour les modèles d'arbres de décision. Ils permettent d'évaluer la qualité de chaque condition de test et la capacité à classer les échantillons dans une classe. D'une part, le gain d'information représente la différence d'entropie avant et après une partition sur un attribut donné. L'attribut présentant le gain d'information le plus élevé produira la meilleure répartition, vu qu'il aura la plus petite quantité d'entropie, et donc le moins d'impureté. Ayant l'ensemble de données S de c classes et d'entropie $H(S)$, l'attribue A peut être déterminé par le gain d'information calculé comme

suit :

$$Gain(S, A) = H(S) - \sum_{v \in valeurs(A)} \frac{|S_v|}{|S|} \times H(S_v) \tag{B.1}$$

Impureté de Gini

L'impureté de Gini est la probabilité de classer incorrectement un point de données aléatoire dans l'ensemble de données s'il était étiqueté sur la base de la distribution des classes de l'ensemble de données. Ayant c classes totales, avec une probabilité p_i de choisir un point de données de la classe i, l'impureté de Gini se calcule comme suit :

$$Impurete = 1 - \sum_{i}^{c} p_i^2 \tag{B.2}$$

Où $\sum_{i}^{c} p_i^2$ est Gini, et son impureté est $1 - Gini$.

Choix de la Caractéristique

Une fois que l'impureté d'un ensemble de données est mesurée à l'aide de l'entropie ou du critère de Gini, l'algorithme d'apprentissage des arbres de décision choisit la caractéristique qui maximise la réduction de l'impureté (ou maximise le gain d'information). Cela se fait en évaluant chaque caractéristique possible et en calculant la métrique d'impureté pour chaque partition résultante.

Construction de l'Arbre

L'arbre de décision est construit de manière récursive en choisissant la caractéristique optimale à chaque nœud, en partitionnant l'ensemble de données en sous-ensembles en fonction de cette caractéristique, puis en répétant le processus pour chaque sous-ensemble jusqu'à ce qu'un critère d'arrêt soit atteint, tel que la profondeur maximale de l'arbre ou le nombre minimum d'exemples par feuille.

Prédiction avec l'Arbre de Décision

Une fois l'arbre de décision construit, il peut être utilisé pour effectuer des prédictions en parcourant l'arbre en fonction des caractéristiques de l'exemple de test. La décision finale est prise lorsque l'arbre atteint une feuille, et la classe ou la valeur de régression associée à cette feuille est la prédiction.

Annexe C

Réseaux de Neurones

(*La définition de cet outil est présentée dans la section 2.4.4.1*)

Types de paramètres

Dans la plupart des cas, un réseau de neurone est un système adaptatif qui modifie sa structure en fonction des informations externes ou internes qui circulent dans le réseau pendant la phase d'apprentissage. Les réseaux de neurone modernes sont des outils de modélisation de données statistiques non linéaires. Ils sont généralement utilisés pour modéliser des relations complexes entre les entrées et les sorties ou pour trouver des modèles dans les données. En se basant sur D'Addona, 2014, ils sont généralement définis par trois types de paramètres :

1. Le modèle d'interconnexion entre les différentes couches de neurones.
2. Le processus d'apprentissage pour la mise à jour des poids des interconnexions.
3. La fonction d'activation qui convertit l'entrée pondérée d'un neurone en activation de sortie.

Parties du Modèle

1. **Neurones (ou unités) :** Les unités de base du réseau de neurones. Ils reçoivent des entrées, effectuent des calculs et produisent des sorties.
2. **Couches (Layers) :** Les neurones sont organisés en couches. On distingue généralement trois types de couches dans un réseau de neurones :
   — *Couche d'entrée (Input Layer) :* Reçoit les données d'entrée et transmet les informations aux couches suivantes.
   — *Couches cachées (Hidden Layers) :* Les couches intermédiaires entre la couche d'entrée et la couche de sortie. Elles effectuent des transformations non linéaires sur les données.
   — *Couche de sortie (Output Layer) :* Produit les résultats finaux du réseau de neurones en fonction des données d'entrée et des transformations effectuées par les couches cachées.

3. **Poids (Weights) :** Chaque connexion entre deux neurones est associée à un poids. Les poids déterminent l'importance de l'entrée dans le calcul de la sortie d'un neurone. L'équation suivante montre comment le poids w est utilisé pour calculer la sortie y d'un neurone :

$$y = f\left(\sum_{i=1}^{n}(x_i \cdot w_i) + b\right) \tag{C.1}$$

Où :
— y est la sortie du neurone,
— f est la fonction d'activation du neurone,
— x_i sont les entrées,
— w_i sont les poids associés aux entrées,
— b est le biais (bias).

Méthodes d'Apprentissage des Poids

— **Propagation avant (Feedforward) :** Le processus d'inférence où les données d'entrée sont transmises à travers le réseau de neurones pour obtenir une prédiction en utilisant les poids actuels.
— **Rétropropagation de l'erreur (Backpropagation) :** L'algorithme d'apprentissage qui calcule les gradients de la fonction de perte par rapport aux poids du réseau, puis ajuste les poids pour minimiser la perte.
— **Descente de gradient (Gradient Descent) :** La méthode d'optimisation qui utilise les gradients calculés par la rétropropagation pour mettre à jour les poids du réseau. Il existe plusieurs variantes de la descente de gradient, telles que la descente de gradient stochastique (SGD) et l'Adam.

Types de Fonctions d'Activation

— **Fonction Sigmoïde :** Une fonction en forme de "S" qui transforme les valeurs en un intervalle entre 0 et 1. Elle était couramment utilisée dans les réseaux de neurones plus anciens.

$$\sigma(x) = \quad \frac{1}{1 + e^{-x}} \tag{C.2}$$

— **Fonction ReLU (Rectified Linear Unit) :** Une fonction qui remplace les valeurs négatives par zéro et laisse les valeurs positives inchangées. Elle est largement utilisée dans les réseaux de neurones profonds en raison de sa simplicité et de sa capacité à atténuer le problème du "vanishing gradient".

$$f(x) = \begin{cases} x, & \text{si } x > 0, \\ 0, & \text{autrement.} \end{cases} \tag{C.3}$$

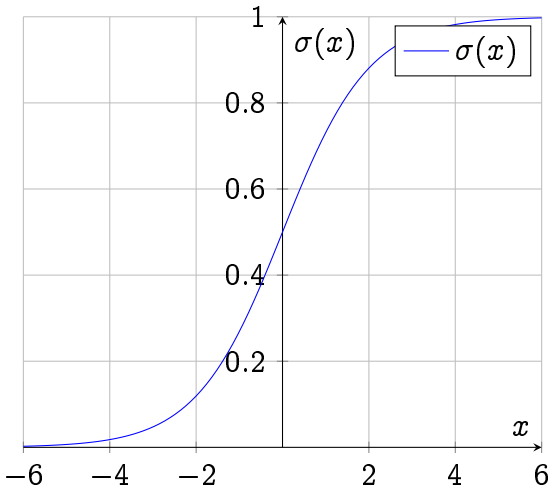


Figure C.1 – Tracé de la fonction sigmoide.

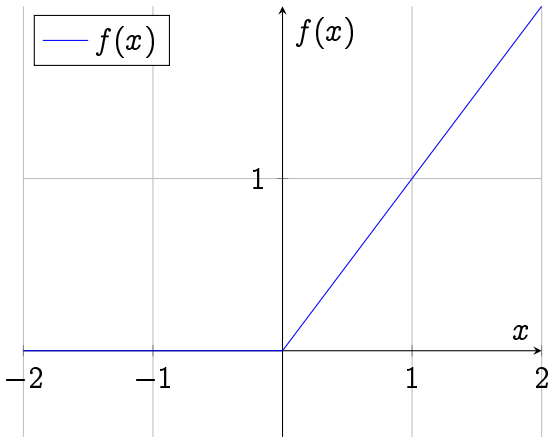


Figure C.2 – Tracé de la fonction ReLU.

— **Fonction Tangente Hyperbolique (tanh) :** Une fonction similaire à la fonction sigmoïde, mais elle transforme les valeurs en un intervalle entre -1 et 1. Elle est parfois utilisée dans les couches cachées des réseaux de neurones.

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{C.4}$$

— **Fonction Linéaire :** Une fonction qui maintient les valeurs inchangées. Elle est généralement utilisée dans les couches de sortie pour les régressions linéaires.

Apprentissage en Rétropropagation

L'apprentissage en rétropropagation (ou backpropagation) est l'algorithme d'apprentissage principal utilisé pour entraîner des réseaux de neurones. Il s'agit d'une méthode d'optimisation qui ajuste les poids du réseau afin de minimiser une fonction de perte décrivant l'écart entre les prédictions du réseau

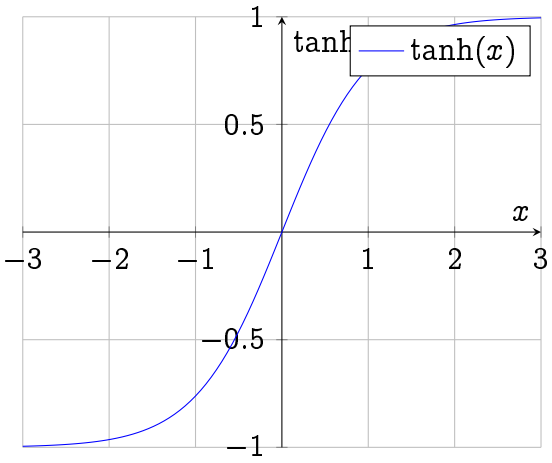


Figure C.3 – Tracé de la fonction tangente hyperbolique.

et les valeurs attendues pour un ensemble de données d'entraînement.

Calcul des Gradients

Le processus d'apprentissage en rétropropagation commence par le calcul des gradients de la fonction de perte par rapport aux poids du réseau. Ces gradients indiquent la direction dans laquelle les poids doivent être ajustés pour minimiser la perte. Les gradients sont calculés à l'aide de la règle de la chaîne (rule of chain) de calcul différentiel.

Soit L la fonction de perte du réseau et w un poids quelconque du réseau. Le gradient $\frac{\partial L}{\partial w}$ par rapport à ce poids est calculé comme suit :

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \text{sortie}} \cdot \frac{\partial \text{sortie}}{\partial \text{entrée}} \cdot \frac{\partial \text{entrée}}{\partial w}$$

où :

— $\frac{\partial L}{\partial \text{sortie}}$ est la dérivée de la fonction de perte par rapport à la sortie du réseau.

— $\frac{\partial \text{sortie}}{\partial \text{entrée}}$ est la dérivée de la fonction d'activation de la couche par rapport à son entrée.

— $\frac{\partial \text{entrée}}{\partial w}$ est la dérivée de l'entrée de la couche par rapport au poids w.

Mise à Jour des Poids

Une fois les gradients calculés pour tous les poids du réseau, la mise à jour des poids est effectuée pour minimiser la fonction de perte. Cela se fait généralement à l'aide de l'algorithme de descente de gradient. La formule générale de la mise à jour des poids est la suivante :

$$w_{\text{nouveau}} = w_{\text{ancien}} - \alpha \cdot \frac{\partial L}{\partial w}$$

où w_{nouveau} est le poids mis à jour, w_{ancien} est le poids précédent, α est le taux d'apprentissage (learning rate), et $\frac{\partial L}{\partial w}$ est le gradient par rapport à ce poids.

Opérations dans les réseaux de neurones

Normalisation

La normalisation, telle que la normalisation par lots (Batch Normalization), est utilisée pour stabiliser l'apprentissage en normalisant les activations intermédiaires dans le réseau. Elle permet d'accélérer la convergence de l'apprentissage en réduisant le déséquilibre des valeurs.

Dropout

Le dropout est une technique de régularisation qui consiste à désactiver aléatoirement un pourcentage de neurones lors de l'entraînement. Cela permet d'éviter le surapprentissage en introduisant du bruit dans le réseau.

Réseaux de Neurones Convolutionnels (CNN)

Les CNN ont révolutionné le domaine de la vision par ordinateur et sont couramment utilisés dans des applications de traitement d'images.

Les réseaux de neurones convolutionnels sont notamment accordés avec l'opération de convolution. Dans la suite, nous présentons les opérations que les couches des CNN peuvent contenir.

Opérations dans les réseaux convolutionnels

Les CNN sont composés de plusieurs opérations qui permettent d'extraire des caractéristiques pertinentes à partir des données en entrée.

Convolution

La convolution est l'opération fondamentale dans un CNN. Elle consiste en la superposition d'un noyau de convolution (ou filtre) sur une région de l'entrée, suivie de la multiplication élément par élément entre les valeurs du noyau et de la région. La somme des produits est ensuite calculée pour produire une seule valeur dans la carte de caractéristiques (feature map) de sortie.

L'opération de convolution pour une entrée X avec un noyau K est définie comme suit :

$$Y(i,j) = (X * K)(i,j) = \sum_m \sum_n X(i+m, j+n) K(m,n) \tag{C.5}$$

où :

— Y est la carte de caractéristiques de sortie.

— K est le noyau de convolution.
— i et j sont les coordonnées spatiales dans la carte de caractéristiques de sortie.
— m et n sont les coordonnées spatiales dans le noyau de convolution.

Pooling (Sous-échantillonnage)

Le pooling est une opération couramment utilisée dans les CNN pour réduire la dimension spatiale des cartes de caractéristiques. Il permet de réduire la complexité du modèle et d'obtenir une certaine invariance aux translations. Une opération de pooling typique est le pooling max, qui consiste à prendre la valeur maximale dans une région donnée de la carte de caractéristiques.

L'opération de pooling max pour une entrée X avec une fenêtre de pooling de taille 2×2 est définie comme suit :

$$Y(i,j) = \max\left(X(2i,2j), X(2i,2j+1), X(2i+1,2j), X(2i+1,2j+1)\right) \tag{C.6}$$

où :
— Y est la carte de caractéristiques de sortie après pooling.
— X est la carte de caractéristiques d'entrée.
— i et j sont les coordonnées spatiales dans la carte de caractéristiques de sortie.

Aplanissement (Flattening) des Données

L'opération d'aplanissement est couramment utilisée pour transformer des données structurées de matrice en un format vectoriel. Elle permet de convertir des données multidimensionnelles, telles que des images ou des tableaux, en un vecteur unidimensionnel.

L'aplanissement consiste à prendre chaque élément des données structurées et à les disposer dans un ordre séquentiel pour former un vecteur. Par exemple, pour une image représentée sous forme de matrice, chaque pixel est extrait et placé dans le vecteur dans un ordre spécifique.

Réseaux de Neurones Récurrents (RNN)

Cellule RNN

La cellule RNN est l'unité de base d'un réseau de neurones récurrent. Elle prend une entrée à un certain moment t et produit une sortie à ce même moment t, ainsi qu'une sortie qui est transmise à la cellule pour le pas de temps suivant $t+1$. La sortie de la cellule à l'instant t dépend donc de l'entrée à cet instant x_t et de la sortie précédente h_{t-1}.

La formule générale pour la sortie d'une cellule RNN est définie comme suit :

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{C.7}$$

où :
— h_t est la sortie de la cellule à l'instant t.
— f est une fonction d'activation, généralement la fonction tangente hyperbolique (tanh).
— W_{hx} est la matrice de poids pour l'entrée.
— W_{hh} est la matrice de poids pour la sortie précédente.
— b_h est le biais de la cellule.

Propagation dans le Temps

La propagation dans le temps est le processus par lequel un réseau de neurones récurrent est utilisé pour traiter une séquence. L'entrée est alimentée dans le réseau pas à pas, et la sortie dépend de l'histoire des entrées précédentes. Le processus de propagation dans le temps est généralement représenté par la formule suivante :

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{C.8}$$

où h_t est la sortie à l'instant t, x_t est l'entrée à l'instant t, h_{t-1} est la sortie précédente, et les autres termes sont les poids et le biais de la cellule.