Système embarqué de fusion multi-capteurs pour la détection et le suivi d'obstacles statiques et dynamiques

Mokhtar Bouain

Thèse de doctorat

Pour obtenir le grade de Docteur de

l'UNIVERSITÉ POLYTECHNIQUE HAUTS-DE-FRANCE

Doctorat en informatique

Présentée et soutenue par:

Mokhtar BOUAIN

Le 27/05/2019, à Valenciennes

Ecole doctorale :
Sciences Pour l'Ingénieur (ED SPI 072)

Equipe de recherche, Laboratoire :
Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines (LAMIH – UMR 8201)

Système embarqué de fusion multi-capteurs pour la détection et le suivi d'obstacles statiques et dynamiques

Présidente du jury:

- Pascale LE GALL, Professeur de CentraleSupélec - Laboratoire MICS

Rapporteurs:

- Francis ROUSSEAUX, Professeur de l'Université de Reims - Laboratoire CReSTIC
- Jean-Philippe DIGUET, Directeur de Recherche CNRS au Laboratoire LAB-STICC

Directeur de thèse:

- Rabie BEN ATITALLAH, MdC, HDR de l'Université de Galatasaray, Turquie

Co-encadrant:

- Denis BERDJAG, MdC de l'Université Polytechnique Hauts-de-France, LAMIH

Responsable industriel:

- Nizar FAKHFAKH, Docteur et responsable industriel

Résumé

Ce mémoire s'inscrit dans le cadre d'une Convention industrielle de formation par la recherche (*Cifre*) en partenariat avec l'entreprise NAVYA Technology, spécialisée dans la fabrication des véhicules électriques autonomes. La technologie de la navette *ARMA* conçue par *Navya*, utilise des capteurs proprioceptifs et extéroceptifs pour la localisation (*SLAM* : Simultaneous Localization and Mapping) ainsi que pour la détection et le suivi d'obstacles statiques et dynamiques (*DTMO* : Detection and Tracking of Moving Objects). Dans cette thèse, nous nous intéressons à la fonctionnalité *DTMO* qui a pour objectif de détecter les objets autour du véhicule afin d'éviter les collisions et assurer une conduite autonome, sécurisée et fiable.

Nos contributions à travers cette thèse sont les suivantes : **(1)** Nous avons proposé une méthodologie de calibration extrinsèque entre un système mono-vision et un capteur LIDAR 2D. Nous avons développé un environnement de calibration qui inclut l'extraction des caractéristiques de la mire de calibration par les deux capteurs. Nous avons validé cette contribution sur la navette autonome ARMA. **(2)** Étant donné qu'un véhicule autonome se situe dans un environnement incertain, nous avons choisi la théorie de croyance (ou Dempster Shafer) et ses outils pour modéliser les connaissances et les incertitudes. Nous avons mis à contribution une approche de fusion entre une caméra et un radar dans le but d'améliorer la fiabilité de détection et de classification des piétons et des véhicules. Pour classifier les obstacles, les connaissances sont déduites à partir du paramètre Radar Cross Section (RCS) fourni par le radar, qui reflète la nature de l'obstacle détecté, et la classification effectuée par la caméra. **(3)** Dans une troisième contribution, nous avons exploré l'espace de conception d'une plateforme embarquée dédiée à la détection d'obstacles. Parmi les technologies d'accélération du traitement, nous avons choisi la solution FPGA permettant d'offrir une puissance de calcul performante, par le moyen de ses ressources reconfigurables, avec une faible consommation de puissance tout en garantissant un coût maitrisé. Nous avons présenté une conception d'une plateforme embarquée à base d'un système sur puce hétérogène ($Zynq-7000$) pour fusionner les postions d'obstacles détectés par une caméra stéréoscopique et un LIDAR en utilisant l'approche bayésienne.

Keywords : Véhicule autonome, calibration des capteurs, fusion multi-capteurs, exploration de l'espace de conception, FPGA

Abstract

This thesis is a part of an Industrial Agreements for Training through Research (*Cifre*) in partnership with NAVYA Technology, for autonomous vehicles. *Navya* designed the shuttle *ARMA* which used proprioceptive and exteroceptive sensors for localization (*SLAM* : Simultaneous Localization and Mapping) as well as the detection and tracking of obstacles (*DTMO* : Detection and Tracking of Moving Objects).

In this thesis, we were interested in the *DTMO* task which aimed to detect objects around the vehicle to avoid collisions and to ensure secure and reliable autonomous driving. Our contributions through this thesis were as follows : **(1)** We proposed an extrinsic calibration methodology between a single-vision system and a 2D LIDAR sensor. We presented a complete implementable toolchain to extract the co-features for both types of sensors. We validated this contribution on the autonomous shuttle *ARMA*. **(2)** Since an autonomous vehicle is located in an uncertain environment, we chose the belief theory (or Dempster-Shafer) to model and manage knowledge and uncertainties. We proposed a camera-radar fusion methodology to improve the reliability of detection and classification for both pedestrians and vehicles. **(3)** Our implementation target is a 100% electric autonomous shuttle. So, it was necessary to use an embedded platform to respect constraints such as power consumption. In this contribution, FPGA was our solution to host the intensive processing of autonomous vehicle perception algorithms. We presented an embedded solution based on heterogeneous System On Chip (Zynq-7000) to fuse the positions of obstacles detected by a stereoscopic camera and a LIDAR using the Bayesian approach.

Keywords : Autonomous vehicle, sensor calibration, sensor data fusion, design space exploration, FPGA

Remerciements

Tout d'abord, je souhaite remercier mes parents, ma mère Zohra et mon père Ibrahim, pour m'avoir toujours soutenu, aussi mes sœurs Mariam, Asma et Chiraz et ma fiancée Mouna. J'ai toujours pu compter sur eux et je sais que je leur dois beaucoup.

J'exprime toute ma gratitude envers ceux qui ont accepté de valider ce travail de thèse. J'adresse mes vifs remerciements à M. Jean-Philippe DIGUET, directeur de recherche CNRS au LAB-STICC et à M. Francis ROUSSEAUX, professeur de l'Université de Reims, d'avoir accepté de juger mon travail et m'avoir faite l'insigne honneur d'être rapporteurs de mon mémoire de thèse. Mes remerciements vont également à Mme Pascale LE GALL, professeur de CentraleSupélec, d'avoir bien voulu examiner ce travail de thèse.

Je tiens à remercier, mon directeur de thèse, Rabie Ben Atitallah, qui m'a encadré et dirigé dans mes recherches tout au long de ces années, je lui remercier pour ses conseils et sa confiance. Je remercie et j'exprime mes vives reconnaissances à Denis Berdjag d'avoir passé un temps précieux à travailler avec moi. Il a toujours été disponible, et s'est toujours intéressé à l'avancée de mes travaux. Je remercie mon responsable industriel, Nizar fakfak qui m'a accompagné durant ma présence à Navya.

Enfin, je tiens à exprimer ma gratitude à tous les membres du laboratoire LAMIH, en particulier Mr Thierry Marie Guerra et Mr Said Hanafi, et à toute l'équipe de Navya. Un grand merci à mon collègue Karim qui m'a aidé pour préparer ma soutenance.

MERCI à tous et à bientôt :)

Mokhtar

Publications de l'auteur

1. **Mokhtar Bouain**, Denis Berdjag, Nizar Fakhfakh, Rabie Ben Atitallah, An Extrinsic Sensor Calibration Framework for Sensor-fusion based Autonomous Vehicle Perception, *In the 14th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Madrid, Spain, July 2017.
2. **Mokhtar Bouain**, Karim MA Ali, Denis Berdjag, Nizar Fakhfakh, Rabie Ben Atitallah, An embedded multi-sensor data fusion design for vehicle perception tasks, In *the 10th International Conference on Computer Science and Information Technology (ICCSIT)*, Florence, Italy, October, 2017. **Accepted and published in :** *Journal of Communications (JOCM)*, vol. 13, no. 1, pp. 8-14, 2018. Doi : 10.12720/jcm.13.1.8-14.
3. **Mokhtar Bouain**, Denis Berdjag, Nizar Fakhfakh, Rabie Ben Atitallah, Exploring High-Level Synthesis Tools for Vehicle Perception Tasks, *In the 9tH European Congress on Embedded Real Time Software and Systems (ERTS)*, Toulouse, France, January February, 2018.
4. **Mokhtar Bouain**, Denis Berdjag, Nizar Fakhfakh, Rabie Ben Atitallah, Multi-Sensor Fusion for Obstacle Detection and Recognition : A Belief-based Approach, *In the 21st International Conference on Information Fusion (FUSION 2018)*, University of Cambridge, United Kingdom, July, 2018.
5. **Mokhtar Bouain**, Denis Berdjag, Nizar Fakhfakh, Rabie Ben Atitallah, A comprehensive approach for camera/LIDAR frame alignment, *Book Chapter in Lecture Notes in Electrical Engineering (LNEE),* 2018.

Abréviations et Notations

Abréviations :

LIDAR	:	Light Detection and Rangin
CAN	:	Controller Area Network
SOC	:	System On Chip
HW	:	Hardware
SW	:	Software
IP	:	Intellectual Property
FPGA	:	Field Programmable Gate Arrays
VHDL	:	VHSIC Hardware Description Language
RCS	:	Radar Cross Section
GPS	:	Global Positioning System
HLS	:	High-Level Synthesis
SLAM	:	Simultaneous Localization And Mapping
DTMO	:	Detection and Tracking of Moving Objects
TBM	:	Transferable Belief Model
BBA	:	Basic Belief Assignment
ACF	:	Aggregate Channel Features

Notations :

$\overrightarrow{()}$	:	Vecteur
$\overrightarrow{t}$	:	Vecteur de translation
R	:	Matrice de rotation
Ω	:	Cadre de discernement
m^{Ω}	:	Fonction de masse de croyance du cadre de discernement Ω
$BetP^{\Omega}$	:	Probabilité pignistique
P	:	Fonction de probabilité
k	:	Temps discret
q	:	Quaternion

Table des matières

Table des figures

Liste des tableaux

Chapitre I

Introduction

Sommaire

I.1 Contexte

La mobilité durable, fluidifiée et sécurisée des personnes et des biens est au cœur des politiques de développement de transports urbains. C'est un des enjeux majeurs auxquelles sont confrontées d'une part, les villes dans leur développement face aux problématiques de pollution et de congestion de trafic, d'autre part, les Autorités Organisatrice de Transport (AOT) et les grands opérateurs tels que la $SNCF$, ayant à satisfaire une demande de transport de plus en plus personnalisée. Dans ce contexte, la question de la mobilité du « premier ou dernier kilomètre » est un des défis à relever. Elle concerne à la fois le trajet qui sépare le domicile de l'usager à un point de connexion avec un réseau de Transports en Commun en Site Propre ($TCSP$) ou non, et celui qui sépare un point de connexion d'un $TCSP$ à un lieu de destination spécifique tels que les campus universitaires, les centres villes piétonniers, les grands sites industriels privés, les aéroports, les parcs à thème, ou bien encore les centres hospitaliers.

Une solution émergente consiste à proposer des véhicules autonomes de petit transport collectif susceptibles de pouvoir se déplacer à la demande dans des environnements diverses et variés. Seules quelques métropoles au monde se sont emparées des potentialités apportées par ces engins légers et robotisés qui préfigurent sans doute les transports urbains du futur. À l'échelle internationale et en février 2018, l'Autorité des Routes et des Transports de Dubaï a déclaré avoir mis en service deux petites navettes autonomes et électriques qui peuvent chacun transporter six passagers. Ces nouveaux minibus ont été testés dans le cadre du programme Dubai Future Accelerators et à terme, se multiplier pour former une flotte complète et permettre une autonomie d'un quart des transports collectifs de Dubaï d'ici à 2030. De son côté, la compagnie de chemins de fer allemande Deutsche Bahn a présenté en octobre 2017 son premier minibus sans chauffeur pouvant transporter 12 passagers. Il s'agit d'une navette urbaine intelligente qui roule dans les ruelles d'une station thermale et achemine les passagers à la gare. Au plan national et en 2016, le producteur et fournisseur d'électricité en France et en Europe (EDF) a mis en service

six navettes autonomes de Navya [1] qui sont réparties sur les 220 hectares du site nucléaire de Civaux pour la mobilité du personnel. Comparées à un bus à moteur thermique, ces navettes électriques autonomes permettent de réduire les émissions de CO_2 de 44 tonnes par an, tout en offrant un meilleur service : le temps d'attente des passagers a été divisée par trois [11, 12].

La conception des véhicules autonomes à la fois sûrs, durables et efficients en termes de coût et d'autonomie constitue un défi d'ordre technologique et scientifique. En particulier, ce type de véhicule autonome requiert d'une part des équipements de localisation et de navigation dédiés à la gestion des obstacles statiques et dynamiques afin d'assurer le guidage temps réel, et d'autre part des moyens d'interfaçage plastique, de communication et de collecte d'informations pour garantir la sûreté de fonctionnement du système et la sécurité des passagers.

Ce mémoire s'inscrit dans le cadre d'une Convention industrielle de formation par la recherche (Cifre) en partenariat avec NAVYA Technology [12]. NAVYA est une entreprise française spécialisée dans la fabrication des véhicules électriques autonomes. La navette électrique $ARMA$ de la société NAVYA (Fig. I.1) a été conçue comme complément aux transports traditionnels sur le « premier ou dernier kilomètre » permettant de transporter jusqu'à 15 personnes. Cette navette sans conducteur s'intègre parfaitement dans les différents environnements nécessitant une mobilité simple, sécurisée et respectueuse de l'environnement.

FIGURE I.1: La navette autonome ARMA de la société Navya

1. Société française spécialisée dans la fabrication des véhicules électriques autonomes

Les fonctionnalités de base d'un système informatique du véhicule autonome peuvent être classées en trois grandes tâches, à savoir la perception, la planification et le contrôle. Afin d'assurer une navigation autonome, ces fonctionnalités illustrées par la fig. I.2, doivent interagir entre elles et avec l'environnement. Elles sont définies comme suit [13] :

- **Perception** [13] : désigne la capacité d'un système donné à collecter des informations en utilisant un ensemble de capteurs, et par la suite extraire des connaissances pertinentes de l'environnement. Pour un véhicule autonome, la perception de l'environnement est la compréhension contextuelle de l'environnement en se basant sur deux fonctionnalités principales qui sont : **(1)** Simultaneous Localization and Mapping ($SLAM$) [14] : c'est la tâche de localisation et cartographie simultanées. En fait, à partir des données mesurées, le véhicule se localise avec la fonctionnalité $SLAM$ qui permet de cartographier l'environnement. **(2)** Detection and Tracking of Moving Objects ($DTMO$) [15] : cette fonctionnalité permet de détecter et suivre les obstacles statiques et dynamiques autour du véhicule.

  La perception est une fonction fondamentale permet au véhicule autonome de recueillir des informations cruciales de l'environnement de conduite et de construire un modèle qui inclut, par exemple les positions des obstacles détectés, leurs vitesses, et même les prédictions de leurs états futurs. Pour percevoir cet environnement, un véhicule doit être équipé d'un ensemble de capteurs. Ces derniers peuvent être proprioceptifs (interne à l'instrument) [16] qui effectuent leurs mesures par rapport à ce qu'ils perçoivent localement du déplacement de véhicule tels qu'un odomètre et une centrale à inertie (IMU : Inertial Measurement Unit) ou extéroceptifs [16] qui se basent sur des mesures prises par rapport à leur environnement global tels qu'une caméra, un radar et un LIDAR...
- **Planification** [17, 18] : est le processus de prise de décision visant à déterminer une trajectoire, en général la meilleure, qui sera suivie par un véhicule autonome pour aller d'une localisation initiale à une autre finale dans un environnement défini. La trajectoire doit tenir compte des obstacles détectés tout

en optimisant les heuristiques conçues.

– **Contrôle** [19] : est la capacité du véhicule autonome à exécuter les actions pertinentes générées par le planificateur.

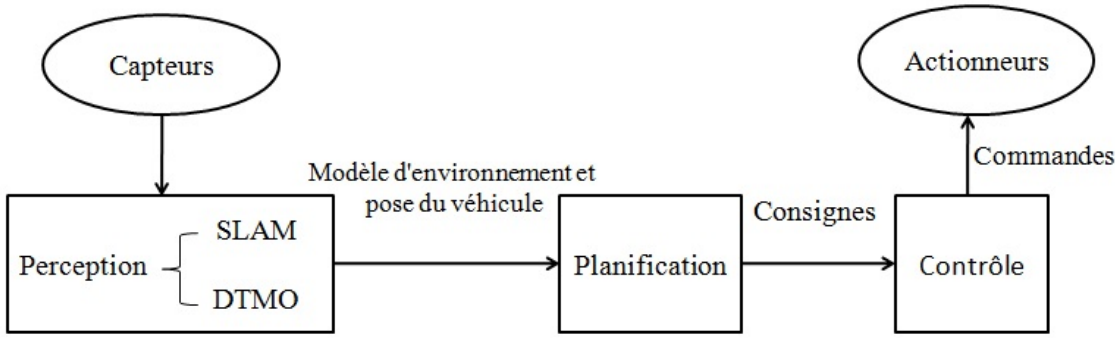

FIGURE I.2: Fonctionnalités principales de conduite autonome

Dans le contexte du véhicule autonome, notre sujet de thèse couvre principalement trois axes qui sont : l'alignement des capteurs, la fusion de données multi-capteurs pour la détection d'objets et le traitement intensif sur des calculateurs embarqués. La fig. I.3 présente le contexte de notre recherche.

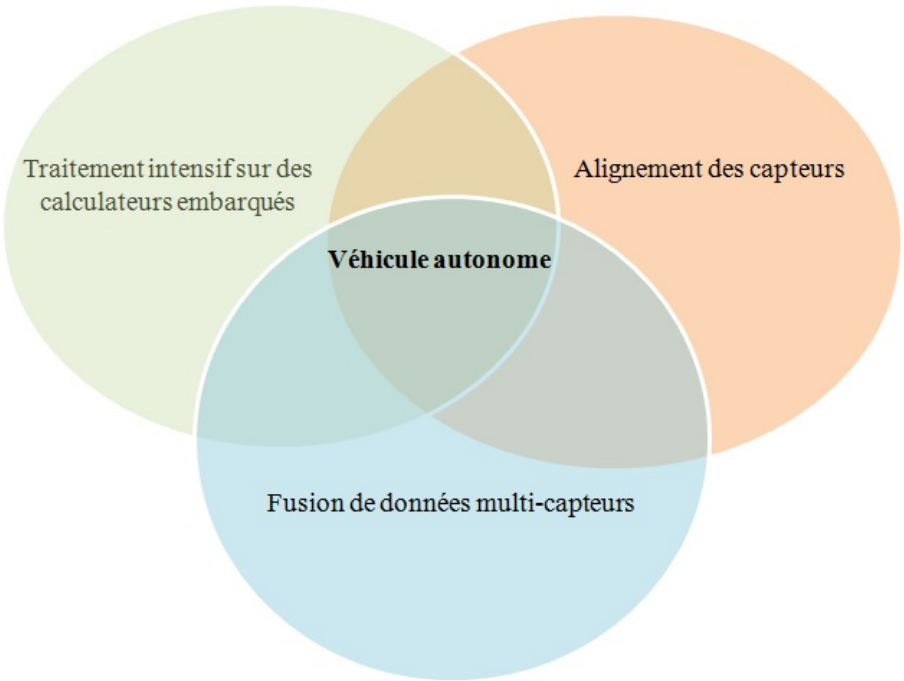

FIGURE I.3: Le contexte de notre recherche dans le domaine du véhicule autonome

I.2 Problématique

Avant d'aborder notre problématique, nous définissons dans ce mémoire les termes suivants :

- **Imprécision et incertitude** [20, 21] : Dans [20], une information X est modélisé par un couple (Valeur, Confiance). L'élément valeur est une valeur approchée de la valeur réelle de X et la confiance exprime un jugement sur la validité de l'information. L'imprécision concerne alors le contenu de l'information relatif à la composante *valeur* tandis que l'incertitude est relative à la composante *confiance*.
- **Imperfection des données** [22] : Les données fournies par les capteurs sont toujours affectées par un certain niveau d'imprécision et d'incertitude dans les mesures.
- **La théorie des fonctions de croyance** [23] : est un cadre formel pour le calcul et le raisonnement à partir d'informations partielles (incertaines, imprécises). Autres dénominations : théorie de $Dempster - Shafer$, théorie de l'évidence, modèle des croyances transférables.
- **Fausse alarme** (appelée aussi fausse alerte ou fausse détection ou faux positif) : si la perception est erronée concernant la présence de l'objet, c'est à dire le capteur détecte un écho ou une forme et la perception le considère comme un objet alors qu'il n'existe pas.
- **Détection manquée** (appelée aussi faux négatif) : si l'objet est présent mais le capteur ne le détecte pas.
- **Transformation rigide** [24] : est définie comme une transformation qui conserve la forme et la taille d'un objet et se décompose en deux opérations géométriques qui sont la rotation et la translation.

La technologie de la navette $ARMA$ utilise des capteurs proprioceptifs et extéroceptifs pour la localisation ($SLAM$) ainsi que la détection et le suivi d'obstacles statiques et dynamiques ($DTMO$). Dans cette thèse, nous nous intéressons à la fonctionnalité $DTMO$ qui a pour objectif de détecter les objets autour du véhicule afin d'éviter les collisions et par la suite assurer une conduite autonome, sécurisée et

fiable. Le déploiement des véhicules autonomes dans différents milieux tels qu'urbain implique que ces véhicules doivent être capables de naviguer d'une façon autonome dans un milieu complexe, dynamique et en présence des événements imprévus. À titre d'exemple, le véhicule doit cohabiter avec des carrefours, avec la coexistence de mobiles de plus en plus divers tels que des cyclistes et des piétons qui ne sont pas toujours disciplinés. Pour prendre en considération ces contraintes, le véhicule doit être doté de fonctionnalités avancées de perception qui sont basées sur la fusion d'informations diversifiées issues de plusieurs capteurs.

En fait, pour un véhicule autonome, un seul capteur n'est pas suffisant pour garantir une perception précise si le capteur a une grande incertitude, fournit de mesures insuffisantes et bruitées ou s'il a une faible résolution et un champ de vision limité. Plus précisément, la caméra a une bonne performance en termes de résolution et permet de réaliser des tâches importantes avec une bonne précision telle que la classification des obstacles (véhicule, piéton...) [25, 26]. Par contre, elle a une mauvaise performance pour fonctionner dans toutes les conditions métrologiques ainsi que pour l'estimation de la vitesse des obstacles. Pour un capteur de type radar, il a une bonne performance pour détecter les positions des obstacles ainsi que leurs vitesses relatives et il a un bon fonctionnement dans toutes les conditions métrologiques (plus fiable à la pluie, au brouillard, à la neige). Par contre, il a l'inconvénient de produire une quantité significative des fouillis (échos parasites). Pour le capteur LIDAR, il permet de détecter les positions latérales des obstacles avec une précision de résolution plus performante que celle du radar, mais il est sensible aux bruits et aux conditions métrologiques [27]. Par conséquent, chaque capteur a ses propres avantages et inconvénients. Compte tenu de la complémentarité des capteurs caméra, LIDAR et radar, cette combinaison est couramment utilisée dans la conception des véhicules autonomes. Il est donc important de développer des approches de fusion multi-capteurs pour tirer parti des avantages de chaque capteur. Dans ce contexte, l'objectif d'une architecture multi-capteurs est d'améliorer la précision de perception afin d'éviter les fausses alarmes et réduire les détections manquées dans un environnement incertain.

Plusieurs approches ont été proposées pour combiner les informations provenant de différentes observations des capteurs. Chacune a pour objectif de fournir un cadre scientifique pour modéliser et gérer les incertitudes. La théorie bayésienne [1] a été l'une des premières approches utilisées pour accomplir la fusion avec la prise de décision. Il s'agit d'une approche probabiliste qui repose sur un cadre mathématique rigoureux : la théorie des probabilités. Dans le cadre probabiliste, les imperfections de l'information et les informations elles même sont modélisées à partir des distributions des probabilités ou des mesures statistiques [28]. Cependant, cette approche a ses limites telles que sa complexité issue de la nécessité de spécifier un grand nombre de probabilités a priori pour pouvoir appliquer correctement les méthodes de raisonnement probabilistes [1]. Pour traiter les limitations perçues dans les méthodes probabilistes, certaines techniques alternatives de modélisation d'incertitude ont été proposées telles que :

- **La logique floue** [29] : qui a trouvé une grande popularité en tant qu'une méthode de représentation de l'incertitude ainsi que pour traiter les applications de fusion multi-capteurs. La logique floue permet de résoudre tous les problèmes dans lesquels on dispose de connaissances imprécises, soumises à des incertitudes de nature non-probabiliste [30].
- **La théorie de l'évidence** [31] : ou la théorie de $Dempster-Shafer$, c'est une théorie mathématique basée sur la notion des preuves utilisant les fonctions de croyance et le raisonnement plausible. Tandis que la théorie de probabilité fait intervenir la notion de probabilité a priori ce qui est gênant lorsque nous ne pouvons pas l'évaluer, la théorie de l'évidence permet d'éviter cette notion.

Une autre brique importante pour la perception, est la prise de décision en présence d'un ensemble de capteurs. En fonction du résultat de l'analyse des données, le véhicule doit prendre une décision de conduite. Par exemple, les caméras et les LIDARs détectent une forme. Les données recueillies sont analysées et fusionnées, ensuite la prise de décision aboutit à une action qui active ou désactive certaines commandes comme accélérer ou freiner. En Mars 2018, un véhicule autonome de la société $Uber$, équipé de différents capteurs tels que la caméra, le LIDAR et le radar, a eu un accident avec une piétonne traversant la route, causant sa mort. Les

capteurs du véhicule ont détecté la femme qui traversait la route avec son vélo, mais la perception a décidé qu'il s'agit d'un faux positif. Par conséquent, l'analyse, la fusion et la prise de décision sont des éléments complémentaires dans la perception et représentent les centres d'intérêt de la communauté scientifique de navigation autonome vu leur importance dans la sécurité routière.

De plus, certains capteurs ne sont pas conçus nativement pour fonctionner ensemble, ainsi, il est nécessaire de maîtriser la synchronisation et le calibrage géométrique des données réceptionnées. Le problème de synchronisation provient du fait que les fréquences de fonctionnement des capteurs, soient homogènes ou hétérogènes, peuvent être différentes ainsi que leurs données qui sont réceptionnées à différents instants [22]. Quant au calibrage géométrique, c'est le processus d'alignement des repères capteurs. Il s'agit de transformer chaque donnée à partir de son repère du capteur local dans un repère commun qui unifie tous les repères capteurs utilisés [22]. Nous pouvons distinguer une autre appellation de ce processus, celle de calibration extrinsèque. Le terme extrinsèque précise que les paramètres à estimer proviennent de "l'extérieur" des capteurs. Ces paramètres représentent une transformation rigide.

Par ailleurs, le véhicule autonome doit traiter simultanément des données diversifiées provenant de différents capteurs tels que la caméra radar et le LIDAR. De ce fait, le volume de données à traiter et à fusionner est important et nécessite le développement des algorithmes sophistiqués ainsi que des calculateurs demandant de plus en plus de ressources et consommant toujours plus d'énergie électrique. Pour l'instant, le traitement et l'analyse de données sont réalisés sur des PCs industriels. Cette solution, rapide à mettre en œuvre, n'est pas dimensionnée pour traiter un volume de données important surtout avec la demande croissante du calcul dû à l'ajout d'autres capteurs ou à l'implémentation des algorithmes intensifs de traitement du signal tels que les réseaux profonds (Deep Learning) et la segmentation (clustering). Une solution a été prise pour augmenter la puissance du calcul est l'ajout d'un PC à chaque besoin. Cependant, l'augmentation du nombre de PCs n'est pas une solution fiable en termes de consommation électrique (élevée), taille et communication entre eux. C'est pourquoi, la société NAVYA cherche à trouver une solution efficace répondant à un ensemble des exigences tels qu'une puissance de traitement élevée,

une faible consommation d'énergie électrique et un cycle de développement court.

Dans ce cadre, nous avons identifié trois problématiques à traiter dans notre thèse :

1. **Calibration des repères capteurs caméra/LIDAR** : Nous nous intéressons dans cette problématique à la calibration entre les capteurs caméra et LIDAR. Pour ce faire, nous avons besoin d'une méthodologie pour déterminer la position et l'orientation relatives des deux repères capteurs. Ce problème géométrique, qui établit une association entre les données des deux capteurs, doit être formulé, résolu et optimisé afin de déterminer les paramètres extrinsèques de calibration. Cette méthodologie confronte plusieurs défis tels l'établissement et la résolution de la fonction objectif qui définit le problème, l'extraction des caractéristiques des deux repères capteurs (exemple : extraction des formes de la mire de calibration) et le temps nécessaire pour la mise en œuvre de ce processus.

2. **Fusion multi-capteurs caméra/radar pour la détection des obstacles** : Nous nous intéressons dans cette problématique à la fusion de données entre les capteurs caméra et radar afin d'améliorer la précision de détection et de classification des obstacles. Puisque chaque capteur a ses propres avantages et inconvénients, l'enjeu scientifique consiste à développer une approche de fusion pour tenir compte de la complémentarité des capteurs. Cet enjeu couvre, la détermination du niveau de fusion (bas, intermédiaire ou haut), l'exploitation des observations fournies par ces capteurs et le cadre scientifique pour la représentation des connaissances ainsi que pour modéliser et gérer les incertitudes.

3. **Traitement intensif sur des calculateurs embarqués** : Le traitement et l'analyse des données issues des capteurs sont pour l'instant réalisés sur des PCs industriels. Cette solution, rapide à mettre en œuvre, n'est pas dimensionnée pour traiter un volume des données intensif afin de respecter la forte contrainte de traitement temps-réel. C'est dans ce contexte que Navya a besoin de trouver une solution performante embarquée pour ses calculateurs, qu'elle soit de bas coût et de basse consommation énergétique.

I.3 Contributions

Nos contributions à travers cette thèse sont les suivantes :

1. **Calibration des repères capteurs caméra/LIDAR** : Nous avons proposé une méthodologie de calibration extrinsèque entre un système mono-vision et un capteur LIDAR $2D$. Nous avons développé un environnement de calibration qui inclut l'extraction des caractéristiques des deux repères capteurs et la détermination des paramètres de calibration. Afin d'automatiser ce processus dans la mesure du possible, nous avons implémenté une chaîne d'outils complète pour extraire les caractéristiques : détection des lignes de la mire, en utilisant la transformée de $Hough$, assurée par la caméra et détection des points des bords de la mire en utilisant un processus de segmentation par le capteur LIDAR. Concernant le problème géométrique de calibration, nous avons reformulé et étendu une approche analytique afin de réduire le nombre de poses requises dans l'objectif de réduire le temps nécessaire de la mise en œuvre de ce processus. Pour évaluer notre contribution, nous avons comparé les résultats obtenus par rapport aux solutions existantes présentées dans notre étude de l'état de l'art. Les résultats obtenus ont validé avec succès l'implémentation de notre contribution sur la navette autonome ARMA.
2. **Fusion multi-capteurs caméra/radar pour la détection des obstacles** : Étant donné qu'un véhicule autonome se situe souvent dans un environnement incertain, nous avons choisi la théorie des fonctions de croyance (ou $Dempster-Shafer$) et ses outils pour modéliser et gérer les incertitudes. Cette théorie très expressive, permet de représenter les différents niveaux d'ignorance, ne nécessite pas de probabilités a priori, et gère les situations de conflit lorsque des preuves opposées apparaissent. Compte tenu de la complémentarité des capteurs caméra et radar, nous avons mis à contribution une approche de fusion entre une caméra et un radar dans le but d'améliorer la précision de détection et de classification des piétons et des véhicules. Les hypothèses pour classifier les obstacles sont déduites à partir des vitesses et des paramètres de Radar Cross Section (RCS) des obstacles fournis par le radar et la classifica-

tion assurée par la caméra. À noter que le paramètre RCS reflète la nature d'obstacle détecté et qui se diffère d'un obstacle à un autre (piéton, moto, véhicule...).

3. **Traitement intensif sur des calculateurs embarqués** : Étant donné que notre cible d'implémentation est une plateforme automobile autonome 100% électrique, nous avons visé à utiliser des calculateurs embarqués pour respecter les contraintes d'autonomie telles que la consommation d'énergie électrique. Par conséquent et dans cette contribution, le FPGA est notre candidat pour accueillir le traitement des algorithmes intensifs de perception du véhicule autonome. Nous avons présenté une conception d'une plateforme embarquée à base d'un système sur puce hétérogène le $Zynq-7000$ inclut un FPGA et un processeur embarqué pour fusionner les postions des obstacles détectés par une caméra stéréoscopique et un LIDAR en utilisant l'approche bayésienne. De plus, nous avons prouvé, à travers une implémentation matérielle d'un filtre de Kalman, que les logiciels de synthèse de haut-niveau représente une solution candidate à utiliser au développement de véhicule autonome pour accélérer l'implémentation matérielle des algorithmes intensifs.

I.4 Plan

En respectant l'ordre des contributions, le manuscrit est organisé selon le plan suivant :

Chapitre 2 État de l'art : Dans ce chapitre, nous commencerons par introduire les éléments et les modules impliqués autours de nos contributions à travers cette thèse. Puis, nous exposerons les travaux de recherche qui traitent le problème de calibration des capteurs caméra/LIDAR. Ensuite, nous présenterons le contexte de fusion multi-capteurs et la perception du véhicule autonome ainsi que ses fonctionnalités principales. Après, nous nous concentrerons sur la fonctionnalité $DTMO$ en montrant l'évolution de cette fonctionnalité de l'architecture mono-capteur vers celle de multi-capteurs et les niveaux de fusion possibles qui peuvent être appliqués. Les architectures ainsi que les outils utilisés pour la conception de système embarqué

tels que les outils de synthèse de haut niveau font également partie de notre analyse de l'existant. À partir de cette étude, nous positionnerons nos travaux et nous donnerons les grandes lignes de nos contributions.

Chapitre 3 Calibration extrinsèque caméra/LIDAR : Dans ce chapitre, nous présenterons notre méthodologie proposée de calibration extrinsèque entre un système mono-vision et un capteur LIDAR $2D$. Nous décrirons l'environnement développé pour réaliser ce processus. Au début, nous exposerons notre approche analytique puis nous présenterons la chaîne d'extraction des caractéristiques proposée : la détection des lignes de la mire, en utilisant la transformée de $Hough$, assurée par la caméra et la détection des points des bords de la mire en utilisant un processus de segmentation par le capteur LIDAR. Pour évaluer les résultats obtenus de notre contribution, nous les comparons par rapport aux solutions existantes présentées dans l'état de l'art.

Chapitre 4 Fusion multi-capteurs : caméra/radar : Dans ce chapitre, nous présenterons notre approche de détection et de classification des obstacles basée sur la théorie des fonctions de croyance. Nous commencerons par introduire la théorie des fonctions de croyance : son contexte, ses outils et ses opérateurs de combinaison. Puis nous détaillerons, les différentes étapes nécessaire de l'approche proposée de fusion entre les capteurs caméra et radar. Les résultats de simulation et les tests expérimentaux préliminaires seront présentés.

Chapitre 5 Exploration de l'espace de conception d'une plateforme embarquée de fusion multi-capteurs : Ce chapitre sera consacré à l'exploration de l'espace de conception de la plateforme embarqué dédiée au véhicule autonome.

Chapitre 6 Conclusion et perspectives : Nous conclurons cette thèse par le bilan des travaux effectués et nous détaillerons les contributions apportées avant d'aborder plusieurs perspectives à nos travaux.

Chapitre II

État de l'art

Sommaire

II.1 Introduction

L'objectif principal de cette thèse est de concevoir une plateforme embarquée de fusion multi-capteurs pour la détection des obstacles dans l'environnement d'une navette autonome. L'emploi d'une architecture multi-capteurs a pour but d'améliorer la précision de détection des obstacles. Cependant, lorsque le système est composé d'un ensemble de capteurs, chacun d'eux opère en général dans un repère relatif qui lui est propre, par conséquent, il est nécessaire de les représenter dans un référentiel commun. D'autre côté, l'industrialisation des véhicules autonomes exige d'avoir des éléments de calcul haute performance afin de permettre aux véhicules de prendre des décisions en temps réel dans des environnements complexes.

Cet ensemble d'exigences, nous amène à diviser ce chapitre comme suit : Dans la section II.2, nous commençons par introduire les éléments et les modules impliqués autours de nos contributions à travers cette thèse. Ensuite, nous présentons un tour d'horizon concernant la calibration entre les capteurs caméra et LIDAR dans la section II.3, la perception de véhicule autonome, en particulier la fonctionnalité *DTMO* dans la section II.4 et l'exploration des implémentations embarqués dédiées à la perception des véhicules autonomes dan la section II.5. Enfin nous positionnons nos travaux dans la section II.6.

II.2 Système embarqué de fusion multi-capteurs pour la fonctionnalité DTMO

Avant d'aborder l'état de l'art, il convient de présenter les éléments et les modules impliqués autours de nos contributions à travers cette thèse. La figure II.1 décrit la chaîne du développement de la fonctionnalité *DTMO* d'un véhicule autonome jusqu'à son implémentation sur une plateforme embarquée. Cette chaîne se décompose en :

– **Les entrées** : incluent les capteurs utilisés qui sont la caméra, le LIDAR et le radar permettent de détecter des obstacles de natures différentes tels que les piéton, les véhicules et les motos.

- **Protocoles de communication** : permettent de transférer les données réceptionnées par les capteurs, aux unités de calcul, à travers le réseau embarqué du véhicule. Nous citons le protocole Ethernet pour assurer l'acquisition de données de caméra ainsi que de LIDAR et le protocole CAN pour le radar.
- **Processus de calibration des capteurs** : permet d'unifier tous les repères capteurs utilisés dans un seul référentiel. Ce processus est effectué avant la mise en service du véhicule. L'entrée de ce bloc est un ensemble de mesures des capteurs et la sortie sera les paramètres des calibrations.
- **Processus de fusion multi-capteurs pour le *DTMO*** : consiste à fusionner les différentes données issues des capteurs. Ce processus a pour objectif d'éviter toutes collisions avec les obstacles au cours de la navigation. Notons que ce processus et celle de calibration sont développés avec des langages de programmation de haut niveau tels que le C/C++ et le Matlab.
- **Synthèse de haut niveau** : afin que la plateforme embarquée accueillie le processus de fusion multi-capteurs pour la fonctionnalité *DTMO*, il est nécessaire de synthétiser ce processus, écrit en langages de programmation de haut niveau, en langage de description matérielle.
- **Plateforme embarquée** : c'est la plateforme cible qui accueillera la perception du véhicule autonome, en particulier la fonctionnalité de *DTMO*, aussi, elle a pour rôle de gérer les flux de données.

Notons qu'il existe d'autres éléments et fonctionnalités qui interagissent avec l'environnement du véhicule mais nous avons présenté seulement ceux qui sont inclus dans le périmètre de ce travail de thèse. Dans la suite, nous présentons un état de l'art concernant la calibration des capteurs caméra/LIDAR, la fusion multi-capteurs pour la fonctionnalité de DTMO et les solutions existantes des plateformes embarquées .

II.3 Calibration extrinsèque entre les capteurs caméra et LIDAR

Dans cette section nous commençons par introduire la problématique et les défis de calibration entre les capteurs caméra et LIDAR. Ensuite nous présentons les

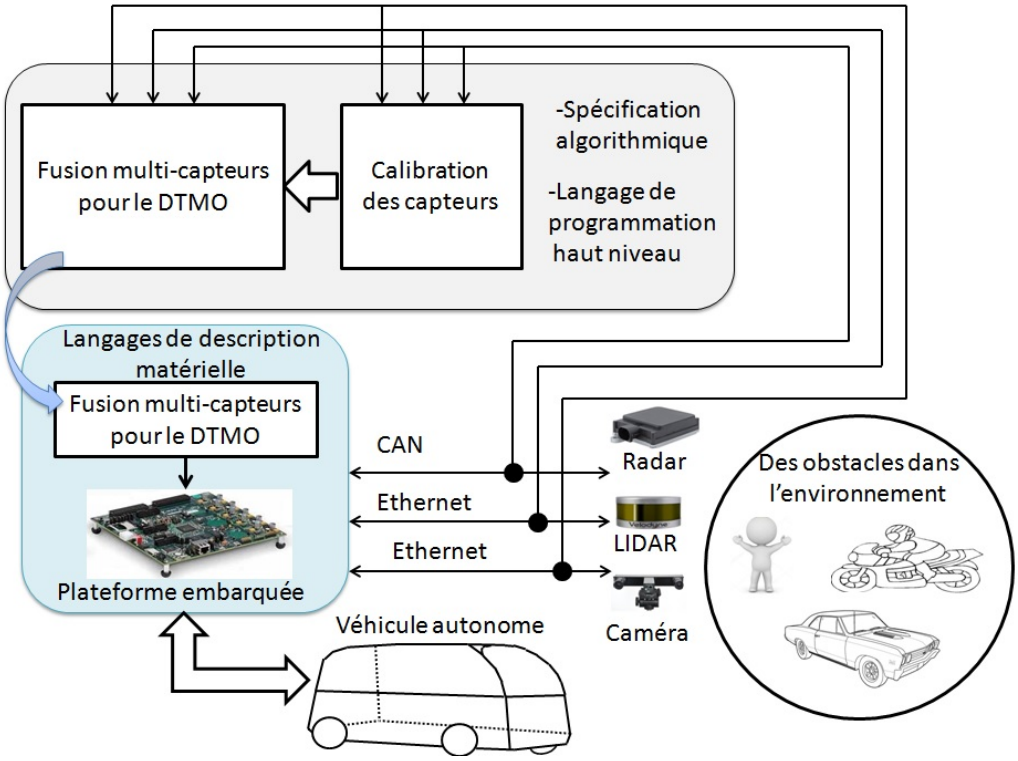

FIGURE II.1: Chaîne du développement de la fonctionnalité *DTMO*

travaux existants pour effectuer ce processus en mettant l'accent sur les approches proposées pour résoudre le problème géométrique de calibration.

II.3.1 Formulation du problème de calibration des capteurs

La calibration extrinsèque des capteurs caméra/LIDAR est le processus d'estimation de la transformation rigide entre les deux repères capteurs [32]. Il s'agit de déterminer la relation entre les repères des capteurs pour assurer le passage d'un repère à un autre. La figure II.2 présente les repères que nous trouvons dans un véhicule :

- Repère capteur : Chaque capteur a son propre repère tel que le repère de caméra ou celui de LIDAR.
- Repère véhicule : Pour unifier tous les repères de différents capteurs utilisés, un repère véhicule est souvent fixé.

Chaque deux repères sont liés entre eux par une transformation rigide qui se décompose en une matrice de rotation R et un vecteur de translation $\overrightarrow{t}$ illustrée par

l'équation suivante :

$$\overrightarrow{p}_d = {}^dR_s \, \overrightarrow{p}_s + {}^d\overrightarrow{t}_s \tag{II.1}$$

avec $\overrightarrow{p}_s$ est un point donné exprimé dans le repère source, $\overrightarrow{p}_d$ son correspondant exprimé dans le repère destination.

Dans notre cas, nous nous intéressons à la calibration entre le capteur caméra et celui de LIDAR. Le défi consiste à estimer la transformation rigide permettant la projection des points $3D$, exprimés par rapport au repère LIDAR, dans le repère $2D$ de caméra. Pour ce faire, le principe est de placer une mire de calibration dans les champs de vision des capteurs caméra et LIDAR (figure II.2). L'étape suivante consiste à extraire les caractéristiques de la mire dans les deux repères capteurs, à différentes positions et orientations, afin d'établir une relation mathématique qui permet de déterminer les paramètres de calibration.

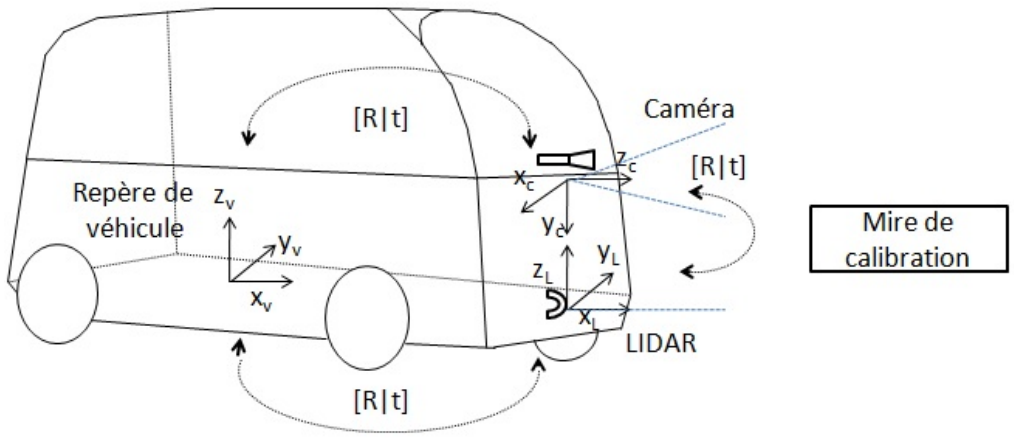

FIGURE II.2: Transformations entre les repères capteurs du véhicule

II.3.2 Défis de calibration

Afin de développer une approche de calibration, certains défis sont confrontés ce qui rend ce processus délicat. Nous pouvons distinguer les défis engendrés par les erreurs et les bruits de mesure tels que :

- Les incertitudes et les bruits des mesures des capteurs peuvent dégrader la qualité de calibration [22]. Pour palier à ce problème, il est prouvé dans [33,34], que l'augmentation du nombre de poses permet de réduire la répercussion de ces bruits.
- Certaines approches de calibration caméra/LIDAR nécessitent la calibration de caméra, en premier lieu, pour déterminer les paramètres intrinsèques de caméra tels que la distance focale et le centre optique. Ces derniers, affectés par des erreurs engendrées par le processus de calibration intrinsèque, seront utilisés pour estimer les paramètres de calibration caméra/LIDAR. Pour minimiser la répercussion des ces erreurs, des travaux ont été développés afin d'éviter de calculer séparément les paramètres intrinsèques de la caméra d'un côté et les paramètres extrinsèques de la caméra et de LIDAR d'autre côté [33, 35].

Par ailleurs, la calibration caméra/LIDAR confronte des défis qui sont liés à la mise en œuvre de ce processus, et qui sont :

- Le choix de la mire de calibration et ses caractéristiques telles que sa forme qui peut être triangulaire [36], rectangulaire [37], ou circulaire [38], ou elle peut être aussi un damier [34]. Ce choix est important car il précise les caractéristiques géométriques et visuelles qui seront détectées par les capteurs.
- L'établissement et la résolution d'une relation géométrique qui lie les deux repères capteurs qui sont hétérogènes. Cette relation est sous la forme d'une fonction objectif soumise à des contraintes.
- Afin d'automatiser ce processus, l'extraction des caractéristiques des capteurs doit être intelligente. Pour ce faire, les techniques de vision par ordinateur pour détecter les caractéristiques par la caméra (détection des points, lignes, plans...) et les processus de segmentation pour le LIDAR, sont souvent développés.

II.3.3 Calibration extrinsèque Caméra/LIDAR

Plusieurs méthodes existent pour la calibration extrinsèque caméra/LIDAR. Le tableau II.1 présente une comparaison entre différentes approches de calibration caméra/LIDAR. Ces approches utilisent des LIDARs soient mono-nappe ou multi-

nappes (plusieurs lignes de balayage) et des caméras monoculaire ou stéréo (apporter la profondeur de la scène).

Nous pouvons distinguer différents types de mires de calibration tels que triangulaire [39], rectangulaire [36] ou polygone [33]. Également, la mire de calibration peut être un damier [34] pour déterminer à la fois les paramètres intrinsèques de caméra et les paramètres extrinsèques de calibration entre la caméra et le LIDAR. De plus, une approche d'auto-calibration a été proposée dans [40] sans l'utilisation d'une mire. Le principe est de faire l'association entre une carte de distance générée par le LIDAR et la carte de disparité générée par la caméra stéréo. Cette catégorie est souvent favorable car le véhicule fait l'auto-calibration sans aucune intervention mais par contre elle est plus complexe en termes de développement et mise en œuvre.

Dans le but de réduire les bruits de mesure, le processus de calibration est souvent optimisé avec une approche d'optimisation non linéaire itérative telle que Levenberg-Marquardt comme dans [38] ou la méthode de Gradient dans [41].

En pratique, il est difficile d'obtenir ou d'évaluer la vérité terrain des paramètres extrinsèques entre les capteurs caméra et LIDAR. Certains travaux définissent leurs propres critères pour évaluer la qualité de calibration. Dans [34], un indicateur de précision est défini en se basant sur les résultats de projection des points LIDAR dans le plan image. De même dans [42], un critère de performance, reposant sur l'erreur de projection des points LIDAR, a été défini.

II.3.4 Extraction des caractéristiques

Une autre tâche importante pour développer une approche de calibration caméra/LIDAR est l'extraction des caractéristiques. Certains travaux ont présenté des chaînes complètes pour l'extraction des donnés de la mire ou de la scène, dans le cas d'auto-calibration, par les deux capteurs afin de réduire l'intervention de l'utilisateur dans la mesure du possible. Pour ce faire, la chaîne de traitement de caméra utilise les techniques de vision par ordinateur pour détecter les caractéristiques telles que la détection des points, lignes ou plans. Quant au LIDAR, l'extraction de données s'appuie sur le processus de segmentation. De plus, nous trouvons souvent des étapes de filtrage et optimisation pour les deux chaînes d'extraction afin de garder

Approche	Mire de calibration	Type de LIDAR	Type de caméra	Calibration intrinsèque de caméra	Application	Optimisation	Indicateur de performance
[38]	Circulaire	4 nappes	Stéréo	Oui	Véhicule intelligent	Levenberg-Marquardt	Oui
[39]	Triangulaire	1 nappe	mono-culaire	Oui	Robotique	Levenberg-Marquardt	Oui
[36]	Triangulaire	4 nappes	Mono-culaire	Oui	Véhicule intelligent	Levenberg-Marquardt	Oui
[37]	Rectangulaire	1 nappe	Mono-culaire	Oui	Robotique	Non	Oui
[35]	Rectangulaire	1 nappe	Mono-culaire	Non	Robotique	Levenberg-Marquardt	Non
[34]	Damier	1 nappe	Stéréo	Oui	Véhicule intelligent	Levenberg-Marquardt	Oui
[33]	Polygone	32 nappes	Mono-culaire	Non	Robotique	Levenberg-Marquardt	Oui
[41]	Rectangulaire	64 nappes	Mono-culaire	Oui	Véhicule intelligent	Méthode de Gradient	Non
[40]	Sans mire	32 nappes	Stéréo	Oui	Véhicule intelligent	Optimisation par essaims particulaires	Oui
[42]	Sans mire	32 nappes	Mono-culaire	Oui	Véhicule intelligent	Méthode itérative	Oui

TABLE II.1: Comparaison entre des approches de calibration caméra/LIDAR

les données pertinentes et éliminer celles qui sont aberrantes.

Notons que l'extraction repose sur la forme spéciale de la mire. Par exemple dans [33], les caractéristiques à extraire sont les quatre sommets d'une mire sous la forme d'un polygone. La caméra détecte les sommets avec l'algorithme FAST pour la détection de coins [43] et le LIDAR les détecte avec un processus de segmentation qui se base sur l'algorithme de RANdom SAmple Consensus (RANSAC) [44]. Pour les approches d'auto-calibration, les caractéristiques sont souvent des donnée pertinentes de la scène de route telles que les lignes de marquage au sol [42].

II.3.5 Résolution du problème de calibration

Le problème de calibration extrinsèque Caméra/LIDAR est généralement résolu en établissant des contraintes géométriques à partir de l'association de différentes caractéristiques entre les mesures observées simultanément dans les deux repères capteurs. La deuxième étape consiste à résoudre ce problème géométrique. Nous pouvons classifier les méthodes pour résoudre le problème de calibration en trois catégories :

(i) La première approche [35] résout le problème géométrique avec une méthode linéaire telle que la Décomposition en Valeurs Singulières (SVD), et utilise

ensuite cette solution comme une première approximation pour appliquer une optimisation non-linéaire itérative telle que les algorithmes de Gauss-Newton [45] ou de Levenberg-Marquardt. Autrement dit, la première approximation sera mise à jour d'une manière itérative dans le but d'être optimisée.

(ii) La deuxième approche [37] consiste à résoudre analytiquement un problème de minimisation représenté par une fonction objectif.

(iii) La troisième approche [39] combine les deux premières, c'est à dire une résolution analytique suivie par une optimisation non-linéaire.

II.3.6 Synthèse

Nous avons présenté dans cette section la calibration extrinsèque caméra/LIDAR qui a pour but d'estimer la transformation rigide permettant d'assurer le changement de repère d'un capteur à un autre. Ce problème géométrique, qui établit une association entre les caractéristiques des deux capteurs, doit être formulé, résolu et optimisé afin de déterminer les paramètres extrinsèques de calibration.

Nous pouvons distinguer deux catégories de calibration caméra/LIDAR : la première catégorie utilise une mire pour estimer les paramètres ; et la deuxième, celle d'autocalibration, utilise les caractéristiques pertinentes de la scène telles que le marquage au sol de la route. La deuxième catégorie est une solution favorable car le véhicule effectue l'auto-calibration sans aucune intervention mais par contre elle est plus complexe en termes de développement, mise en œuvre et nécessite un LIDAR multi-nappes qui permet de fournir des caractéristiques $3D$ plus riches en termes d'information. Pour le LIDAR mono-nappe, ce qui est notre cas, l'utilisation d'une mire est nécessaire pour estimer les paramètres de calibration. En outre, l'extraction des caractéristiques des capteurs doit être intelligente afin d'automatiser la calibration et réduire l'intervention de l'utilisateur. Cette étape nécessite le développement des algorithmes de vision par ordinateur et de segmentation.

Pour conclure, nous distinguons trois paramètres à la sortie de ce processus, qui sont la transformation rigide (rotation et translation), un indicateur de performance pour évaluer la qualité de calibration, et une erreur de calibration qui sera injectée

dans les incertitudes lors de la prochaine étape, celle de fusion de données.

II.4 Perception pour le véhicule autonome : fonctionnalités et fusion multi-capteurs

Après avoir unifié tous les repères capteurs dans un même référentiel, celui de véhicule (voir figure II.2), l'étape suivante consiste à fusionner les différentes données issues des capteurs de véhicule autonome. Dans cette section, nous présentons les principaux outils théoriques pour fusionner les données. Ensuite, un tour d'horizon de perception de véhicule autonome est présenté incluant ses fonctionnalités principales et son évolution vers les architectures multi-capteurs qui implique la fusion de données.

II.4.1 Approches de fusion multi-capteurs

II.4.1.1 Approche probabiliste : Fusion Bayésienne

La fusion bayésienne s'appuie sur le théorème de Bayes. Ce dernier, est un résultat important dans l'étude des modèles probabilistes [1]. Considérons deux variables aléatoires x et z que nous cherchons à définir leur fonction de densité de probabilité conjointe notée $P(x, z)$. Soient :

- $P(x|z)$ **:** La distribution a posteriori qui décrit les vraisemblances associées à x sachant l'observation z.
- $P(z|x)$ **:** Les observations récupérées sont modélisées comme étant des fonctions de densité de probabilité conditionnelle (probabilité de z sachant x).
- $P(x)$ **:** La probabilité a priori de x.
- $P(z)$ **:** La distribution marginale ou a priori de z qui sert à normaliser la distribution postérieure.

Le théorème de Bayes [1] est :

$$P(x|z) = \frac{P(z|x)P(x)}{P(z)} \tag{II.2}$$

Il est possible d'appliquer le théorème de Bayes pour fusionner un ensemble d'observations provenant de plusieurs sources différentes. Soit Z^n l'ensemble de mesures suivant :

$$Z^n \triangleq \{z_1 \in Z_1, ..., z_n \in Z_n\} \tag{II.3}$$

Le but est d'utiliser cette information (mesure) pour construire une distribution a posteriori $P(x|Z^n)$ qui décrit les vraisemblances relatives des différentes valeurs de l'état d'intérêt $x \in X$ compte tenu des informations obtenues. Le théorème de Bayes peut être directement utilisé pour calculer cette fonction de distribution à partir de :

$$P(x|Z^n) = \frac{P(Z^n|x)P(x)}{P(Z^n)} = \frac{P(z_1, ..., z_n|x)P(x)}{P(z_1, ..., z_n)} \tag{II.4}$$

L'équation II.4 peut être réduite [1] de la manière suivante :

$$P(x|Z^n) = [P(Z^n)]^{-1}P(x)\prod_{i=1}^{n} P(z_i|x) \tag{II.5}$$

L'équation II.5 présente un mécanisme simple et direct pour calculer la vraisemblance relative suivant différentes valeurs d'un état à partir d'un nombre quelconque d'observations ou d'autres informations.

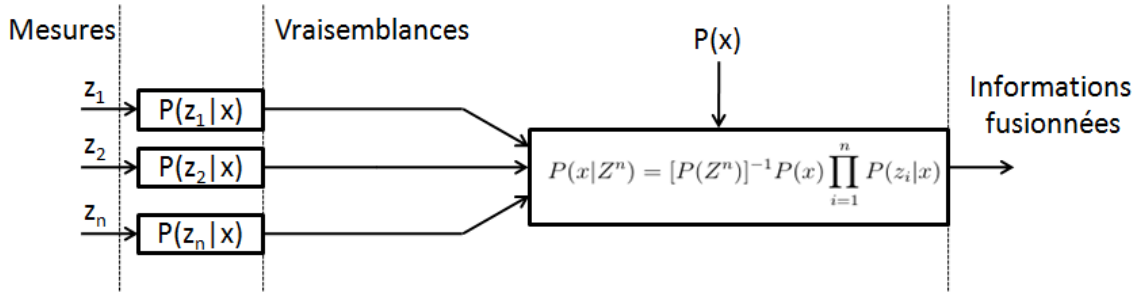

FIGURE II.3: Fusion Bayésienne [1]

II.4.1.2 Approche crédibiliste : Théorie de l'évidence

La théorie de Dempster-Shafer [31] est une théorie mathématique basée sur la notion de preuves utilisant les fonctions de croyance et le raisonnement plausible. La théorie de Dempster-Shafer, appelée aussi la théorie des fonctions de croyance, est une généralisation de la probabilité bayésienne. Le domaine des méthodes probabilistes est "*tous les sous-ensembles possibles*", tandis que le domaine du raisonnement plausible est "*tous les ensembles de sous-ensembles*" [46]. La théorie de probabilité fait intervenir la notion de probabilité a priori ce qui est gênant lorsque nous ne pouvons pas l'évaluer. La théorie des fonctions de croyance nous permet d'éviter cette notion. De plus, elle permet de modéliser les imprécisions des mesures et elle prend en compte qu'une hypothèse peut être en dehors du cadre d'étude. Par ailleurs, pour combiner les informations provenant de différentes sources d'informations l'approche bayésienne utilise la règle de Bayes, tandis que l'approche crédibiliste utilise d'autres règles de combinaison telles que la conjonctive, Dempster, Cautious [47], et chacune a ses propres critères d'utilisation.

II.4.1.3 Estimation multi-capteurs avec le filtre de Kalman

Le problème posé est l'estimation de l'état réel inconnu d'un environnement donné à partir d'une suite de mesures qui est fournie par un groupe de capteurs. Notons que, définir et résoudre un problème d'estimation est toujours la clé d'un système de fusion de données réussi [1, 48].

Le filtre de Kalman est une solution souvent utilisée dans la fusion multi-capteurs et en particulier dans la perception de véhicule autonome. Il s'agit d'un estimateur récursif [8]. L'état du filtre est représenté par deux variables qui sont l'état estimé et la matrice de covariance de l'erreur. Le filtre de Kalman comporte deux phases distinctes :

(i) *Phase de prédiction* permettant de prédire l'état estimé courant du système à partir de l'état précédent. Cette phase est caractérisée par une grande incertitude.

(ii) *Phase de correction* (mise à jour de mesure) en utilisant les mesures qui per-

mettent de calculer l'erreur de prédiction.

Notons que nous pouvons distinguer plusieurs types de filtre de Kalman, tels que :

- **Le filtre de Kalman conventionnel (linéaire) :** Pour utiliser ce filtre, il faut vérifier les deux hypothèses suivantes : **(i)** Le système étudié doit être linéaire **(ii)** Les bruits de modèles et de mesure doivent être Gaussiens et aussi indépendants entre eux. L'algorithme de ce filtre est détaillé dans l'annexe A.
- **Le filtre de Kalman étendu :** L'emploi de ce filtre est dans le cas où le modèle de prédiction d'état ou celui de mesure ne sont pas linéaires. Ce filtre, vise à résoudre un tel problème en linéarisant les fonctions de transition d'état non linéaires.

II.4.2 Perception pour le véhicule autonome

Dans le contexte du véhicule autonome, Pendleton [13] a défini la perception comme suit : "*la perception implique l'obtention des données sur l'environnement à l'aide des capteurs, traiter ces données et déduire une représentation interne (modèle interne) de cet environnement externe*". Pendant ce processus, le véhicule a besoin d'estimer sa propre position dans l'environnement, estimer les positions relatives des objets statiques ainsi que les objets mobiles afin de les suivre au fil du temps.

Les deux fonctionnalités principales de perception pour un véhicule autonome sont (voir figure II.4) :

(i) Simultaneous Localization and Mapping ($SLAM$) [14] : C'est la fonctionnalité de localisation et cartographie simultanées. À partir des données mesurées avec un ensemble de capteurs proprioceptifs et extéroceptifs, le $SLAM$ permet au véhicule à la fois de se localiser et cartographier l'environnement.

(ii) Detection and Tracking of Moving Objects ($DTMO$) [15] : Cette fonctionnalité permet de détecter et suivre les obstacles statiques et dynamiques autour de véhicule.

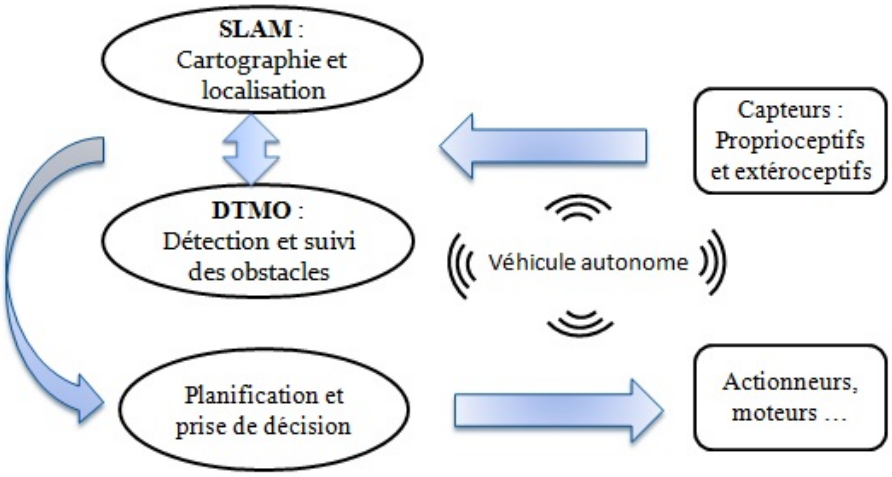

FIGURE II.4: Perception pour un véhicule autonome

Ces deux fonctionnalités envoient des informations au bloc de contrôle pour assurer la prise de décision et la planification de trajectoire en sécurité et en adéquation avec l'objectif du déplacement.

Dans [2, 3], une modélisation du problème de perception est proposée en s'appuyant sur les fonctionnalités de $SLAM$ et de $DTMO$. La figure II.5 illustre graphiquement cette modélisation du problème de perception.

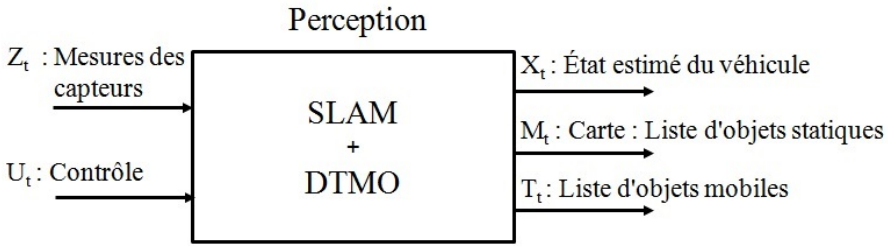

FIGURE II.5: Modélisation de perception du véhicule autonome [2, 3]

Les entrées de cette modélisation sont : (i) Z_t les mesures des capteurs (LIDAR, radar,...) accumulées jusqu'à l'instant t. (ii) U_t l'entrée du contrôle équivalente aux mesures du mouvement données par un odomètre ou une Unité de Mesure Inertielle (IMU) :

$$Z_t \triangleq z_{0:t} = \{z_0, z_1, ..., z_t\} \tag{II.6}$$

$$U_t \triangleq u_{1:t} = \{u_1, u_2, ..., u_t\} \tag{II.7}$$

où z_i est l'entrée du capteur et u_i la mesure du mouvement à l'instant t.
La première sortie X_t est l'ensemble d'états estimés du véhicule (position et orientation) jusqu'à l'instant t. Donc, si x_i est l'état du véhicule à l'instant i, on a :

$$X_t \triangleq x_{0:t} = \{x_0, x_1, ..., x_t\} \tag{II.8}$$

Notons que l'absence de u_0 signifie que la première mesure du mouvement n'est disponible que lorsque le véhicule roule de l'état x_0 à x_1.
La seconde sortie, le map M_t est l'ensemble d'objets statiques détectés à l'instant t :

$$M_t = \{m_0, m_1, ..., m_K\} \tag{II.9}$$

où K est le nombre total d'objets dans l'environnement et chaque m_k spécifie la position et les propriétés de chaque objet dans l'environnement.
La dernière sortie T_t est l'ensemble des objets mobiles détectés o_i à l'instant t :

$$T_t = \{o_0, o_1, ..., o_L\} \tag{II.10}$$

où L est le nombre total d'objets mobiles suivis dans l'environnement.

Souvent dans la littérature de navigation autonome, les fonctionnalités $SLAM$ et $DATMO$ sont traitées comme des problèmes distincts. Cependant, des travaux de recherche récents ont proposé plusieurs méthodes qui exécutent ces deux fonctionnalités simultanément et d'une manière indissociable [49, 50]. Wang et al. [51] ont développé la première approche qui combine le $SLAM$ et le $DTMO$ pour les véhicules autonomes et cette fusion a donné naissance à un nouveau concept qui est le Simultaneous Localization, Mapping, and Moving Object Tracking ($SLAMMOT$).

II.4.3 SLAM

Le *SLAM* est une fonctionnalité importante pour soutenir d'autres fonctionnalités comme la planification de trajectoire et le *DTMO*. Afin de détecter et suivre les objets mobiles autour d'un véhicule en mouvement, des informations de localisation précises sont nécessaires à établir. Cette précision est de l'ordre de 10 *cm* selon le projet du véhicule autonome *Google car* [52].

Le *SLAM* comprend l'estimation simultanée de l'état du véhicule et la construction d'un modèle de "map" (carte) de l'environnement inconnu perçu par les capteurs. L'état du véhicule est souvent décrit par sa pose (position et orientation), bien que d'autres paramètres peuvent être inclus dans l'état, tels que la vitesse et les paramètres de calibrations [53]. D'autre part, la carte est une représentation des éléments d'intérêt (position des repères, obstacles) décrivant l'environnement dans lequel la plateforme navigue. Donc, cette fonctionnalité est composée d'un ensemble de méthodes permettant au véhicule autonome de construire une carte d'un environnement et en même temps de se localiser en utilisant cette carte.

II.4.3.1 Représentation de la carte

Afin de construire une carte, Hamzaoui [54] a analysé trois approches fondamentales de représentation de l'environnement, qui sont :

(i) *L'approche directe* [55] : utilise les données brutes des mesures du capteur pour représenter l'environnement sans aucune extraction d'amers ou de caractéristiques prédéfinies (lignes, coins...). Elle est souvent adapté à l'utilisation de capteur LIDAR. Dans ce cas, la carte sera représentée par un nuage de points.

(ii) *L 'approche basée sur les caractéristiques géométriques (feature-based)* [56] : réduit les données des mesures en caractéristiques prédéfinies. Elle utilise des primitives géométriques telles que des lignes, des coins ou des cercles. Ensuite, la création de la carte consiste à estimer les paramètres des primitives afin qu'ils correspondent au mieux aux observations. Cette approche est incompatible avec les environnements trop complexes et non structurés.

(iii) *L 'approche basée sur une grille d'occupation (grid-based)* [57] : Dans cette

représentation, l'environnement est divisé en cellules rectangulaires. Il s'agit d'une discrétisation de l'espace en cellules régulières. Outre de cette discrétisation, une mesure de probabilité d'occupation est estimée pour chaque cellule indiquant si celle-ci est occupée ou non.

II.4.3.2 Le SLAM pour les véhicules autonomes

Pour qu'un véhicule autonome se localise en ville, la majorité des solutions commercialisées se base sur les systèmes de positionnement par satellites GPS. Même si ces systèmes offrent une précision suffisante hors agglomération, celle-ci se dégradent considérablement en villes, dans un milieu urbain, à cause des phénomènes connus sous le nom du canyon urbain (réflexion du signal GPS sur les façades des bâtiments). Pour surmonter ce problème, les solutions basées sur d'autres capteurs semblent une alternative prometteuse [58] tels que le LIDAR [55], le radar [59] et la caméra [60].

Le $SLAM$ basé sur le LIDAR consiste à faire la correspondance entre les données du nuage de points déjà acquis par le LIDAR et celle de la carte connue a priori (données qui sont déjà enregistrées) pour obtenir la position du véhicule dans la carte [58]. Moosmann et al. [55] ont proposé un algorithme de $SLAM$ en utilisant le LIDAR $VelodyneHDL-64E$. Leur objectif est de créer une carte quadrillée (*gridmap*) en $3D$ en minimisant les bruits qui peuvent affecter la précision de la carte grâce à la haute résolution offerte par le capteur utilisé. Cependant, l'utilisation d'un tel capteur qui est onéreux, nécessite des calculateurs performants pour traiter la quantité d'information (nuage de points) réceptionnée. Afin de réduire cette quantité d'information à traiter, Javanmardi et al. [58] ont présenté une technique de localisation pour les véhicules autonomes dans un milieu urbain (dédiée aux capteurs LIDAR 64 nappes). L'approche utilisée pour la création de la carte est celle de directe, de plus, ils extraient les surfaces planes et les enregistrent comme des cartes a priori. De ce fait, la taille de la carte globale a été réduite (25 millions points pour 1000 plans).

Dans [59], une approche nommée *RoughCough* est développée permettant au véhicule de se localiser avec le radar. Le principe est de créer une carte quadrillée

à partir de la mise en correspondance entre les mesures du radar et les points des repères (landmark) déjà enregistrés dans la base de données. Par contre, la résolution du radar est moyenne, c'est qui peut affecter la précision de localisation.

À part le LIDAR et le radar, la caméra assure aussi la fonctionnalité de localisation, connue sous le nom de $SLAM$ visuel. Il s'agit de toutes approches de $SLAM$ qui prennent des images en entrées pour que le véhicule se localise. La différence par rapport aux approches déjà mentionnées est la nécessité de calculer la profondeur à partir d'images consécutives. Grâce aux progrès faits en vision par ordinateur (détection et suivi de points d'intérêt), Davison et al. [61] ont développé la première implémentation temps réel de $SLAM$ visuel monoculaire en 2003. Pire et al. [60] ont utilisé une caméra stéréoscopique permettant de fournir la profondeur afin d'améliorer l'analyse de l'environnement et assurer la localisation.

Par ailleurs, la fusion multi-capteurs augmente à la fois la fiabilité et la précision des observations environnementales utilisées pour le problème de $SLAM$ [62]. En fait, fusionner deux sources d'information de profondeur, donne des résultats plus performants par rapport à celui d'un seul capteur pour le $SLAM$ [63].

II.4.4 DTMO avec une architecture mono-capteur

Dans la littérature de perception pour les véhicules autonomes, certains travaux ont effectué la tâche de détection et de suivi d'objets avec un seul type du capteur. Dans [64], Jian et al. ont proposé un algorithme pour la détection des véhicules en utilisant un capteur LIDAR (64 nappes) dans un environnement urbain complexe. Cet algorithme résout la segmentation et la classification du nuage des points avec une machine à vecteurs de support. La détection peut être aussi effectuée avec une caméra mono-vision pour la détection des véhicules [65] ou des piétons [25], avec une caméra stéreo [66] ou avec des caméras "fisheye" [67] pour garantir une perception de 360° autour du véhicule. Également, Liu et al. [68] ont proposé une approche de $DTMO$ avec le radar. Cependant, ces approches mono-capteur, présentent des limitations car un seul capteur n'est pas suffisant pour garantir une perception précise si le capteur a une grande incertitude, fournit de mesures insuffisantes et bruitées ou s'il a une faible résolution et un champ de vision limité.

II.4.5 DTMO : De mono-capteur vers multi-capteurs

Parmi les capteurs les plus utilisés dans les véhicules autonomes, nous citons le LIDAR, le radar et la caméra. Nous montrons dans ce qui suit que chaque capteur a ses propres avantages et ses inconvénients, c'est pourquoi seule une fusion multi-capteurs permet d'améliorer la précision et la fiabilité de détection.
Le tableau II.2 présente une étude comparative de performance des principaux capteurs qui sont utilisés dans les véhicules autonoms [9]. Selon ce tableau, la caméra a une bonne performance en termes de résolution latérale et l'insensibilité au bruit. Par contre, elle a une mauvaise performance pour fonctionner dans toutes les conditions métrologiques ainsi que pour l'estimation des vitesses des obstacles. En ce qui concerne le LIDAR, il a une bonne résolution longitudinale et latérale, et une performance moyenne pour l'estimation des vitesses. Cependant, c'est un capteur qui est sensible aux conditions métrologiques et aux bruits. Quant au radar, il a une bonne résolution longitudinale et permet de fournir des estimations précises pour les vitesses des obstacles détectés mais il est sensible aux bruits.

	Caméra	LIDAR	Radar
Portée de détection	●●●○○	●●●●○	●●●●●
Résolution longitudinale	●●●○○	●●●●●	●●●●●
Résolution latérale	●●●●●	●●●●●	●●●○○
Estimation de vitesse	○○○○○	●●●○○	●●●●●
Bruits	Insensible	Sensible	Sensible
Conditions météorologiques	Sensible	Sensible	Insensible
Éclairage	Sensible	Insensible	Insensible

TABLE II.2: Tableau comparatif des performances des capteurs : caméra, LIDAR et radar [9]

Une autre étude a été faite par Josip et al. [27] pour comparer ces capteurs. Cette étude a favorisé aussi le capteur radar pour estimer les positions des obstacles ainsi que leurs vitesses relatives et pour son bon fonctionnement dans toutes les conditions métrologiques (plus fiable à la pluie, au brouillard, à la neige). Par contre, il a l'inconvénient de produire une quantité significative d'échos parasites. Pour le capteur LIDAR, il permet d'estimer les positions latérales des obstacles avec une

précision plus efficace que celle du radar. Il peut aussi estimer la zone d'occupation d'un objet et fournir une représentation détaillée de la scène [69]. Pour les capteurs de vision, une caméra stéréoscopique fournit une détection de cible avec une haute résolution latérale, mais avec une précision en profondeur moins certaine.
Compte tenu de la complémentarité des capteurs caméra, LIDAR et radar, cette combinaison est couramment utilisée pour les applications des véhicules autonomes. Il est donc important de développer des approches de fusion multi-capteurs pour tirer profit de chacun des capteurs.

II.4.6 DTMO avec une architecture multi-capteurs

La fonctionnalité $DTMO$ se compose de deux sous-tâches principales qui sont la détection et le suivi. Le module de détection envoie à celui du suivi une liste d'objets détectés avec leurs informations telles que les positions ou les classes (piéton, vélo, moto ou véhicule). Le deuxième module, assure le suivi des objets détectés. Notons qu'un objet qui est suivi au cours du temps est appelé une piste (track).
Après la détection, il faut distinguer les objets mobiles de ceux qui sont statiques de la carte afin d'estimer leurs états futurs. Plusieurs approches ont été développées pour détecter les objets mobiles suivant les types des capteurs. Pour la détection qui repose sur les capteurs de vision, nous distinguons principalement trois approches pour détecter les objets mobiles qui sont : approche basée sur le mouvement, approche basée sur l'apparence et celle basée sur une caméra stéréoscopique [70]. La détection d'un objet mobile peut se déduire dans le cas d'utilisation des capteurs de télédétection qui mesurent les distances ou qui renvoient les vitesses relatives des objets détectés tels que le LIDAR ou le radar, et selon les valeurs des vitesses nous pouvons classifier les objets mobiles de ceux statiques de la carte.
D'autre coté, l'emploi d'une architecture multi-capteurs, pour la fonctionnalité de $DTMO$, permet d'améliorer la précision et la fiabilité de perception afin d'éviter les fausses alarmes et réduire les détections manquées dans un environnement incertain. Mais avant tout, la question qui se pose, quelles informations faut-il fusionner ? Pour répondre à cette question, nous classifions les informations à fusionner en deux catégories : la première pour fusionner les positions des obstacles détectés et la

deuxième est celle de fusion d'hypothèses, qui sont déduites à partir des mesures, pour déterminer une probabilité de détection. Dans la suite, nous détaillerons ces deux catégories.

II.4.6.1 Fusion de positions des obstacles détectés

L'objectif de cette fusion est d'améliorer la détection des positions des obstacles en s'appuyant sur les architectures multi-capteurs. Cette fusion permet de réduire les détections manquées et éviter les fausses alarmes.

Dans [71], une approche de fusion de haut niveau, est développée en utilisant le filtre de Kalman pour fusionner les postions d'obstacles détectés par les capteurs radar et LIDAR. Baig et al. [72] ont proposé une approche de fusion de positions d'obstacles par la technique Bayésienne. Ils ont prouvé qu'une architecture multi-capteurs, qui comporte une caméra stéréo et un LIDAR, a permis de réduire les détections manquées par rapport à une architecture mono-capteur.

Dans un contexte de robotique embarquée, Salina et al. [73] ont proposé une approche basée sur le théorème de central limite [74] pour fusionner les positions d'obstacles détectés par une caméra infrarouge et un capteur ultrasonique. Notons que pour fusionner les mesures avec cette technique, il suffit de les pondérer suivant leurs matrices de covariance des bruits. Dans [75], les positions sont fusionnées en utilisant le filtre de Kalman, avec une phase de mise à jour séquentielle [1]. L'emploi d'une telle méthode est justifié par le fait que les temps d'acquisition des mesures, qui sont fournies par le LIDAR et le radar, sont asynchrones. Dans [27], Josip et al. ont proposé une approche, en utilisant le filtre de Kalman Étendu, pour fusionner les positions détectées par une caméra stéréo et un radar.

II.4.6.2 Fusion des hypothèses

Dans le contexte de robotique, Durrant [1] a présenté une approche du réseau bayésien pour fusionner les hypothèses provenant de différentes sources afin d'estimer les positions d'obstacles détectés. Pour appliquer cette méthode, les probabilités a priori doivent être connues. Premebida et al. [76] ont aussi utilisé la technique bayésienne pour combiner les hypothèses à l'issue de deux classifications effectuées

avec une caméra et un LIDAR, ce qui a permis d'obtenir une classification d'objets (piéton et véhicule) plus précise.

Par ailleurs, pour fusionner les différentes hypothèses qui sont déduites à partir de l'environnement du véhicule à travers les capteurs caméra et LIDAR, Fayad et al. [77] ont utilisé la théorie de croyance ou Dempster-Shafer. Dans [78], l'approche utilisée pour fusionner les différentes hypothèses déduites à partir de caméra, LIDAR et radar est celle de croyance. Le privilège de cette approche par rapport à celle de Bayésienne est qu'elle est très expressive permettant de représenter les différents niveaux d'ignorance, elle ne nécessite pas des probabilités a priori et elle gère les situations de conflit lorsque des preuves opposées apparaissent [76].

II.4.7 Niveaux de fusion de données pour le DTMO

La figure II.6 présente les fonctionnalités principales de perception et les trois niveaux possibles de fusion de donnés pour le $DTMO$, qui sont :

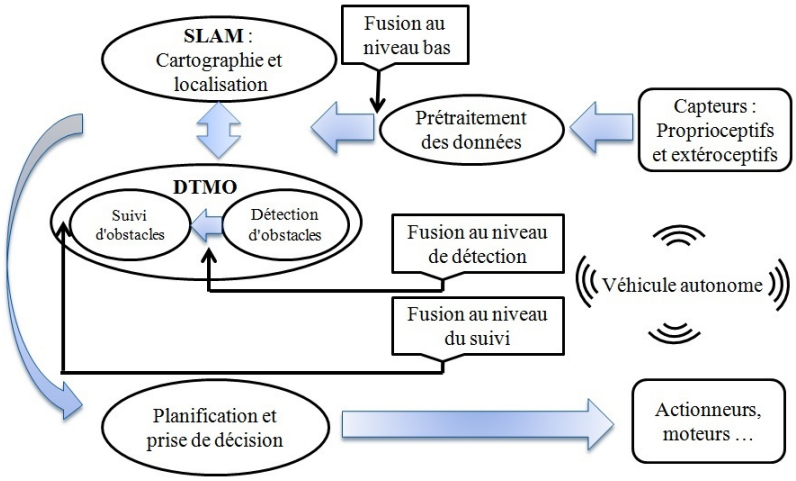

FIGURE II.6: Niveaux de fusion de données : bas, moyen et haut

(i) **Fusion de bas niveau :** Il s'agit de combiner plusieurs sources de données brutes ou pré-traitées pour produire de nouvelles données plus informatives et synthétiques que les entrées brutes. La fusion à ce niveau, permet d'avoir une représentation fusionnée de la carte qui sera utilisée par le $SLAM$ dans le but d'améliorer la détection de la fonctionnalité de $DTMO$.

Dans [79], une approche de fusion de bas niveau a été développée entre une caméra monoculaire et un LIDAR. Pour ce faire, chaque capteur fournit une grille d'occupation qui sera fusionnée avec l'autre pour obtenir une seule carte fusionnée. L'avantage de fusionner les données à ce niveau est la possibilité de classer les données à un stade très précoce en fusionnant les données brutes provenant de différents capteurs. Cependant, elle nécessite une bande passante de données élevée et une fréquence de traitement rapide et elle peut être complexe à mettre en œuvre dans la pratique [80]. En outre, l'architecture de fusion de bas niveau n'est pas modulaire, car l'ajout d'un nouveau capteur nécessite des modifications majeures dans le module de fusion.

(ii) **Fusion de moyen niveau (ou de caractéristiques) :** À ce niveau, les caractéristiques telles que les formes, les textures ou les positions seront fusionnées. Plus précisément, chaque capteur fournit une liste d'objets détectés qui sera fusionnée avec les autres pour avoir une seule liste fusionnée. Cette dernière, sera envoyée au module du suivi. La fusion à ce niveau est souvent utilisée pour fusionner les positions d'objets détectés ou leurs caractéristiques (classification) [27, 49]. L'avantage de ce genre de fusion est qu'il permet de réduire la bande passante des données transmises, car le temps nécessaire pour fusionner les caractéristiques extraites est inférieur à celui de données brutes. En outre, la modification d'une architecture de fusion de moyen niveau est moins complexe par rapport à celle du bas niveau.

(iii) **Fusion de haut niveau (ou de pistes) :** Chaque source d'entrée donne une décision, et toutes les décisions seront combinées. Il s'agit de la fusion de listes d'objets suivis. En effet, le module du suivi fournit des listes d'objets suivis qui seront combinées pour avoir une seule liste fusionnée de pistes. Dans [80], une approche de fusion de haut niveau a été développée entre des capteurs caméra, LIDAR et radar. L'avantage de ce niveau de fusion est son architecture modulaire.

II.4.8 Classification d'objets détectés

La précision de détection d'un objet en mouvement est essentielle pour une conduite autonome. Cette précision inclut la position et la classe de chaque objet détecté. La classe peut être un piéton, un vélo, une moto ou un véhicule. Tout d'abord, en incluant la classification d'objets, le modèle de l'environnement sera identifié avec plus de précision, ce qui augmente la fiabilité de perception. De plus, étant donné qu'un véhicule autonome circule dans un environnement urbain dynamique qui inclut différents types d'objets, il sera judicieux de classifier ces objets afin de prédire leurs états futurs. En fait, il est nécessaire de suivre tous les objets dans l'environnement du véhicule au cours de sa navigation, et reconnaître en plus les classes d'objets afin de les traiter selon leurs types [81]. À titre d'exemple, le temps nécessaire pour que le véhicule fait un freinage lorsque il détecte un obstacle, n'est pas le même entre un piéton et un véhicule.
La classification a pour objectif de réduire les faux positifs et les détection manquées, ce qui permet d'améliorer la détection et le suivi d'obstacles. Dans [49], une approche de classification a été développée et intégrée dans le module de $DTMO$ suivant deux niveaux : avant et après la tâche du suivi. Dans les deux cas, il est prouvé que la classification a amélioré la perception en réduisant les détections manquées. En plus, la classification améliore la précision d'association de données entre les mesures et les pistes dans la phase de suivi en cas de conflit [68].

II.4.9 Synthèse

Dans cette section, nous avons présenté un état de l'art concernant la perception pour le véhicule autonome. Cette perception se compose de deux fonctionnalités qui sont le $SLAM$ pour la localisation et cartographie simultanées, et le $DTMO$ pour la détection et le suivi d'obstacles. Dans cette thèse, nous nous intéressons au $DTMO$. Ce dernier, a été amélioré par l'utilisation des architectures multi-capteurs. En fait, les approches mono-capteur, présentent des limitations car un seul capteur n'est pas suffisant pour garantir une perception précise si le capteur a une grande incertitude, fournit de mesures insuffisantes et bruitées ou s'il a une faible résolu-

tion et un champ de vision limité. Donc, nous avons montré que chaque capteur a ses propres avantages et ses inconvénients, c'est pourquoi une fusion multi-capteurs permet d'améliorer la précision et la fiabilité de détection. Nous avons présenté les approches de fusion multi-capteurs qui ont prouvé l'amélioration de fiabilité de perception. Nous avons aussi identifié les niveaux possibles de fusion pour le $DTMO$, qui sont : bas, moyen et haut. Nous avons vu que la fusion de moyen niveau (ou de caractéristiques) telle que la fusion des positions des obstacles détectés est souvent favorisée grâce à sa performance et sa flexibilité d'implémentation. Une autre tâche importante pour garantir la précision de perception, est la classification qui permet de classifier les obstacles détectés dans l'environnement du véhicule au cours de sa navigation.

Par ailleurs, l'emploi d'une architecture multi-capteurs nécessite des calculateurs performants pour traiter et analyser les différentes données issues des capteurs ainsi que pour exécuter des algorithme intensifs tels que la détection des obstacles avec une caméra stéréo. En outre, étant donné que notre cible d'implémentation est une plateforme automobile autonome 100% électrique, nous visons à utiliser des calculateurs embarqués pour respecter les contraintes d'autonomie telles que la consommation d'énergie électrique.

II.5 Systèmes embarqués pour le véhicule autonome

Dans cette section nous présentons nos motivations principales pour concevoir un système embarqué à base de FPGA pour accueillir la perception d'un véhicule autonome et en particulier la fonctionnalité $DTMO$. Également, nous exposons les travaux existants et nos arguments concernant le choix de la solution de FPGA ainsi que les outils de synthèse de haut niveau.

II.5.1 Motivations : vers des calculateurs embarqués performants

L'intégration des logiciels à bord d'un véhicule hautement automatisé est un enjeu majeur. Un important effort de recherche est encore nécessaire sur plusieurs points, tels que la conception des architectures embarquées. La mise en œuvre de ces architecture permet d'améliorer les performances globales et la fiabilité du système tout en minimisant le coût de conception et de production. Ces performances s'articulent autour de trois axes principaux, à savoir la performance temporelle, la consommation d'énergie électrique et la sûreté de fonctionnement.

Le système embarqué dans le véhicule autonome doit être capable de traiter tout un tas d'informations diversifiées provenant des différents capteurs, que ce soit des caméras, radars et LIDAR. En effet, le traitement de différentes tâches de perception nécessite un calcul assez puissant pour atteindre une performance temporelle appropriée afin de définir la conduite à adopter par le véhicule.

La figure II.7 présente une détection d'un véhicule en utilisant l'algorithme de *Vehicle Detector* [82]. Cette implémentation, que nous avons fait sur un PC de processeur *Intel Core i*7 avec une fréquence de 3 GHz, traite une image toutes les six secondes. Ce temps d'exécution ne répond pas à l'exigence temporelle requise par notre domaine d'application.

De plus, la vitesse du véhicule au cours de conduite augmente les exigences du traitement. C'est à dire, plus que la vitesse du véhicule augmente plus que la perception aura besoin des calculateurs plus performants. La figure II.8 présente les distances d'arrêt du véhicule par rapport au débit de traitement des images pour différentes vitesses (étude réalisée par Texas Instruments) [4]. Rappelons que la distance d'arrêt est la somme de la distance parcourue pendant la détection du piéton et la distance parcourue pendant le freinage.
Les hypothèses de ces courbes sont : **(1)** La détection est un piéton **(2)** Format d'image *HD* **(3)** La distance de freinage est calculée selon le code de la route britannique **(4)** Sept images de latence pendant la détection des piétons : environ trois

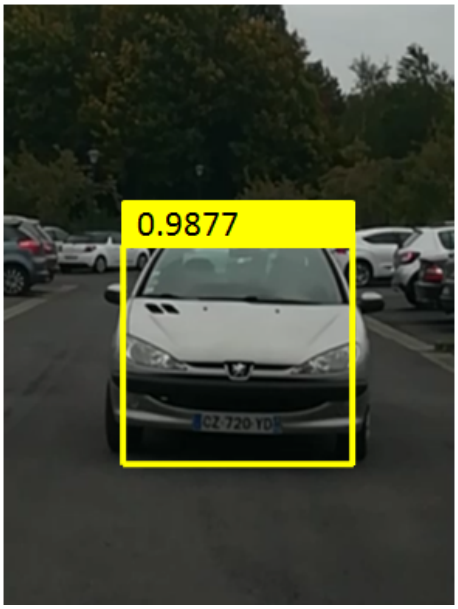

FIGURE II.7: Détection de véhicule avec l'algorithme de *Vehicle Detector* (Campus de Valenciennes)

images de latence de traitement et quatre images de latence de suivi afin d'améliorer la qualité de la détection. Ces courbes prouvent que l'augmentation de puissance du traitement diminue la distance d'arrêt. Par exemple, un véhicule qui roule avec une vitesse de 30 km/h a une distance d'arrêt de 12 m pour 10 fps et 7 m pour 30 fps.

Par ailleurs, un véhicule autonome consomme plus d'énergie électrique à cause de l'utilisation des capteurs supplémentaires tels que le LIDAR et la caméra ainsi que les calculateurs utilisés pour accueillir la perception. Selon une étude réalisée dans [83], la consommation d'énergie supplémentaire dépend beaucoup du type de système autonome utilisé. Elle peut varier de 2,8 à 4% dans les petits véhicules et elle peut augmenter jusqu'au 20% dans les systèmes les plus grands. Selon la même étude, environ de 40% de cette consommation supplimentaire d'énergie dans le véhicule autonome électrique vient des calculateurs. Aujourd'hui, le traitement des données est effectué par des PCs industriels assurant l'acquisition, la synchronisation et le traitement des données reçues. Cependant, les PC industriels consomment énormément d'énergie électrique, ce qui représente une contrainte importante pour les véhicules électriques qui nécessitent la minimisation de la consommation d'énergie

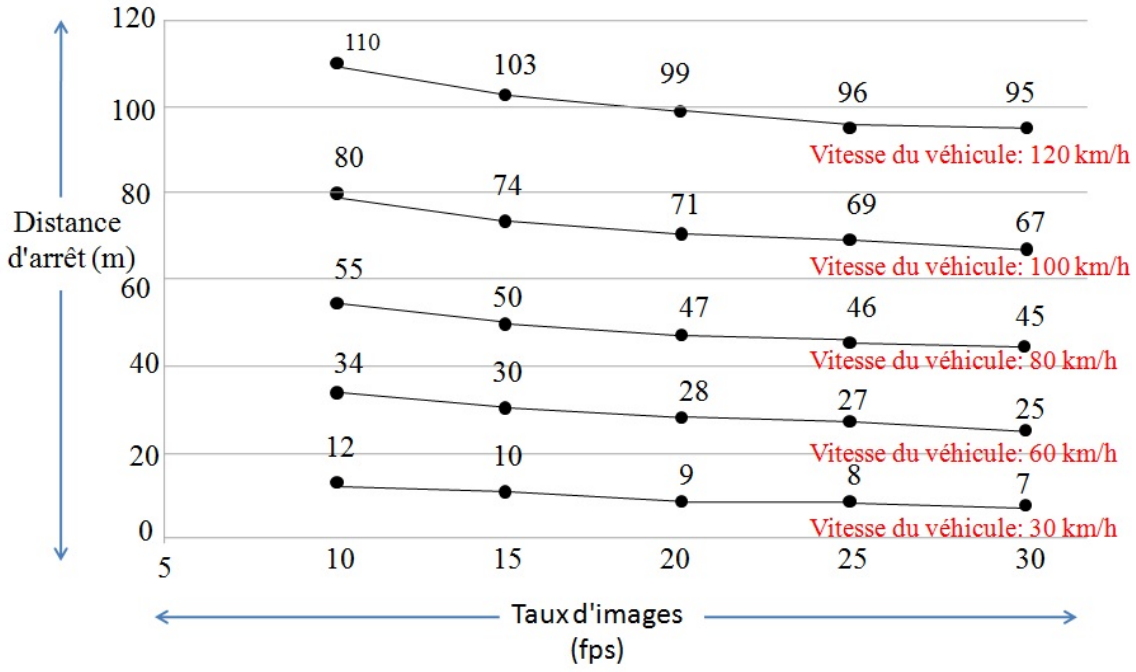

FIGURE II.8: Calcul de distance d'arrêt du véhicule par rapport au débit de traitement des images pour différentes vitesses [4]

.

de chaque élément du système [84]. Donc, afin de mettre en œuvre une perception adéquate du véhicule autonome, la tâche de calcul doit fournir simultanément des performances élevées, une consommation d'énergie réduite, à faible coût [85].

La sûreté de fonctionnement automobile est actuellement régie par la norme ISO 26262 : 2011 [86]. Cette norme permet d'établir un cadre et de définir des exigences sur les systèmes automobiles embarquées afin de garantir leur sécurité. C'est une norme émergente pour les systèmes de sécurité dans les véhicules routiers à moteur qui définit un cadre et un modèle d'application automobile. Elle traite aussi la sûreté de fonctionnement des systèmes électriques et électroniques incorporés au sein des véhicules routiers. Quant à la spécification $ASIL - D$ (Automotive Safety Integrity Levels), elle représente le degré de rigueur qui devrait être appliqué dans le développement, la mise en œuvre et la vérification d'une exigence afin d'éviter un risque résiduel déraisonnable dans le produit final (véhicule) [87]. La lettre D indique le niveau le plus critique de cette norme.

II.5.2 Systèmes sur puce : solutions et tendances

L'évolution très rapide des technologies de fabrication des circuits intégrés sur silicium, étant de plus en plus fiables et de plus en plus performantes, permet de réaliser des systèmes numériques entièrement intégrés sur une même puce.

Les besoins primordiaux de ce type de système apparaissent lorsque l'application de l'utilisateur a un besoin fort d'intégration de plusieurs composants d'une part et du calcul intensif d'autre part pour mettre en œuvre un système embarqué qui est soumis à des contraintes telles que la consommation d'énergie et la mémoire.

Pour fournir la puissance de calcul nécessaire, de nombreuses technologies sur le marché peuvent être utilisées comme solution, à savoir :

- **Unité centrale de traitement (CPU : Central Processing Unit)** : offre une plateforme pour l'implémentation logicielle des applications. Ce genre de circuit est standard, destiné au grand public et son coût du développement n'est pas cher. Cependant, il a une consommation d'énergie élevée et des performances plus faibles par rapport à celui du matériel [88]. Pour augmenter la puissance du traitement, la première solution était d'augmenter les fréquences de fonctionnement, mais cette solution a fini par atteindre ses limites à cause des problèmes de surchauffe et la consommation d'énergie. Donc, la technologie multi-cœur [89] a fait son apparition qui consiste à rassembler plusieurs cœurs physiques, dans une même puce, pour fonctionner simultanément afin d'exploiter le parallélisme logiciel. Cette solution a pour objectif de permettre à la CPU d'exécuter plusieurs tâches en parallèle pour réduire le temps d'exécution. Également, une autre solution a été proposée, celle des processeurs many-cœurs qui représente une évolution naturelle des processeurs multi-cœurs. Chaque CPU comporte de plusieurs dizaines à plusieurs centaines de cœurs de calcul reliés par des réseaux sur puce à faible latence. Cette solution offre un potentiel de performance, mais avec une hausse sensible de la complexité de programmation [90].
- **Application-Specific Integrated Circuit (ASIC)** : est un circuit exclusivement dédié à une application bien précise et à un utilisateur spécifique. La

croissance de la technologie ASIC a été alimentée par les pressions concurrentielles et les exigences permanentes des clients pour fournir des niveaux plus élevés d'intégration dans les systèmes électroniques afin d'atteindre les objectifs de performance : coût et qualité. Géneralement, les concepteurs d'ASIC sont leurs utilisateurs [91]. La production d'ASIC est généralement réservée à la production de gros volumes à cause de son coût de développement très élevé.

- **FPGA** : est un circuit ré-configurable qui se caractérise par une grande flexibilité permettant de le réutiliser à volonté dans des algorithmes différents en un temps court. Également, le progrès continu de ces technologies permet de faire des composants toujours plus rapides et à plus haute intégration. Cette solution permet de réduire la taille de la carte, diminuer la consommation d'énergie, réduire la communication d'Entrées/Sorties.
- **Graphics Processing Unit (GPU)** : L'unité de traitement graphique a fait des progrès importants en tant qu'un accélérateur du calcul parallèle [92]. L'histoire de la naissance de ces processeurs parallèles combine deux domaines, la science computationnelle et l'industrie des applications vidéos intensives [93]. Les processeurs GPU sont massivement parallèles et capables de gérer jusqu'à des milliers des tâches en parallèle. Leur parallélisme massif les rend aussi intéressants comme processeurs de calcul matriciel. Enfin, leur rapport performance/prix les privilégie pour les applications qui demandent du calcul intensif [94].
- **Digital Signal Processor (DSP)** : est un processeur du traitement numérique du signal, plus utilisé dans le calcul des signaux numériques comme le filtrage, la compression vidéo...
- **Système sur puce hétérogène** : contrairement à un système homogène qui comprend un seul type de calculateur, un système hétérogène rassemble plusieurs calculateurs de différents types tels que CPU, DSP, FPGA et GPU. Ces calculateurs sont connectés entre eux à travers des bus spécifiques, et partagent un certain nombre de ressources telles que les mémoires. Une combinaison est souvent utilisée dans le contexte du système embarqué est celle de

CPU/FPGA. Xilinx a proposé le SOC $Zynq-7000$ *All Programmable SoC* qui inclut un FPGA et un processeur $ARM\ Cortex-A9\ MP$. Pareillement, Altera a proposé son SOC sur le marché qui est *Intel FPGAs* incluant un FPGA avec un processeur $ARM\ Cortex-A9\ MP$ [95]. Nous pouvons citer une autre combinaison, celle de CPU/GPU [96].

Par ailleurs, la conception d'un système numérique complexe nécessite souvent la mise en œuvre d'un processeur embarqué qui est une fonctionnalité très précieuse de SOC. Deux catégories des processeurs embarqués peuvent être distinguées. Ils peuvent être soient des blocs Intellectual Property (IP : description matérielle compilée permettant de réaliser un type particulier du traitement de données dans un FPGA), donc on parle des processeurs Softcores tels que le processeur Leon [97], soient implémentés en dur dans le circuit électronique et on parle dans ce cas des processeurs Hardcores tels que ARM Cortex-A9. Un softcore est une implémentation du processeur disponible sous forme de description haut niveau, dans un langage de description matérielle comme le VHDL ou le Verilog, ou sous la forme de Netlist. À noter que le processeur Softcore, qui sera implémenté dans un circuit programmable, offre une flexibilité pour être reconfiguré à chaque utilisation afin de s'adapter aux nouvelles contraintes. Mais il se caractérise par sa grande consommation des éléments programmables du circuit. Quant à un Hardcore (gravé en dur dans le silicium), qui est souvent standard et ne peut pas être modifié, il est vendu validé (sans bug), optimisé en taille et en vitesse. Par conséquent, il est moins flexible que le Softcore mais plus optimisé en terme de taille et puissance de traitement.

II.5.3 FPGA : Une solution pour accueillir la perception

Les circuits reconfigurables FPGAs sont désormais des plateformes cibles privilégiées pour la mise en œuvre des applications de traitement de signal intensif sur des systèmes autonomes pour plusieurs raisons, à savoir :

- Les FPGAs offrent du calcul à haute performance à des fréquences du fonctionnement plus basses [98].
- Il peut accueillir des architectures massivement parallèles en profitant de la quantité énorme de logiques programmables disponibles sur une seule puce [99].

- Un FPGA est un bon candidat pour déployer un système économe en énergie grâce à sa faible consommation d'énergie [100]. De plus, les FPGA se caractérisent par leur faible consommation d'énergie par rapport aux GPU et CPU [101].
- Le cycle de vie de nombreux circuits tels que le micro-contrôleur et le DSP varient entre 5 et 7 ans car leurs fournisseurs ont tendance à rendre les circuits, déjà matures, obsolètes, alors que le cycle de vie des circuits FPGA varie entre 15 et 20 ans [101].
- Un FPGA supporte la fonctionnalité de reconfiguration dynamique (DPR : Dynamic Partial Reconfiguration), où certaines parties matérielles peuvent être reconfigurées durant l'exécution tandis que les autres IPs reste inchangés. Il s'agit du changement de programmation de circuit logique programmable alors qu'il est en activité [102]. Cette fonctionnalité peut être exploitée dans les applications de traitement vidéo où le même système peut prendre en charge différentes configurations et à chaque changement de configuration, le FPGA charge le Bitsrteam correspondant [103].
- Outre que le rôle du calcul, un FPGA peut être utilisé comme une plateforme de communication avec l'existence des ports de communication standard sur la carte tels que le bus CAN, Ethernet, FPGA Mezzanine Card (FMC) [84]. De plus, sa structure flexible d'Entrée/Sortie permet de mettre en œuvre un système d'acquisition et de traitement de latence déterministe et rapide pour réceptionner des données de taille considérable provenant de différents capteurs [104].

Par conséquent, les attraits de cette nouvelle tendance technologique nous amène à considérer le composant FPGA comme une solution clef pour proposer des solutions innovantes dans le domaine de véhicule autonome.

II.5.4 Fonctionnalités de perception à base de solution FPGA

La solution FPGA offre une grande flexibilité dans la conception des systèmes embarqués, ce qui a permis d'attirer l'attention de la communauté scientifique pour mettre en œuvre des systèmes de robotiques embarqués. L'emploi d'une telle solution

a pour but d'augmenter la puissance du calcul face aux algorithmes intensifs et les exigences qui se modifient au cours de navigation du véhicule. Pour atteindre la performance requise, améliorer la réactivité et la prise de décision, qui sont des critères indispensables pour la sécurité routière, plusieurs approches sont proposées.

Dans [73], Salina et al. ont proposé une implémentation matérielle à base de FPGA pour la détection d'obstacles en utilisant les capteurs caméra infrarouge et ultrasonique. L'implémentation matérielle a été effectuée sur la carte *Papilio One* qui inclut un FPGA *Spartan* $3E$. La technique proposée a permis de rendre le système plus rapide, efficace et moins complexe.

Dans [96], Bauer et al. ont proposé une architecture matérielle pour la détection des piétons. Pour ce faire, trois calculateurs ont été couplés ensemble qui sont : un GPU, un FPGA et une CPU. Le but est de développer une approche de fusion de caractéristiques détectées par deux caméras, une monoculaire et une infrarouge. Chaque calculateur a un rôle précis, le FPGA extrait les caractéristiques, le GPU pour faire la classification et la CPU s'occupe de synchroniser et fusionner les données. L'implémentation de cette approche a permis de réduire les fausses alarmes tout en assurant un traitement en temps réel. Par contre, l'architecture matérielle utilisée nécessite la maîtrise de différents environnements de conception (flot de conception, langages de programmation...).

Dans le cadre d'une collaboration industrielle avec NAVYA, Ali et al. [101] ont implémenté une solution matérielle parallèle pour algorithme de stéréo matching « Multi-window Sum of Absolute Difference » sur une plateforme embarquée $Zynq-7000$. Cet algorithme permet la détection des obstacles autour d'une navette autonome. Ils ont présenté une méthodologie pour modifier le code de description de haut niveau afin de l'adapter efficacement à l'implémentation matérielle. Ils ont également exploré l'espace de conception pour différentes alternatives en termes de performance, ressources matérielles, fréquence et consommation d'énergie.

Au niveau de calibration des capteurs, une approche a été présentée dans [105] pour accélérer le processus de calibration géométrique de caméra en s'appuyant sur la solution FPGA. Cette implémentation a permis d'améliorer la performance d'exécution environ 1.3-1.5 fois par rapport à l'implémentation logicielle sur un PC.

Dans [106], Lyu et al. ont proposé une implémentation matérielle sur un FPGA d'un algorithme de segmentation de route en temps réel, basé sur un réseau de neurones et en utilisant un capteur LIDAR. Le temps d'exécution de l'approche proposée, environ 16,9 ms, a prouvé sa bonne performance par rapport aux travaux précédents. Dans [107], Schwiegelshohn et al. ont développé une approche pour la détection et la reconnaissance des panneaux de signalisation en utilisant le FPGA.

Dans ces travaux cités, le FPGAs a été une solution appropriée pour les différentes tâches de perception telles que la fusion, la segmentation, la calibration, la détection et la reconnaissance des obstacles. Cependant, l'augmentation du nombre de tâches accueillies par le FPGA à la fois, a ses limites notamment celle de sa capacité d'occupation matérielle. Une piste pour remédier à ce problème est l'emploi de la reconfiguration dynamique partielle DPR [84].

II.5.5 La synthèse de haut niveau (HLS)

La méthodologie de conception conventionnelle basée sur le codage manuel présente plusieurs contraintes liées au coût, au risque d'erreur lors de l'intégration des parties SW/HW, vérification, validation et temps du développement. En fait, avec la complexité des applications, les concepteurs ont besoin d'un lien direct entre la spécification et l'implémentation d'une application donnée pour satisfaire les contraintes des systèmes embarqués. Les concepteurs doivent relever de nombreux défis pour déployer leurs applications à savoir, le test et la vérification des algorithmes complexes dans les délais restreints de de Time-To-Market, la nécessité d'outils d'automatisation de développement pour augmenter la productivité de conception, réduction du coût de conception, réduction de puissance consommée... Pour palier à ces problèmes, une approche de conception ont été proposée afin de surmonter les limitations des méthodes de conception conventionnelles.

Cette approche, appelée HLS, a pour but d'accélérer la création des IPs en permettant d'implémenter directement une seule spécification SW/HW dans des circuits programmables sans qu'il soit nécessaire de coder manuellement les solutions au niveau Register Transfer Level (RTL).

Un outil HLS produit à partir d'un langage de programmation de haut niveau tel que C/C++, une spécification matérielle qui effectue la même fonction [108] (Figure II.9). Les outils HLS offrent aux développeurs la possibilité d'utiliser les architectures matérielles sans avoir une expertise matérielle.

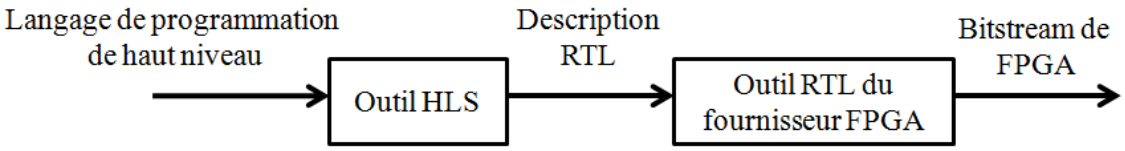

FIGURE II.9: Flot de conception des outils de synthèse de haut niveau (FPGA)

Les logiciels HLS ont été récemment appliqués à une variété d'applications (automobile, imagerie médicale, réseaux de neurones), avec des avantages significatifs en termes de performance, du coût et du temps nécessaire pour la mise en oeuvre d'une application donnée [109].

II.5.5.1 Classification des logiciels HLS

En se basant sur le langage de programmation en entrée, nous pouvons distinguer trois catégories pour les outils HLS académiques et commerciaux :

(i) **Conception basée sur le langage de programmation C/C++** [110] : Permet aux développeurs de produire les modules HW directement à partir du langage C/C++. Dans cette catégorie, nous citons à titre d'exemple les outils Vivado HLS d'Xilinx et Intel HLS Compiler d'Altera.

(ii) **Architecture pilotée par les modèles (Model Driven Architecture)** [111] : Cette catégorie a comme entrée des modèles décrits en langage UML tel que Gaspard.

(iii) **Conception basée sur un modèle (MBD : Model-Based Design)** [112] : La génération des codes VHDL ou C est fournie à partir des modèles qui sont représentés sous formes des blocs tels que Matlab/Simulink et Labview.

II.5.5.2 Outils HLS dans le contexte automobile

Dans [113], une implémentation matérielle, avec l'outil *Vivado HLS*, a été proposé pour la détection de voie afin que le véhicule puisse se localiser. L'implémentation sur la carte *Zynq* a permis de détecter la voie de circulation en temps réel pour un taux d'images de 130 images par seconde et une résolution de 480×270.

Sur la même plateforme cible et avec le même outil de synthèse, un filtre d'occupation bayésien est implémenté pour la localisation en temps réel d'un véhicule [114].

Les auteurs de [110] ont implémenté sur le *SoC Zynq* un algorithme de détection et suivi d'objets par un radar automobile. Cet algorithme traite simultanément plusieurs observations d'objets qui sont détectés par le radar pendant chaque balayage. Ce travail souligne l'utilisation de *Vivado HLS* pour obtenir une réduction de plus de 65% à la fois en termes de temps d'exécution et de consommation d'énergie par rapport à une implémentation sur une PC.

Le travail effectué dans [115] utilise aussi *Vivado HLS* pour implémenter l'algorithme Semi-Global Matching sur un FPGA *Virtex* 7. Cet algorithme est souvent utilisé dans les systèmes de vision stéréo dédiés aux applications automobiles. Le cycle de conception a été réduit tout en conservant une meilleure qualité de résultat et performance.

Dans ces travaux cités, les outils HLS ont été utilisés pour réaliser les différentes tâches liées à la perception d'un véhicule intelligent telles que la localisation et la détection d'obstacles. Ces outils permettent aux développeurs d'accélérer le temps du développement par rapport au codage manuel afin d'utiliser des calculateurs embarqués performants.

II.5.6 Synthèse

Dans cette section, nous avons présenté les motivations qui nous ont poussé à explorer les solutions embarquées dédiées à la perception pour les véhicules autonomes. Ces motivations, s'articulent autour de trois axes principaux, à savoir la performance temporelle, la consommation d'énergie électrique et la sûreté de fonctionnement. Après la présentation des technologies existantes des calculateurs embarqués, nous

avons retenus la solution de FPGA. Le choix de cette solution est justifié par différentes raisons telles que : (i) Les FPGAs offrent du calcul à haute performance à des fréquences du fonctionnement plus basses. (ii) Les FPGA se caractérisent par leur faible consommation d'énergie par rapport aux GPU et CPU, qui représente un facteur important dans le cas du véhicule autonome 100% électrique. D'autre côté, nous nous intéressons aux outils HLS pour synthétiser des implémentations matérielles. Ces outils ont prouvé leur efficacité d'accélérer le cycle du développement de création des IPs, avec des avantages significatifs en termes de performance, du coût et du temps nécessaire pour la mise en œuvre d'une application donnée.

II.6 Positionnement

Comme nous avons décrit précédemment, le positionnement de nos travaux de recherche inclut la calibration des capteurs, la fusion multi-capteurs et l'exploration des implémentations embarquées dédiées à la perception de véhicule autonome, et en particulier la fonctionnalité *DTMO*.

Tout d'abord, nous proposons une méthodologie de calibration entre les capteurs caméra et LIDAR en utilisant une mire rectangulaire. Cette méthode devra répondre aux contraintes de précision et de rapidité de mise en œuvre. De plus, il sera nécessaire de développer une chaîne de traitement permettant d'extraire les caractéristiques de la mire de calibration par les deux capteurs afin de réduire l'intervention de l'utilisateur dans la mesure du possible.

Ensuite, nous proposons une approche de fusion entre les capteurs caméra et radar en s'appuyant sur la théorie des fonctions de croyance. Cette théorie est bien adaptée au problème de fusion de données et modélise bien la performance des différents capteurs [21]. L'emploi d'une architecture multi-capteurs est important pour tirer profit des avantages de chaque capteur. L'approche proposée a pour objectif d'améliorer la précision de détection et la reconnaissance des obstacles.

Dans un troisième temps, afin d'assurer une conduite autonome pour un véhicule 100% électrique, la tâche du calcul doit fournir simultanément des performances élevées, une faible consommation d'énergie et une faible dissipation thermique, à faible

coût. Donc, dans la troisième partie, nous allons présenter une conception d'une plateforme embarquée à base de la carte $Zynq-700$ pour fusionner les positions des obstacles détectés entre une caméra stéréoscopique et un LIDAR. D'autre coté nous allons souligner l'importance des outils HLS dans les applications automobiles intelligentes pour produire des accélérateurs matérielles.

II.7 Conclusion

Dans ce chapitre, nous avons effectué un tour d'horizon de l'état de l'art actuel. Nous avons abordé trois axes principaux de notre recherche bibliographique afin d'évaluer et comparer les solutions existantes ainsi que les verrous scientifiques et technologiques. En s'appuyant les résultats de nos recherches, nous visons à concevoir un système embarqué de fusion multi-capteurs pour la détection des obstacles autour d'un véhicule autonome.

Afin de fusionner les données, la première étape consiste à calibrer les capteurs. Dans le chapitre suivant, nous présentons notre méthodologie pour réaliser ce processus.

Chapitre III

Calibration extrinsèque caméra/LIDAR

Sommaire

III.1 Introduction

Ce chapitre présente notre approche proposée pour effectuer le processus de calibration extrinsèque caméra/LIDAR. Les travaux entrepris inclut également une chaîne de traitement pour extraire les caractéristiques de la mire de calibration par les deux capteurs : détection des lignes par la caméra et le processus de segmentation effectué par le LIDAR. Pour évaluer les résultats obtenus de notre contribution, nous les comparons avec l'approche originale [37] d'un coté et avec l'approche itérative [35] d'autre coté.

Ce chapitre est organisé comme suit : Dans la section III.2 nous présentons les modèles et les caractéristiques des capteurs utilisés ainsi que les processus de calibration de caméra monoculaire, de caméra stéréo et de caméra/radar. Dans la section III.3 nous détaillons notre contribution pour la calibration caméra/LIDAR en analysant la répercussion de la précision de ce processus sur la fusion de données. Les résultats ainsi que les comparaisons nécessaires par rapport aux approches de calibration originale et itérative, sont présentés dans la section IV.5.

III.2 Modèles et calibrations des capteurs

Dans cette section, nous commençons par introduire les modèles des capteurs utilisés, durant cette thèse (caméra, caméra stéréoscopique, LIDAR et le radar). Ensuite, nous présentons dans la deuxième partie de cette section les techniques utilisées pour calibrer les capteurs : caméra, caméra stéréo et caméra/radar.

III.2.1 Modèles et calibrations des capteurs

III.2.1.1 Caméra

Afin de pouvoir effectuer des calculs numériques ou des raisonnements géométriques à partir d'images, il est nécessaire d'avoir un modèle qui décrit comment le monde $3D$ se projette sur une image $2D$, qui représente la projection réalisée par une caméra. Il existe beaucoup de modèles, qui décrivent plus ou moins bien les ca-

ractéristiques d'une caméra (optique, électronique, mécanique). Le modèle typiquement utilisé en vision par ordinateur, présente un bon compromis entre la simplicité des équations associées et la proximité aux phénomènes physiques modélisés, est le modèle sténopé (Pinhole) qui représente en effet une projection perspective [116] (la transformation géométrique réalisée par un système optique qui forme l'image d'objets situés dans l'espace). Ce modèle représente une caméra par un plan image correspondant à la surface photosensible, un centre optique et un point principal.

La figure III.1 décrit un modèle sténopé typique [5]. Supposons que O est le centre de caméra qui représente l'origine du système de coordonnées de caméra et O_1 est le centre du système de coordonnées de l'image, appelé le point principal O_1. La distance entre le centre de caméra et le plan image est la distance focale f. Notons que la ligne qui passe par O et O_1, aussi perpendiculaire au plan de l'image est appelée axe principal. Dans le modèle sténopé, la projection du point p_2, situé au plan caméra, dans le plan image est le point p_1.

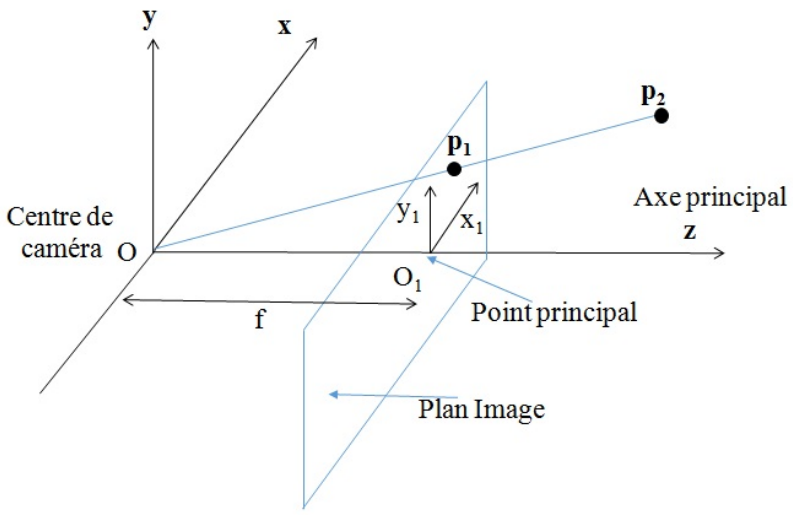

FIGURE III.1: Repères géométriques de caméra [5]

Le modèle sténopé transforme un point $3D$ de la scène observée dans le plan image en utilisant trois transformations. Nous décrivons dans la suite les transformations nécessaire pour un modèle sténopé considéré idéal, c'est-à-dire sans tenir compte des distorsions de l'objectif.

-Transformation entre le repère monde et celui de caméra : Cette transformation rigide est composée par une rotation ${}^{m}R_{c}$ et une translation ${}^{m}\overrightarrow{t}_{c}$. Les paramètres de cette transformation sont appelés paramètres extrinsèques de la caméra :

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = {}^{m}R_{c} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} + {}^{m}\overrightarrow{t}_{c} \tag{III.1}$$

avec $\begin{bmatrix} x & y & z & 1 \end{bmatrix}^{T}$ est les coordonnées homogènes d'un point $3D$ donné dans l'espace (repère monde), exprimé dans le repère caméra par $\begin{bmatrix} x_c & y_c & z_c & 1 \end{bmatrix}^{T}$.

-Transformation entre le repère caméra et celui du capteur (plan rétinien) : C'est une projection perspective entre le repère caméra et celui du capteur qui convertit un point $3D$ $[x_c, y_c, x_c]$ en un point image $[x, y]$:

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{III.2}$$

avec s est le facteur d'échelle et (f_x, f_y) est la distance focale f.

-Transformation entre le repère capteur et celui d'image : C'est la transformation qui permet de transformer les coordonnées d'un point image (x, y) (unité métrique) en coordonnées discrètes (u, v) (pixels) :

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_x & 0 & c_x \\ 0 & k_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{III.3}$$

avec (c_x, c_y) en pixels représente le centre de caméra et (k_x, k_y) représente le nombre

de pixels par unité de longueur le long des directions x et y du capteur.

III.2.1.2 Caméra stéréoscopique

D'après la partie précédente, chaque point de l'espace tridimensionnel est associé à un point de l'espace bidimensionnel de l'image. La troisième dimension est alors perdue et aucune information de profondeur n'est donc disponible. Donc, une infinité de points $3D$ est associée à chaque point de l'image, tout le long de la droite tridimensionnelle passant par le centre optique et le point image. Par triangulation et grâce à l'utilisation d'une deuxième caméra, il est alors possible de reconstruire cette information de profondeur en déterminant le point d'intersection des deux droites tridimensionnelles. Par conséquent, il est nécessaire d'exprimer la géométrie des deux capteurs dans un même référentiel [6].

La figure III.3 décrit les trois référentiels ainsi que les trois transformations rigides (T, T_1 et T_2) permettant le changement de repère d'un référentiel à un autre. Notons que R_m est le référentiel de la scène et R_{c1}, R_{c2} sont respectivement les référentiels de la caméra gauche et celle de droite.

La figure III.4 montre comment la profondeur des objets est déterminée dans le problème de correspondance stéréo. Supposons deux caméras de distance focale f au même niveau horizontal, séparées l'une de l'autre par une distance de ligne de base (*baseline*) b (le segment reliant les deux centres optiques). Le pixel p situé dans l'espace sera situé au point x_R dans l'image droite et le point x_L à celle de gauche. Donc, le principe est de trouver pour chaque pixel x_R dans l'image droite, le meilleur pixel correspondant x_L dans l'image gauche qui doit être situé sur la même ligne d'image. La différence entre les deux points sur le plan image est définie comme la disparité d représenté dans la figure III.3.(b). Par conséquent, la profondeur de pixel p est calculée avec l'équation III.4.

$$Profondeur = \frac{b * f}{d} \tag{III.4}$$

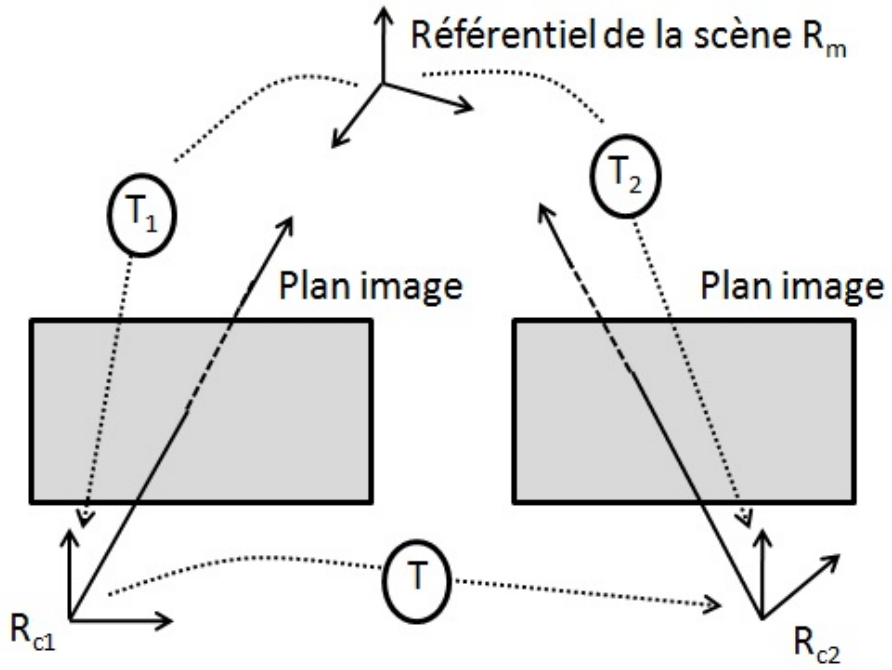

FIGURE III.2: Les trois référentiels tridimensionnels du capteur de stéréo-vision [6]

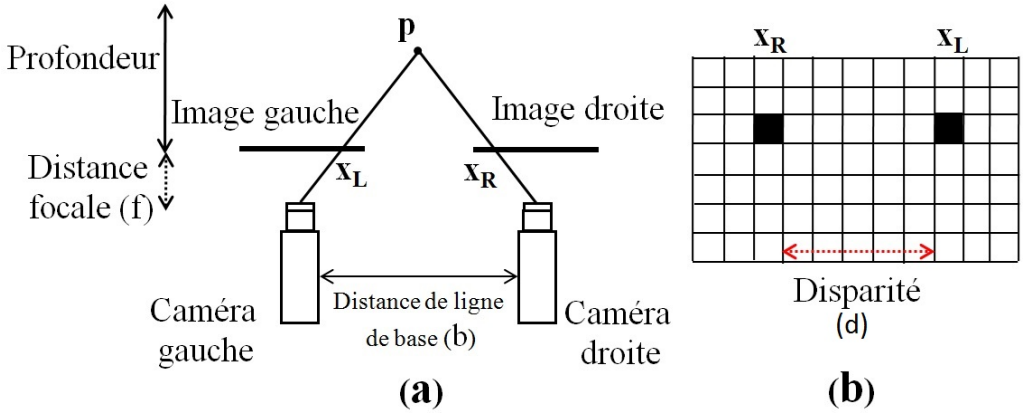

FIGURE III.3: Calcul de la profondeur d'un objet dans un problème de correspondance stéréo

III.2.1.3 LIDAR

Nous utilisons dans ce travail un capteur LIDAR mono-nappe. Ce capteur émet en permanence des faisceaux de lumière laser, puis mesure le temps nécessaire au retour de la lumière dans le capteur. La figure III.4 décrit le principe du fonctionnement de LIDAR pour un seul faisceau de lumière (domaine infrarouge). La mesure

de la distance se déduit à partir du retard entre l'émission de l'impulsion électromagnétique et sa réception à partir de la formule suivante [117] :

$$d = \frac{c\ \Delta_t}{2} \tag{III.5}$$

avec $c = 3.10^8\ m/s$ la vitesse de la lumière et Δ_t le temps écoulé entre l'émission et la réception de l'impulsion électromagnétique. Notons que le nombre de faisceaux dépend de la technologie de LIDAR utilisée.

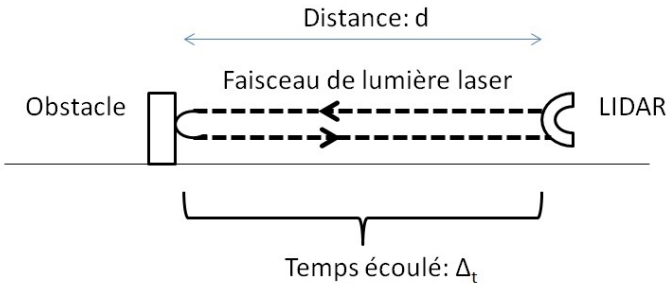

FIGURE III.4: Principe du fonctionnement de LIDAR pour un seul faisceau de lumière

Le LIDAR fournit cycliquement une série de mesures des distance brutes (r_i, θ_i) dans un système de coordonnées polaires avec r_i la distance et θ_i l'angle. Ces mesures peuvent être converties en cordonnées cartésiennes $[x_i, y_i]^T$, où :

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} r_i\, cos(\theta_i) \\ r_i\, sin(\theta_i) \end{bmatrix} \tag{III.6}$$

Dans le système de coordonnées du capteur LIDAR, nous spécifions le point qui émet les rayons laser comme l'origine du repère.
Ce capteur présente des erreurs des mesures qui sont caractérisées par les constructeurs dans les documents techniques (écart-type ou variance d'erreur). Par ailleurs, la transmission de données, du capteur LIDAR utilisé dans cette thèse, est effectuée avec la liaison Ethernet et suivant le protocole TCP/IP.

III.2.1.4 Radar

Le radar fournit des mesures des positions des obstacles détectés ainsi que leurs vitesses relatives. Ce capteur comporte principalement un émetteur, une antenne et un récepteur, et utilise le rayonnement radar pour analyser son environnement. À la différence du LIDAR qui emploie la lumière (région infrarouge du spectre électromagnétique), le radar utilise les ondes radio-fréquence. Il utilise la propriété des ondes électromagnétiques de se réfléchir sur tout obstacle, créant ainsi une onde de retour susceptible d'être décelée par un récepteur adapté à ce signal. Ce principe peut être mis en évidence lorsque le signal émis est une suite d'impulsions électromagnétiques. Chaque impulsion de durée très brève de l'ordre de quelque microsecondes se propage dans atmosphère à la vitesse de la lumière [118]. Pour calculer la distance, le radar utilise le même principe de LIDAR. En outre, le radar peut fournir la vitesse d'une cible due à l'effet Doppler [118].

Par ailleurs, nous utilisons le radar 24 GHz à courte portée qui est configuré via le protocole CAN. Il peut être configuré suivant deux modes :

(i) **Cluster** : se compose des multiples réflexions ou échos qui ont une position et un mouvement similaires et peuvent être donc combinées.

(ii) **Piste** : se compose des clusters qui sont suivis dans le temps. Plus précisément, il s'agit des obstacles détectés et qui sont suivis par le radar. La position de chque objet est calculée dans le système de coordonnées cartésiennes.

Pour résumer, les Clusters correspondent aux objets non suivis et les pistes sont dérivées à partir des algorithmes du filtrage et du suivi qui conservent également les historiques de leurs déplacements. Dans cette thèse, nous configurons le radar pour utiliser le mode piste où le capteur envoie des listes d'objets détectés. Le nombre maximal de pistes est 25 pistes (objet détecté) qui sont mises à jour chaque 33 millisecondes. Chaque Piste a un identificateur unique, une position, une vitesse relative et une valeur RCS .

III.2.2 Calibration des capteurs : caméra, caméra stéréo et caméra/radar

Nous avons expliqué précédemment les modèles et les caractéristiques techniques des capteurs utilisés. Dans cette section, nous présentons les différentes calibrations nécessaires des capteurs caméra, caméra stéréo et caméra/radar, et qui seront utilisées soient par d'autres processus de calibration, tels que la calibration de caméra/LIDAR qui nécessite la calibration de caméra, soient par la fusion multi-capteurs caméra/radar qui nécessite la calibration des capteurs caméra et radar.

III.2.2.1 Calibration de camera monoculaire

En vision par ordinateur, dès lors que l'on souhaite obtenir des informations métriques, il est nécessaire de calibrer la caméra. Le calibrage d'une seule caméra (application monoculaire) revient à estimer ses paramètres intrinsèques ainsi que sa position par rapport au référentiel du monde. L'approche classique consiste à estimer les paramètres de la caméra à partir de la géométrie d'une mire connue avec exactitude. Les paramètres à estimer sont alors les paramètres intrinsèques et extrinsèques (paramètres pour l'orientation et position). Considérons quelques techniques de calibration existantes basées sur une mire de calibrage. Certaines nécessitent un damier, comme la bibliothèque *OpenCV* [119] ou le $ToolboxMatlab$ [120].

L'utilisation d'un damier est assez pratique car sa détection est rapide par les bibliothèques de vision par ordinateur telle que *OpenCV*. Dans ce travail, nous utilisons un damier pour faire la calibration de caméra. L'entrée de ce processus sera des images VGA 640×480 de différentes poses avec différentes orientations de damier et la sortie sera les paramètres intrinsèques et extrinsèques de caméra.

III.2.2.2 Calibration de camera stéréoscopique

Configurer un capteur stéréoscopique revient à calibrer les deux caméras (paramètres intrinsèques de chacune des caméras) et déterminer la position et l'orientation relative des deux caméras [6]. La calibration du système de vision stéréo est

effectuée en deux étapes. La première étape consiste à calibrer chaque caméra indépendamment comme déjà décrit dans la partie précédente. La deuxième étape est l'étalonnage extrinsèque entre les deux caméras qui consiste à estimer les paramètres extrinsèques entre eux (matrice de rotation et vecteur de translation).

III.2.2.3 Calibration de caméra/radar

Pour effectuer la calibration de caméra/radar, nous avons utilisé la méthode décrite dans [121]. L'idée est d'utiliser une mire pour déterminer sa position dans les deux repères capteurs, et ensuite exprimer les coordonnées de cette mire dans chaque repère afin de trouver la transformation rigide entre le repère de caméra et celui de radar. Cette méthode nécessite l'utilisation d'une mire en métal (réflecteur) qui a une forme d'un trièdre carré. Ce réflecteur assure une continuité de détection de la mire pour un point donné dans le système de coordonnées radar, autrement dit, il permet d'avoir une bonne réflexion des ondes émises. Le RCS du réflecteur est calculé par l'équation III.7 :

$$RCS \;\; de \;\; la \;\; mire = \frac{4.\pi.L^4}{\lambda^2} \tag{III.7}$$

avec L est la longueur du réflecteur et λ est la fréquence de radar.
Ensuite, nous avons besoin de la position du réflecteur dans le système de coordonnées radar et sa positon dans celui de caméra. En utilisant les données collectées, nous résolvons le problème de *Perspective-n-Point* (PnP) [122] pour déterminer la transformation rigide.

III.3 Calibration extrinsèque caméra/LIDAR

Dans la section précédente, nous avons décrit les différentes calibrations requises pour développer les travaux de cette thèse. Dans cette section, nous présentons notre première contribution qui concerne la calibration des capteurs caméra et LIDAR. Le but de cette calibration est de déterminer les paramètres de la transformation rigide entre les deux repères capteurs. Pour ce faire, en se basant sur [37], nous

ré-développons la solution de calibration existante afin de l'utiliser pour une mire contentant plusieurs lignes. L'emploi de cette mire permet d'effectuer la calibration entre ces capteurs tout en réduisant le nombre de poses requis, ce qui permet de réduire le temps nécessaire pour la mise en œuvre de cette méthode. De plus, nous promouvons cette approche par le développement d'une chaine d'extraction des caractéristiques pour les deux capteurs afin d'automatiser ce processus dans la mesure du possible. Dans la suite, nous détaillons les différentes étapes du développement de notre contribution.

III.3.1 Formulation du problème et concepts de base

Notre but est de déterminer la transformation rigide $[{}^{C}R_{L}|{}^{C}\overrightarrow{t}_{L}]$, qui permet de trouver le correspondant d'un point LIDAR $\overrightarrow{p}_{L} = [x_L, y_L, z_L]^T$ dans le repère caméra $\{C\}$. Nous considérons que $\overrightarrow{p}_{C} = [x_C, y_C, z_C]^T$ le point associé à $\overrightarrow{p}_{L}$ dans le repère caméra :

$$\overrightarrow{p}_{C} = {}^{C}R_{L}\overrightarrow{p}_{L} + {}^{C}\overrightarrow{t}_{L} \tag{III.8}$$

La figure III.5 présente la mire de calibration utilisée ainsi que les contraintes géométriques entre le repères caméra et LIDAR. Tout d'abord, nous définissons le système de coordonnées pour les capteurs comme suit : l'origine O_C est le centre du repère caméra et l'origine O_L est le centre du repère LIDAR. Le plan de champ de vision de LIDAR est défini par le plan à $z_L = 0$. Donc, un point LIDAR est représenté par $\overrightarrow{p}_{L} = [x_L, y_L, 0]^T$.

Nous considérons que la mire de calibration contient d lignes horizontales noires lb_{ih} avec h est le nombre de lignes et i est le numéro de la pose. Nous fixons des lignes équidistantes entre eux, c'est à dire s'il y a d lignes, il est nécessaire que les $(d+1)$ parties soient égales.

Supposons que $\overrightarrow{p}_{li}$ et $\overrightarrow{p}_{ri}$ sont respectivement les points LIDAR des bords gauche et droite projetés sur la mire de calibration. Pour estimer les positions du point $\overrightarrow{p}_{mih}$, suivant les différentes poses, nous divisons la distance entre $\overrightarrow{p}_{li}$ et $\overrightarrow{p}_{ri}$ par $(d+1)$ (Figure III.5).

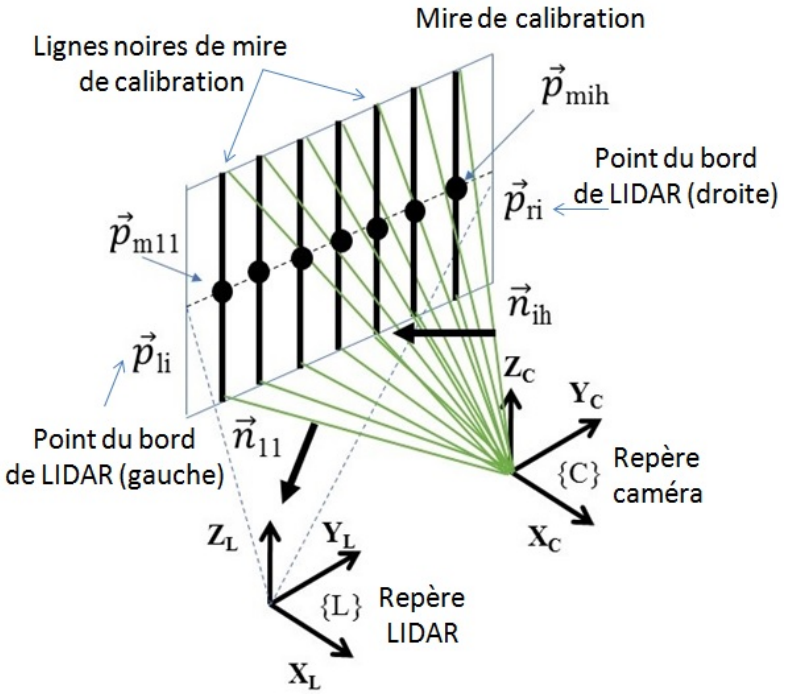

FIGURE III.5: Contraintes géométriques pour la calibration Caméra/LIDAR

Soit $\overrightarrow{p}_{cih}$ l'image du point LIDAR $\overrightarrow{p}_{mih}$ dans le repère caméra. Par ailleurs, le vecteur normal $\overrightarrow{n}_{ih}$ est perpendiculaire au plan T_h défini par l_{bih} et le centre de caméra. Puisque le point $\overrightarrow{p}_{mih}$ (situé sur la ligne l_{bih}) appartient au plan T_h, donc son image $\overrightarrow{p}_{cih}$ dans le repère caméra appartient aussi au T_h, et par la suite :

$$\overrightarrow{n}_{ih}^{T}\,\overrightarrow{p}_{cih} = \overrightarrow{n}_{ih}^{T}({}^{C}R_L\,\overrightarrow{p}_{mih} + {}^{C}\overrightarrow{t}_L) = 0 \tag{III.9}$$

Pratiquement, l'équation III.9 n'est pas satisfaite à cause des bruits des mesures et elle sera différente de zéro. Cette écart est le résidu e_i. Pour n poses, les résidus sont représentés par :

$$\begin{aligned}\overrightarrow{e} = (&\overrightarrow{n}_{11}^{T}({}^{C}R_L\,\overrightarrow{p}_{m11} + {}^{C}\overrightarrow{t}_L), \overrightarrow{n}_{12}^{T}({}^{C}R_L\,\overrightarrow{p}_{m12} + {}^{C}\overrightarrow{t}_L)\\ &, ..., \overrightarrow{n}_{nd}^{T}({}^{C}R_L\,\overrightarrow{p}_{mnd} + {}^{C}\overrightarrow{t}_L))\end{aligned} \tag{III.10}$$

Afin d'estimer les paramètres de transformation $[{}^{C}R_L|{}^{C}\overrightarrow{t}_L]$ avec une précision et minimiser les résidus illustrés par (III.10), nous définissons le problème qui minimise la somme quadratique des erreurs comme suit :

$$J = \underset{{}^{C}t_L, {}^{C}R_L}{argmin} \sum_{i=1}^{n} \sum_{h=1}^{d} (e_{ih})^2 \tag{III.11}$$

$$\begin{aligned} J = \underset{{}^{C}t_L, {}^{C}R_L}{argmin} & \sum_{i=1}^{n} \sum_{h=1}^{d} (\overrightarrow{n}_{ih}^{T} ({}^{C}R_L \, \overrightarrow{p}_{mih} + {}^{C}\overrightarrow{t}_L))^2 \\ & s.t. \, {}^{C}R_L^{T} \, {}^{C}R_L = I, \;\; \det({}^{C}R_L) = 1 \end{aligned} \tag{III.12}$$

Notons que le problème est soumis aux contraintes qui expriment les propriétés de matrice de rotation. Rappelons qu'une matrice de rotation doit satisfaire les propriétés suivantes : ***(i)*** Le déterminant d'une matrice de rotation est égale à 1. ***(ii)*** La multiplication de la matrice transposée par la matrice elle même, est égale à la matrice identité.
L'objectif est de résoudre le problème de minimisation pour déterminer ${}^{C}R_L$ et ${}^{C}\overrightarrow{t}_L$, les paramètres de calibration.

III.3.1.1 Réduction du nombre de variables

La première étape consiste à réduire le nombre de variables pour simplifier l'équation (III.12). Puisque la translation ${}^{C}\overrightarrow{t}_L$ n'est pas impliquée dans les contraintes, elle peut être éliminée du problème d'optimisation. En appliquant la condition nécessaire d'optimalité du premier ordre [45] à la fonction de coût (Eq III.12), le nombre de variables sera réduit.

Lemme 1 *:* Nous pouvons réduire le nombre de variables dans Eq III.12 par l'application de la condition nécessaire d'optimalité du premier ordre :

$$\begin{aligned} J = & \sum_{i=1}^{n} \sum_{h=1}^{d} \Big[\overrightarrow{n}_{ih}^{T} \, {}^{C}R_L \, \overrightarrow{p}_{mih} - (\sum_{j=1}^{n} \sum_{l=1}^{d} \overrightarrow{w}_{ihjl}^{T} \, {}^{C}R_L \, \overrightarrow{p}_{mjl}) \Big]^2 \\ & s.t. \, {}^{L}R_C^{T} \, {}^{C}R_L = I, \;\; \det({}^{C}R_L) = 1 \end{aligned} \tag{III.13}$$

Avec :

$$\overrightarrow{w}_{ihjl} = \overrightarrow{n}_{ih}^{T} \Big(\sum_{j=1}^{n} \sum_{l=1}^{d} \overrightarrow{n}_{jl} \overrightarrow{n}_{jl}^{T} \Big)^{-1} \overrightarrow{n}_{jl} \overrightarrow{n}_{jl}^{T} \tag{III.14}$$

$${}^{C}\overrightarrow{t}_L = -\Big(\sum_{i=1}^{n} \sum_{h=1}^{d} \overrightarrow{n}_{ih} \overrightarrow{n}_{ih}^{T} \Big)^{-1} \Big(\sum_{i=1}^{n} \sum_{h=1}^{d} \overrightarrow{n}_{ih} \overrightarrow{n}_{ih}^{T} \, {}^{C}R_L \, \overrightarrow{p}_{mih} \Big) \tag{III.15}$$

Preuve : En appliquant la condition nécessaire d'optimalité du premier ordre [45] à la fonction de coût (Eq III.12), nous obtenons :

$$\begin{aligned}\frac{\partial J}{\partial {}^C\overrightarrow{t}_L} &= \frac{\partial}{\partial {}^C\overrightarrow{t}_L}\left(\sum_{i=1}^{n}\sum_{h=1}^{d}(\overrightarrow{n}_{ih}^T({}^C R_L\,\overrightarrow{p}_{mih} + {}^C\overrightarrow{t}_L))^2\right) \\ &= \sum_{i=1}^{n}\sum_{h=1}^{d} 2\overrightarrow{n}_{ih}\left[\overrightarrow{n}_{ih}^T\,{}^C R_L\,\overrightarrow{p}_{mih} + \overrightarrow{n}_{ih}^T\,{}^C\overrightarrow{t}_L\right] = 0\end{aligned} \tag{III.16}$$

Supposons que :

$$F = \left(\sum_{i=1}^{n}\sum_{h=1}^{d}\overrightarrow{n}_{ih}\,\overrightarrow{n}_{ih}^T\right)$$

Donc, d'après l'équation III.16, le vecteur de translation ${}^C\overrightarrow{t}_L$ est exprimé comme suit :

$${}^C\overrightarrow{t}_L = -F^{-1}\left(\sum_{i=1}^{n}\sum_{h=1}^{d}\overrightarrow{n}_{ih}\,\overrightarrow{n}_{ih}^T\,{}^C R_L\,\overrightarrow{p}_{mih}\right) \tag{III.17}$$

En remplaçant ${}^C\overrightarrow{t}_L$ par son expression (Eq III.17) dans la fonction de coût (Eq III.12), nous obtenons :

$$\begin{aligned}J &= \sum_{i=1}^{n}\sum_{h=1}^{d}\left[\overrightarrow{n}_{ih}^T\,{}^C R_L\,\overrightarrow{p}_{mih} - \overrightarrow{n}_{ih}^T F^{-1}(\sum_{j=1}^{n}\sum_{l=1}^{d}\overrightarrow{n}_{jl}\,\overrightarrow{n}_{jl}^T\,{}^C R_L\,\overrightarrow{p}_{mjl})\right]^2 \\ &= \sum_{i=1}^{n}\sum_{h=1}^{d}\left[\overrightarrow{n}_{ih}^T\,{}^C R_L\,\overrightarrow{p}_{mih} - (\sum_{j=1}^{n}\sum_{l=1}^{d}\overrightarrow{n}_{ih}^T F^{-1}\overrightarrow{n}_{jl}\,\overrightarrow{n}_{jl}^T\,{}^C R_L\,\overrightarrow{p}_{mjl})\right]^2\end{aligned}$$

Supposons que :

$$\overrightarrow{w}_{ihjl} = \overrightarrow{n}_{ih}^T\left(F\right)^{-1}\overrightarrow{n}_{jl}\,\overrightarrow{n}_{jl}^T \tag{III.18}$$

Alors, l'équation (III.12) est exprimée seulement en fonction de matrice après l'élimination de translation :

$$J = \sum_{i=1}^{n}\sum_{h=1}^{d}\left[\overrightarrow{n}_{ih}^T\,{}^C R_L\,\overrightarrow{p}_{mih} - (\sum_{j=1}^{n}\sum_{l=1}^{d}\overrightarrow{w}_{ihjl}^T\,{}^C R_L\,\overrightarrow{p}_{mjl})\right]^2 \tag{III.19}$$

III.3.1.2 Formulation du problème d'optimisation à l'aide de quaternion unitaire

Une autre représentation possible des rotations, à part la matricielle, est celle utilisant les quaternions. Nous introduisons tout d'abord les quaternions, ensuite nous

montrons comment ils nous permettent de simplifier notre problème d'optimisation.

Les quaternions peuvent être vus comme des vecteurs de quatre dimension ou encore comme des nombres hyper-complexes à trois parties imaginaires [123] :

$$\bar{q} = q_4 + q_1 i + q_2 j + q_3 k \tag{III.20}$$

où i, j et k sont des nombres complexes et q_4 est la partie réelle ou scalaire du quaternion. Nous pouvons représenter une rotation avec un quaternion unitaire. Un quaternion est unitaire si et seulement si sa norme est égale à 1 [124]. La norme de q est calculée comme suit [124] :

$$\| q \|^2 = q_4^2 + q_1^2 + q_2^2 + q_3^2 \tag{III.21}$$

La conversion des matrices et vecteurs en quaternions ainsi que toutes les opérations du calcul sont présentés dans l'annexe B.

Afin de simplifier le problème et réduire davantage le nombre d'inconnues, nous remplaçons la matrice de rotation ${}^{C}R_{L}$ (3×3) par le quaternion unitaire q (4×1).

Lemme 2 *:* Nous réduisons le nombre d'inconnues dans la fonction de coût par l'utilisation de quaternion unitaire, donc l'équation (III.13) est exprimée comme suit :

$$\begin{aligned} J = \sum_{i=1}^{n} \sum_{h=1}^{d} \left[q^T \, S_{ih} \, q \right]^2 \\ s.t. {}^{L}q_{C}^{T} \, {}^{L}q_{C} = 1 \end{aligned} \tag{III.22}$$

avec :

$$S_{ih} = \mathcal{L}(\bar{n}_{ih})^T \mathcal{R}(\bar{p}_{mih}) - \sum_{j=1}^{n} \sum_{l=1}^{d} \mathcal{L}(\bar{w}_{ihjl})^T \mathcal{R}(\bar{p}_{mjl}) \tag{III.23}$$

avec $\mathcal{L}(.)$ et $\mathcal{R}(.)$ sont les matrices de multiplication gauche et droite de quaternion (voir annexe B).

Preuve : En se basant sur les propriétés du calcul de quaternion, la conversion des vecteurs et matrices en quaternions ainsi que la multiplication de quaternion,

nous pouvons redéfinir la fonction de coût comme suite :

$$
\begin{aligned}
J &= \sum_{i=1}^{n}\sum_{h=1}^{d}\left[\bar{n}_{ih}^{T}(q\otimes\bar{p}_{mih}\otimes q^{-1}) - \sum_{j=1}^{n}\sum_{l=1}^{d}\bar{w}_{ihjl}^{T}(q\otimes\bar{p}_{mjl}\otimes q^{-1})\right]^2 \\
&= \sum_{i=1}^{n}\sum_{h=1}^{d}\left[\bar{n}_{ih}^{T}(\mathcal{L}(q)\bar{p}_{mih}\otimes q^{-1}) - \sum_{j=1}^{n}\sum_{l=1}^{d}\bar{w}_{ihjl}^{T}(\mathcal{L}(q)\bar{p}_{mjl}\otimes q^{-1})\right]^2 \\
&\text{Puisque} \quad (\mathcal{L}(q)\bar{p}_{mih})\otimes q^{-1} = \mathcal{R}(q^{-1})(\mathcal{L}(q)\bar{p}_{mih}) \\
&\text{et} \quad (\mathcal{L}(q)\bar{p}_{mjl})\otimes q^{-1} = \mathcal{R}(q^{-1})(\mathcal{L}(q)\bar{p}_{mjl}) \\
&\text{Donc :} \\
J &= \sum_{i=1}^{n}\sum_{h=1}^{d}\left[\bar{n}_{ih}^{T}\mathcal{R}(q^{-1})\mathcal{L}(q)\bar{p}_{mih} - \sum_{j=1}^{n}\sum_{l=1}^{d}\bar{w}_{ihjl}^{T}\mathcal{R}(q^{-1})\mathcal{L}(q)\bar{p}_{mjl}\right]^2 \\
&\text{Puisque} \quad \mathcal{L}(q)\,\bar{p}_{mih} = \mathcal{R}(\bar{p}_{mih})q \\
&\text{et} \quad \mathcal{L}(q)\,\bar{p}_{mjl} = \mathcal{R}(\bar{p}_{mjl})q
\end{aligned}
\tag{III.24}
$$

Donc, la fonction de coût est reformulée par la fonction suivante :

$$
\begin{aligned}
J &= \sum_{i=1}^{n}\sum_{h=1}^{d}\left[q^{T}\mathcal{L}(\bar{n}_{ih})^{T}\mathcal{R}(\bar{p}_{mih})\,q - \sum_{j=1}^{n}\sum_{l=1}^{d}q^{T}\mathcal{L}(\bar{w}_{ihjl})^{T}\mathcal{R}(\bar{p}_{mjl})\,q\right]^2 \\
&= \sum_{i=1}^{n}\sum_{h=1}^{d}\left[q^{T}\left(\mathcal{L}(\bar{n}_{ih})^{T}\mathcal{R}(\bar{p}_{mih}) - \sum_{j=1}^{n}\sum_{l=1}^{d}\mathcal{L}(\bar{w}_{ihjl})^{T}\mathcal{R}(\bar{p}_{mjl})\right)q\right]^2
\end{aligned}
\tag{III.25}
$$

III.3.1.3 Méthode du multiplicateur de Lagrange

Pour résoudre la fonction de coût J, nous utilisons la méthode de multiplicateur de Lagrange [45]. Cette méthode s'applique à toute recherche de minimum en présence de contraintes qui est notre cas. La fonction lagrangienne est définie par :

$$
L(q,\lambda) = J(q) + \lambda(q^{T}q - 1) \tag{III.26}
$$

avec λ est le multiplicateur de Lagrange.
Donc, en utilisant la méthode de Lagrange, nous obtenons le système des equations suivant :

$$
\begin{cases}
\sum_{i=1}^{n}\sum_{h=1}^{d}\left[q^{T}\,S_{ih}\,q\right]\left[S_{ih}+S_{ih}^{T}\right]q + \lambda q = 0 \\
q^{T}q - 1 = 0
\end{cases}
\tag{III.27}
$$

L'équation (III.27) représente un système d'équation de quatre polynômes cubiques et un polynôme quadratique en cinq variables : $\{q_1, q_2, q_3, q_4, \lambda\}$. En outre,

il s'agit d'un système avec des coefficients en virgule flottante, car ces coefficients sont obtenus à partir des mesures provenant des capteurs LIDAR et caméra (la matrice S_{ih} (Eq III.23) est remplie par les valeurs des quaternions des points $\bar{p}_{mih}$ et des normales $\bar{n}_{ih}$). La résolution de ce système permet de déterminer le quaternion, ensuite nous pouvons le convertir en matrice de rotation ${}^{C}R_{L}$. Enfin le vecteur de translation est calculé ${}^{C}\overrightarrow{t}_{L}$ à partir de l'Eq III.15. Ces paramètres, ${}^{C}R_{L}$ et ${}^{C}\overrightarrow{t}_{L}$, définissent la transformation rigide entre les capteurs caméra et LIDAR. Dans la section suivante, nous présentons une approche itérative pour effectuer la calibration caméra/LIDAR afin de la comparer avec notre contribution. Les résultats sont abordés dans la section des résultats expérimentaux.

III.3.2 Approche itérative

En s'appuyant sur les solutions proposées dans l'état de l'art, nous allons comparer les résultats de notre contribution avec ceux de l'approche itérative. L'approche itérative résout le problème géométrique avec une méthode linéaire telle que la SVD, et utilise ensuite cette solution comme une première approximation pour appliquer une optimisation non-linéaire itérative telle que les algorithmes de Gauss-Newton ou de Levenberg-Marquardt, c'est à dire la première approximation sera mise à jour d'une manière itérative dans le but d'être optimisée. La méthode développée dans [35] résout le problème de calibration avec l'approche itérative. L'entrée de cet algorithme est un ensemble de mesures et la sortie est une homographie. Cette dernière est une matrice qui permet la transformation entre les deux repères capteurs. Nous résumons l'algorithme proposé dans [35] comme suit :

Algorithm 1: Algorithme de calibration extrinsèque caméra/LIDAR [35]

Entrées: Mesures : lignes-points
Sorties : Matrice d'homographie H

1. Trouver N > 8 lignes-points mesures
2. Déterminer les coefficients de la matrice qui définit le problème
3. Générer une solution initiale avec la méthode SVD
4. Améliorer la solution initiale en minimisant la fonction objectif avec l'optimisation itérative Levenberg-Marquardt
5. Retourner H

III.3.3 Extraction des caractéristiques

Dans cette partie, nous présentons notre solution proposée de l'extraction des caractéristiques pour les capteurs caméra et LIDAR. L'extraction des caractéristiques a pour objectif de détecter les lignes noires de la mire par la caméra et les deux points des bords qui sont projetés sur la mire par le LIDAR. Notons que les méthodes développées sont applicables à n'importe quelle méthode de calibration qui utilise l'association lignes-points telle que [35].

III.3.3.1 Extraction des points LIDAR

Pour extraire les points projetés du capteur LIDAR sur la mire, nous développons un processus de segmentation (clustering). Chaque segment (cluster) se compose d'un nombre minimum des points qui doivent être éloignés entre eux par une distance inférieure à un seuil fixé Thr. Donc, si la distance euclidienne entre deux points LIDAR consécutifs est inférieure à une distance maximale Thr, un segment sera créé. De plus, chaque segment est défini par un centroïde C_i. Les coordonnées de C_i sont calculées par l'équation suivante :

$$(Cx_i, Cy_i) = (\sum_{i=1}^{n} \frac{p_{x_i}}{n}, \sum_{i=1}^{n} \frac{p_{y_i}}{n}) \tag{III.28}$$

avec p_i un point du capteur LIDAR, p_{i+1} doit appartenir au même cluster et n est le nombre de points d'un cluster donné.
Nous pouvons ajouter un autre paramètre pour fixer le nombre minimum de points qui composent un segment afin d'éliminer les données aberrantes.

La figure III.6 présente les points LIDAR $3D$ projetés dans l'environnement du capteur. Nous utilisons le processus de segmentation dans un espace ouvert. En outre, nous fixons la zone où la mire est placée afin de préciser son emplacement. Donc, cette zone sera limitée par deux valeurs maximale et minimale sur l'axe des ordonnées (les deux lignes vertes dans la figure (III.6). Les points bleus sont les points LIDAR projetés sur la mire de calibration. Notons que pour notre approche

de calibration caméra/LIDAR, nous avons besoin seulement des points des bords qui sont projetés sur la mire.

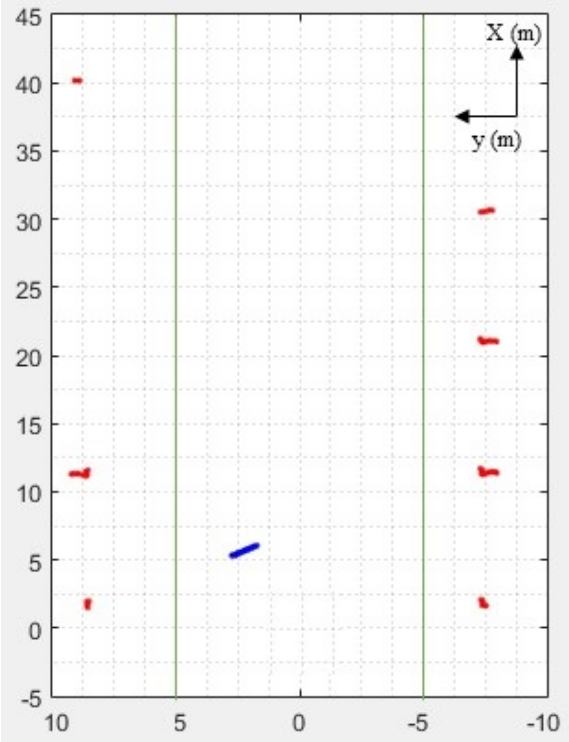

FIGURE III.6: Extraction de points LIDAR projetés sur la mire : les points rouges sont les points de LIDAR (valeurs aberrantes dans le cas de la localisation de mire), les points bleus sont les points LIDAR projetés sur la mire de calibration et les lignes vertes sont les limites de la zone (min et max) où se situe la mire.

III.3.3.2 Extraction des lignes par la transformée de Hough

Dans cette partie, nous détaillons la procédure de détection des lignes noires de la mire utilisée avec la caméra. Cette procédure illustrée par figure III.7 se compose de différentes étapes et s'appuie principalement sur la transformée de *Hough*. Cette transformée permet d'extraire les lignes droites d'une image. Dans la suite, nous présentons les différentes étapes de l'algorithme d'extraction des lignes noires situées dans la mire.

-Filtre de lissage gaussien (Gaussian Smoothing filter) :

Avant d'utiliser la transformée de *Hough*, il est nécessaire d'appliquer un filtre spatial à l'image afin de réduire les bruits et la quantité d'informations à traiter tout

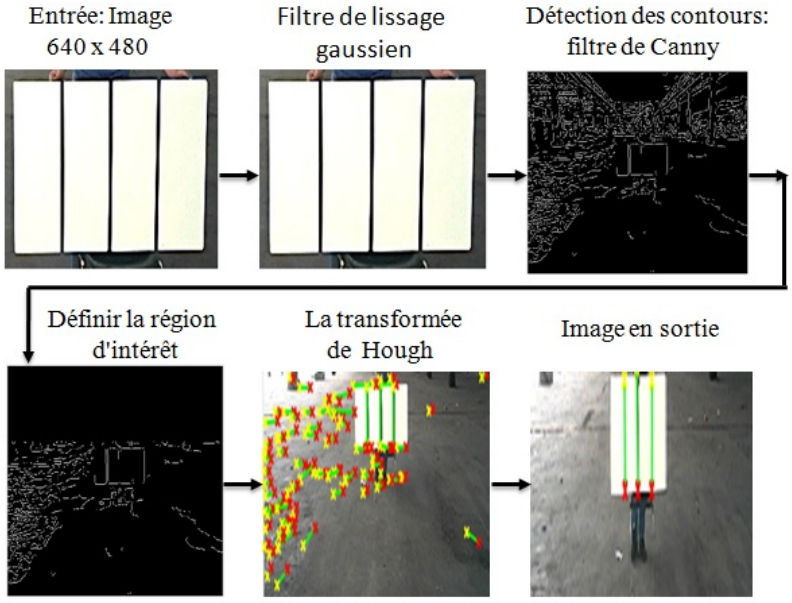

FIGURE III.7: Algorithme d'extraction des lignes noires de la mire avec la transformée de Hough

en préservant le maximum d'informations. Parmi les filtres existants, nous utilisons le lissage gaussien (Gaussian blur) [125] qui est un filtre spatial $2D$ utilisé pour supprimer et réduire les détails et les bruits existants dans une image donnée.

Mathématiquement, appliquer le filtre de lissage gaussien revient à convoler l'image avec un noyau ou un masque gaussien (matrice des pixels). L'idée principale est que les nouveaux pixels de l'image sont créés par une moyenne pondérée des pixels proches, autrement dit, pour créer un nouveau pixel, le noyau gaussien donne plus de poids aux pixels centraux et moins de poids aux pixels voisins.

La figure III.8 présente les résultats de l'application de ce filtre sur une image en variant l'écart-type. En général, un filtre gaussien avec un écart-type $\sigma > 1$ est utilisé pour réduire le bruit, et si $\sigma < 1$ c'est dans le but de fournir une image pour faire un masque flou [126]. Notons que plus l'écart-type σ est grand, plus le flou appliqué à l'image sera considérable. Dans notre cas, ce filtre est utilisé afin de supprimer au maximum les détails ainsi que les bruits.

-Détection des contours avec le filtre de Canny :

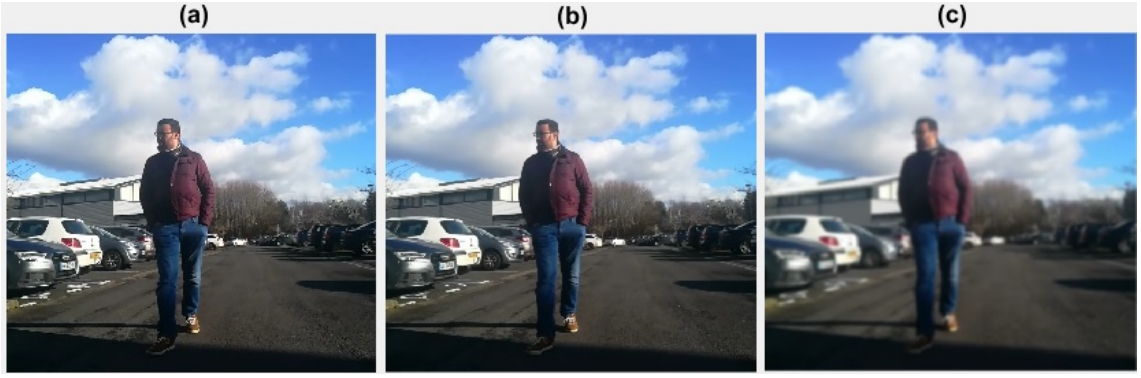

FIGURE III.8: Filtre de lissage gaussien (a) Image originale (b) Filtre de lissage gaussien : moyenne nulle, écart-type $\sigma = 0.5$ (c) Filtre de lissage gaussien : moyenne nulle, écart-type $\sigma = 2$

Après la réduction des détails et bruits, la deuxième étape consiste à appliquer un algorithme de détection des contours. Cet algorithme permet de réduire d'une manière considérable la quantité de données à traiter en conservant seulement les informations les plus pertinentes. L'idée est que dans une image en niveaux de gris, un changement brutal de la valeur de pixel (l'intensité lumineuse) caractérise un contour. Il existe plusieurs méthodes pous assurer la détection des contours telles que le filtre de *Canny* [127] ou celui de *Sobel* [128]. Ces deux méthodes cherchent les extremums de la dérivée première, en général les maximums locaux de l'intensité du gradient. La différence entre ces méthodes est les masques qu'elles utilisent.
Nous utilisons le filtre de *Canny*, car il garantit une meilleure détection des contours ainsi que leurs orientations, notamment dans le cas où l'image contient de bruit de manière significative [129] (présence d'informations parasites qui s'ajoutent de façon aléatoire aux détails). Ce filtre retourne un image avec des contours détectés, à ce stade, la transformée de *Hough* est appliquée.

-Détection des droites avec la transformée de *Hough* : La transformée de Hough est une technique de reconnaissance des formes inventée par *Paul Hough*. Cette technique est un outil standard dans le domaine de vision artificielle. Elle permet de détecter des objets bien précis dans les images tels que les lignes et les cercles [130]. Dans notre cas, nous l'utilisons pour détecter les lignes noires de la mire.
L'idée est de transformer chaque point de l'espace $x-y$ (repère cartésien de l'image)

en espace de paramètres (repère polaire). Cette méthode définit deux paramètres : ***(i)*** r représente la longueur d'une normale de l'origine à cette ligne **(*ii*)** θ est l'orientation de r par rapport à l'axe x. L'équation de chaque ligne est définie par :

$$r = xcos(\theta) + ysin(\theta) \tag{III.29}$$

Après la transformation de tous les points dans l'espace paramétrique, les pics locaux associés aux candidats des lignes dans l'espace $x - y$ sont extraits [130]. Notons que d'autres paramètres sont requis pour la transformation de $Hough$ afin de filtrer les lignes détectées :

(i) Distance entre deux segments : lorsque une distance entre deux segments est inférieure à la valeur spécifiée par l'utilisateur, la fonction de $Hough$ fusionne les deux segments en un seul.

(ii) Longueur de ligne minimale : ce paramètre permet à la fonction de $Hough$ de rejeter les lignes qui sont plus courtes que la valeur spécifiée.

-Filtrage des lignes aberrantes : L'étape précédente permet d'appliquer la transformée de $Hough$ pour détecter les lignes de la mire dans une image pour une pose donnée. Cependant, cette transformée retourne les coordonnées de toutes les lignes dans une image, même les lignes qui correspondent aux données aberrantes. Afin d'effectuer le filtrage et extraire les données pertinentes, il y a des étapes supplémentaires qui sont prises pour effectuer l'extraction des lignes noires. Ces étapes sont basées sur les connaissances a priori :

(i) **Définir la régions d'intérêt** : A priori, nous savons où la mire est installée dans le champ de vision de caméra, donc nous pouvons limiter la zone du traitement. Cette étape consiste à mettre en évidence les zones intéressantes pour être analysées afin de réduire la quantité d'information à traiter et le nombre de fausses détections.

(ii) **Éliminer les lignes courtes** : En fixant un seuil du nombre minimal de points constituant un segment, les lignes qui sont plus courtes que la valeur spécifiée, seront rejetées.

(iii) **Éliminer les lignes horizontales** : Sachant que les lignes noires de la mire dans une pose donnée sont des lignes verticales, les lignes horizontales seront considérées des données aberrantes. La distinction des lignes verticales par rapport à ceux horizontales s'appuie sur les pentes des droites détectées.

(iv) **Sélection suivant la couleur** : En utilisant l'information de couleur nous pouvons sélectionner davantage les informations pertinentes. A priori, nous savons que les lignes de la mire ont des couleurs noires et leurs pixels voisins ont des couleurs blanches.

La sortie de cette étape est les cordonnées des lignes de la mire que nous utilisons pour déterminer les paramètres de calibrations.

III.3.4 Algorithme proposé de calibration caméra/LIDAR

La figure III.9 résume notre algorithme de calibration caméra/LIDAR. Cela nécessite de mettre la mire de calibration dans les champs de vision des capteurs caméra et LIDAR à différentes positions et orientations tout en assurant que tous les capteurs peuvent la détecter. La couleur de la mire est blanche avec trois lignes noires. Pour chaque pose, nous collectons les caractéristiques pour les deux capteurs : les lignes noires détectées par caméra et les points des bords détectés par le LIDAR. L'extraction des lignes noires est basée sur la transformée de *Hough* et l'extraction des points des bords est effectuée par le processus de segmentation. En utilisant ces données, nous résolvons le système d'équations donné par l'Eq III.27 pour obtenir les paramètres extrinsèques de calibration.

III.3.5 Répercussion de la précision de la calibration sur la fusion de données

Pour fusionner les données entre les capteurs, il est nécessaire d'estimer et modéliser les erreurs qui sont impliquées dans le traitement des données. Ces incertitudes sont causées par différents types d'erreurs tels que : **(i)** Les erreurs aléatoires provenant des bruits des mesures, les écart-types de ces erreurs sont souvent fournis par les constructeurs. **(ii)** Les erreurs de calibration. Par conséquent, le processus

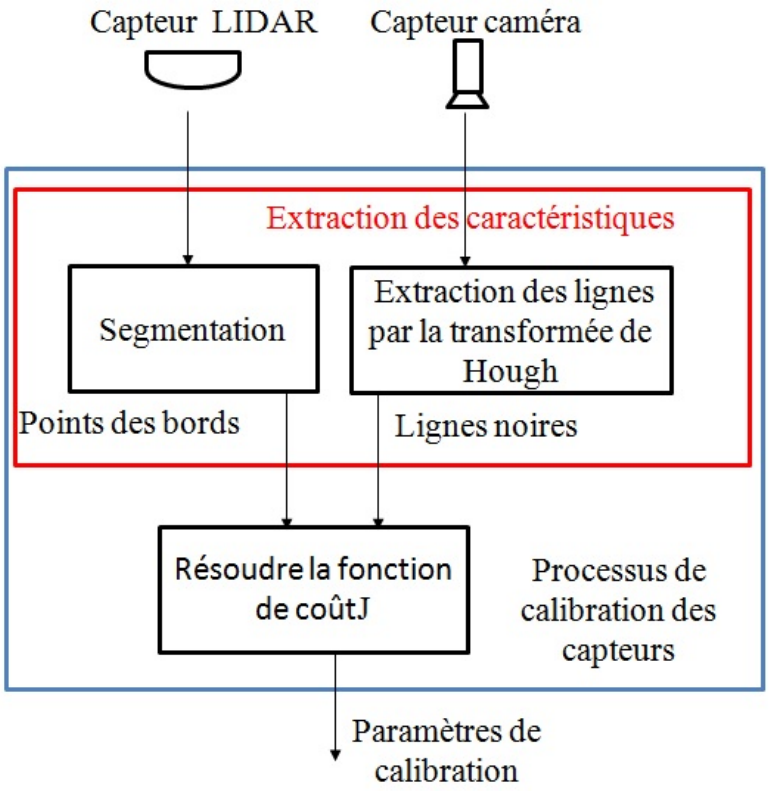

FIGURE III.9: Algorithme proposé de calibration caméra/LIDAR

de fusion de données doit prendre en compte les incertitudes pour assurer un bon fonctionnement.

Afin de montrer la répercussion des erreurs sur le processus de fusion, nous simulons le comportement de deux capteurs qui détectent simultanément la position d'un obstacle donné. Pour ce faire, nous appliquons l'approche bayésienne pour fusionner les positions. Dans [72], Baig et al. utilisent cette approche pour fusionner les positions des obstacles détectés par un LIDAR et une caméra stéréoscopique. Les incertitudes des positions sont représentées par des distributions gaussiennes $2D$. Si X est la vraie position de l'objet détecté, en utilisant la fusion bayésienne, la probabilité fusionnée de la position $P_F[x_F\, y_F]^T$ est donnée par l'équation suivante :

$$P_{rob}(P|X) = \frac{e^{\frac{-(P-X)^T R^{-1}(P-X)}{2}}}{2\pi\sqrt{|R|}} \tag{III.30}$$

avec P est la position fusionnée et R est la matrice de covariance fusionnée données

par :

$$P = \frac{P_1/R_1 + P_2/R_2}{1/R_1 + 1/R_2} \text{ et } 1/R = 1/R_1 + 1/R_2$$

avec P_1 et R_1 sont respectivement la position et la matrice de covariance du capteur 1, P_2 et R_2 sont ceux du capteur 2.

Dans la simulation, nous varions les matrices de covariance d'erreurs des capteurs. Pour cela, nous définissons les bruits des mesures comme des bruits gaussiens avec une moyenne nulle et un écart-type de 120 mm pour les deux capteurs. Nous modélisons le déplacement de l'obstacle détecté par un mouvement uniforme avec une vitesse constante. Notons que $R_1 = R_{m1}$, cela signifie que la matrice de covariance du capteur 1 présente uniquement l'erreur de mesure. Cependant, la covariance du capteur 2 présente à la fois les erreurs de mesure et de calibration donc $R_2 = R_{m2} + R_c$ (car les mesures du capteur 2 seront projetées dans le repère du capteur 1). Nous considérons que l'erreur de calibration du capteur 2 sera faible dans la première simulation tandis que nous l'augmentons dans la seconde simulation.

Les figures III.10 et III.11 présentent les résultats obtenus, les postions détectées par les deux capteurs ainsi que celle fusionnée. Il est clair que dans la première simulation (Figure III.10), lorsque le capteur 2 présente une faible valeur d'erreur de calibration, la position fusionnée est située entre les deux positions qui sont fournies par les capteurs. Par contre, lorsque l'erreur de calibration augmente, la position fusionnée suit la position qui est fournie par le capteur 1, car elle a l'erreur la plus petite (figure III.11).

Selon ces simulations, le résultat de fusion est une combinaison des deux mesures pondérées par leurs matrices des covariances de bruit. Par conséquent, la projection de la mesure du capteur 2 sera biaisée si l'erreur de calibration est importante et la position fusionnée se contentera des mesures du capteur 1.

III.3.6 Évaluation de la qualité de calibration

En pratique, il est difficile d'obtenir ou d'évaluer la vérité terrain des paramètres extrinsèques entre les capteurs caméra et LIDAR. Dans [131], Li et al. définissent un indicateur de précision en se basant sur les résultats de projection des points LIDAR

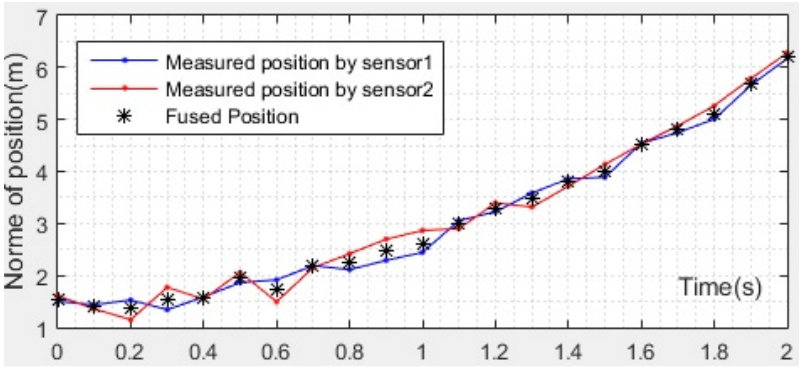

FIGURE III.10: Fusion de deux positions pondérée avec des matrices de covariance similaires

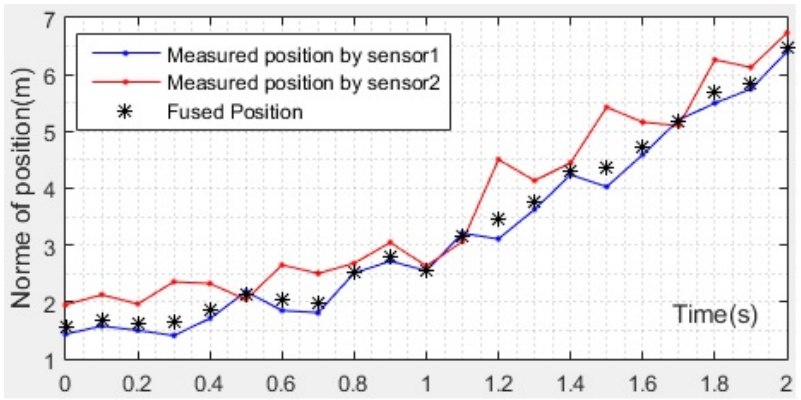

FIGURE III.11: Fusion de deux positions avec l'augmentation de l'erreur de calibration du capteur 2

dans le plan image afin d'évaluer la qualité de calibration. Dans [132], Shang et al. définissent deux critères afin d'évaluer les paramètres de calibration : le premier concerne l'erreur de translation et le second concerne l'erreur d'orientation. Dans nos travaux, nous proposons la somme des carrés des résidus (Eq III.11) comme un indicateur de calibration. Tant que ce critère tend vers zéro, nous obtenons une bonne performance du processus de calibration.

III.4 Résultats expérimentaux

Afin de valider la méthode multi-lignes proposée, nous avons mené une série d'expériences avec la navette autonome *ARMA* de Navya. Cette navette est équipée par un ensemble de capteurs. Parmi ces capteurs, nous utilisons le capteur

LIDAR mono-nappe avec un angle de balayage de 180° et une résolution angulaire de $0,25°$. Également, nous utilisons la caméra avec une résolution de 640×480 qui est modélisée avec le modèle sténopé. Pour synchroniser les données entre les deux capteurs, nous développons une application avec *RTMaps* 4.0 permet de générer les données de LIDAR et de caméra simultanément pour chaque pose. Nous employons une mire de calibration de couleur blanche avec trois lignes noires ($d = 3$).

III.4.1 Extraction des lignes de la mire de calibration

La figure III.12.a montre la mire de calibration utilisée. Nous avons appliqué l'algorithme présenté dans la section III.3.3.2 pour extraire les lignes. La figure III.12.b montre les résultats préliminaires, la détection des lignes qui inclut les lignes noires ainsi que des fausses détections. Quant à la figure III.12.c, elle montre le résultat final de détection.

Notons que le traitement d'image basé sur la détection de couleur n'est pas souvent efficace (une étape de filtrage présentée dans III.3.3.2). À titre d'exemple, si nous cherchons la couleur blanche (son niveau de gris est 255), il est acceptable de choisir les pixels qui ont des valeurs des niveaux de gris appartiennent à l'intervalle [200..250]. Mais en pratique et en raison de différents facteurs tels que les bruits, la résolution et la qualité de caméra, les valeurs des niveaux de gris des pixels blancs peuvent dépasser cette plage (inférieurs à 200).
Pour appliquer notre approche de calibration, nous avons besoin de deux points de chaque ligne noire détectée pour déterminer les vecteurs normaux.

III.4.2 Résultats de calibration extrinsèque caméra/LIDAR : solution analytique

Nous avons déplacé la mire suivant une distance de 3 à 9 m. Afin d'accomplir le processus de calibration, nous avons collecté 21 correspondances point-vecteur normal pour l'approche mono-ligne, et 5 correspondances pour l'approche proposée. Pour comparer les résultats, nous calculons la distance moyenne entre les co-

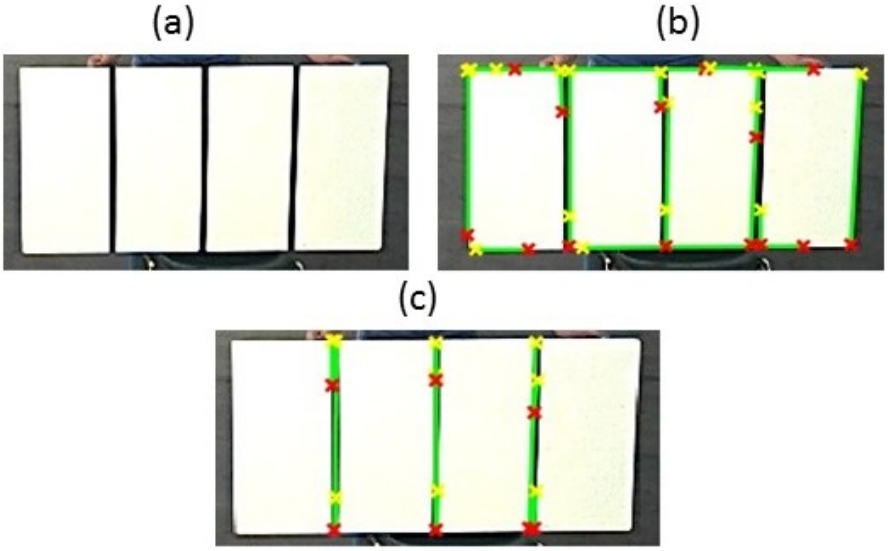

FIGURE III.12: Les résultats d'extraction des lignes noires de la mire de calibration : **(a)** La mire utilisée pour notre approche de calibration **(b)** Les résultats préliminaires : Extraction des lignes noires en présence des fausses détections **(c)** Résultat final : Extraction des lignes noires de la mire

ordonnées des pixels des points LIDAR projetés sur la mire. Le résultat du tableau III.1 présente la distance absolue moyenne pour l'axe u, l'axe v et la distance moyenne entre les pixels correspondants. Selon ce résultat, les deux approches sont très proches. De plus, la valeur de la fonction de coût (Eq III.11) est 0.66 pour l'approche mono-ligne et 0.45 pour la notre, ce qui a permis de réduire la moyenne des carrés des résidus. Par conséquent, ces résultats ont validé le bon fonctionnement de l'approche multi-lignes ainsi que pour celle de mono-ligne avec l'avantage de réduire le nombre requis de poses.

La figure III.13 montre les résultats de projection de deux approches pour quatre poses choisies. Par ailleurs, il est nécessaire de prendre des poses de la mire à partir de différentes orientations pour déterminer les six degrés de liberté. En outre, il est recommandé de faire pivoter la mire, autour de l'axe z, pour permettre au vecteur normal $\overrightarrow{n}_{ih}$ de s'étendre dans les trois directions, afin d'avoir le paramètre de profondeur.

TABLE III.1: La différence moyenne entre les deux approches

Distance moyenne suivant l'axe u (pixel)	Distance moyenne suivant l'axe v (pixel)	Distance moyenne absolue (pixel)
2.399	3.6816	3.3927

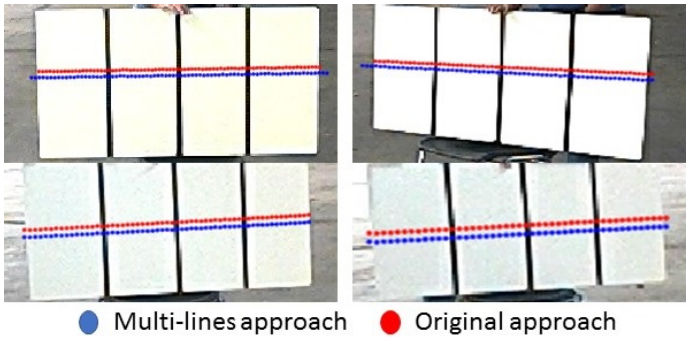

FIGURE III.13: Projection des points LIDAR sur la mire de calibration

III.4.3 Comparaison des résultats : solutions analytique et itérative

Pour l'approche itérative, nous avons collecté 23 correspondances de points-lignes pour estimer l'homographie. La figure III.14 montre les résultats de projection en utilisant la solution itérative pour trois poses. Les lignes rouges représentent la solution initiale et les lignes bleues représentent la solution optimisée en utilisant l'algorithme de $Levenberg - Marquardt$.

Par ailleurs tableau III.2 montre la différence moyenne entre les deux approches. Selon ce résultat, les deux implémentations sont très proches. Cependant, nous avons obtenu des résultats séminaires à ceux de notre approche, mais avec de moins des poses : 23 pour la solution itérative contre seulement 5 de la solution analytique.

TABLE III.2: La différence moyenne entre les deux approches : analytique et itérative

Distance moyenne suivant l'axe u (pixel)	Distance moyenne suivant l'axe v (pixel)	Distance moyenne absolue (pixel)
3.44	3.45	3.8

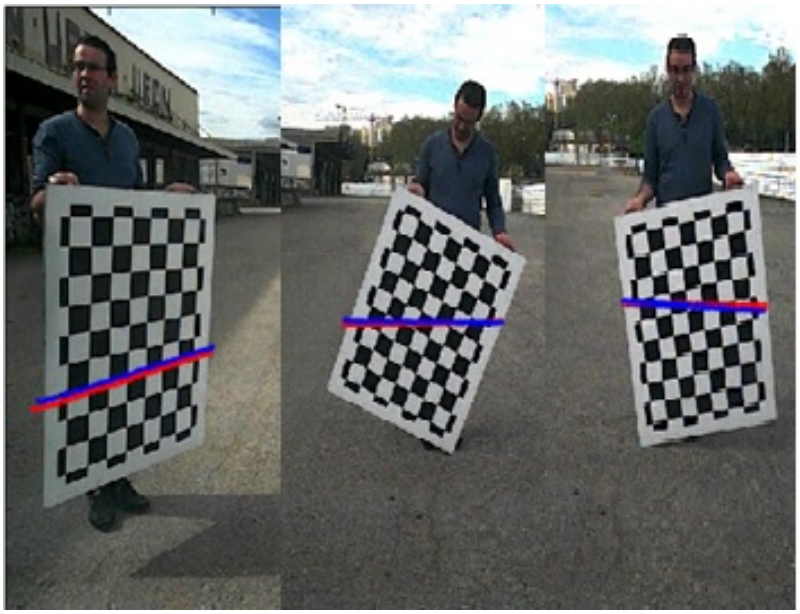

FIGURE III.14: Projection des points LIDAR sur la mire de calibration (Les lignes rouges : solution initiale, Les lignes bleues : la solution optimisée)

III.4.4 Discussion

Nous avons présenté dans ce chapitre notre contribution pour la calibration caméra/LIDAR. Par rapport à d'autres approches dont la précision dépend d'une estimation initiale précise, la solution proposée donne un résultat optimal pour un nombre de poses réduit. En fait, en augmentant le nombre de lignes par pose, nous réduisons les bruits qui affectent la qualité du processus de calibration. Donc, le nombre d'observations a été réduit pour un résultat précis.

Pour les approches analytiques, la détermination du vecteur de translation est délicate car elle est calculée par la somme illustrée par l'équation III.15 qui peut être affectée par les bruits. En effet, l'erreur d'estimation, causée par les bruits dans la première étape va se propager dans la deuxième étape et ainsi de suite, ce qui affecte la précision du résultat final.

Pour résoudre le problème de calibration par la méthode itérative, il est plus simple de déterminer la solution avec des outils de programmation tels que Matlab ou une bibliothèque C/C ++ [133]. Dans ce cas, nous obtenons une solution initiale, par la méthode de SVD, et qui sera mise à jour de manière itérative par une optimisation non-linéaire. D'autre part, l'approche analytique fournit plus de 20 solutions (à

part les solutions complexes), donc nous avons besoin d'un solveur de bonne performance. À partir de ces 20 solutions, la solution choisie doit satisfaire la contrainte de la matrice de rotation (le déterminant est égal à 1).

III.5 Conclusion

Dans ce chapitre, nous avons abordé le problème d'alignement des capteurs caméra/LIDAR $2D$. Pour résoudre ce problème, nous avons proposé une solution qui nous a permis d'avoir un résultat précis pour un nombre de poses réduit (5 poses), ce qui diminue le temps nécessaire pour la mise en œuvre de ce processus. Nous avons aussi présenté une chaîne de traitement, permettant d'extraire les caractéristiques de la mire de calibration par les deux capteurs. Cette chaîne s'appuie sur deux fonctionnalités principales qui sont : la détection des lignes par la caméra, avec la transformée de $Hough$, et un processus de segmentation pour détecter les points des bords de LIDAR projetés sur la mire. Ces deux fonctionnalités ont pour objectif de réaliser une acquisition de données automatisée dans la mesure du possible. En se basant sur l'état de l'art, nous avons comparé notre approche analytique avec celle d'itérative pour valider son bon fonctionnement.

L'emploi de cette calibration peut être illustré par les deux cas suivants : **(1)** Si nous fusionnons les positions des obstacles détectées par une camera stéréo et un LIDAR, dans ce cas, la calibration permet de faire l'association entre les positions qui sont exprimées dans les deux repères différents. **(2)** Le deuxième cas, si le LIDAR détermine les régions d'intérêt dans l'image, nous pouvons ensuite classifier les objets situés dans ces régions sans parcourir toute l'image, ce qui permet d'augmenter la rapidité de traitement.

Dans le chapitre suivant, nous présenterons notre approche de fusion multi-capteurs entre une caméra et un radar.

Chapitre IV

Fusion multi-capteurs : caméra/radar

Sommaire

IV.1 Introduction

Ce chapitre présente notre deuxième contribution celle de fusion multi-capteurs. Comme nous avons mentionné précédemment dans la section II.4.5, chaque capteur a ses propres avantages et inconvénients, c'est pourquoi seule une fusion appropriée aboutit à un résultat précis. Il est donc important de développer une approche de fusion multi-capteurs pour tirer profit des avantages de chaque capteur. Compte tenu de la complémentarité des capteurs caméra et radar, cette combinaison est souvent utilisée dans la recherche pour les applications des véhicules autonomes.

Dans ce chapitre, nous présentons notre approche de détection et classification d'obstacles en s'appuyant sur la théorie des fonctions de croyance. Notre objectif principal est d'améliorer la détection et la reconnaissance d'obstacles, y compris les piétons. Les hypothèses pour classifier les obstacles sont déduites à partir du paramètre RCS fourni par le radar, qui reflète la nature d'obstacle détecté, et la classification assurée par la caméra. Les résultats de simulation et les tests expérimentaux préliminaires sont présentés.

Ce chapitre est structuré comme suit. Dans la section IV.2 nous justifions nos choix des fonctions de croyance pour la fusion multi-capteurs. Dans la section IV.3, nous présentons la théorie de croyance ainsi que ses principaux concepts et ses outils pour modéliser et gérer les incertitudes. Nous détaillons l'approche proposée de fusion entre la caméra et le radar dans la section IV.4. Quant aux résultats, nous les présentons dans la section IV.5.

IV.2 Proposition et justification

Dans le contexte du véhicule autonome, l'emploi de fusion multi-capteurs a pour objectif d'éviter les fausses alarmes et réduire les détections manquées dans un environnement incertain. Dans le cadre de cette thèse, nous choisissons la théorie de croyance, une alternative importante à la théorie de probabilité [49], pour la représentation des connaissances ainsi que pour modéliser et gérer les incertitudes provenant des mesures des capteurs. La théorie de croyance présente plusieurs avantages,

notamment :

(i) Elle est très expressive, permet de représenter différents niveaux d'ignorance, ne nécessite pas de probabilités a priori comme le cas pour l'approche bayésienne, et gère les situations de conflit lorsque des preuves opposées apparaissent [134, 135].

(ii) La théorie de croyance modélise souplement les connaissances et fournit des outils riches permettant de gérer les différents types d'imperfection et de fusionner ces connaissances [136].

(iii) Cette théorie est bien adaptée au problème de fusion de données et modélise bien la performance des différents capteurs [21].

Nous revenons à ces concepts avec plus de détails dans la section suivante. La figure IV.1 décrit une interaction entre un véhicule autonome et son environnement. La détection d'objets est assurée en utilisant un ensemble de capteurs extéroceptifs tels que le radar et la caméra. Ces derniers, permettent au véhicule de percevoir l'environnement grâce aux observations réceptionnées. En s'appuyant sur ces observations, des croyances sont déduites, puis fusionnées afin de prendre une décision dans le but d'obtenir des données plus informatives et par la suite, améliorer la précision de détection.

Afin de modéliser et gérer les incertitudes provenant des mesures des capteurs, nous devons répondre aux question suivantes, comment nous pouvons représenter la connaissance provenant d'une observation d'un capteur donné sous forme des fonctions de croyance ? Comment nous fusionnons les fonctions de croyance, et par la suite comment nous prenons la décision ?

Dans la partie suivante, nous introduisons la théorie de croyance et ses outils théoriques pour la gestion des informations imparfaites.

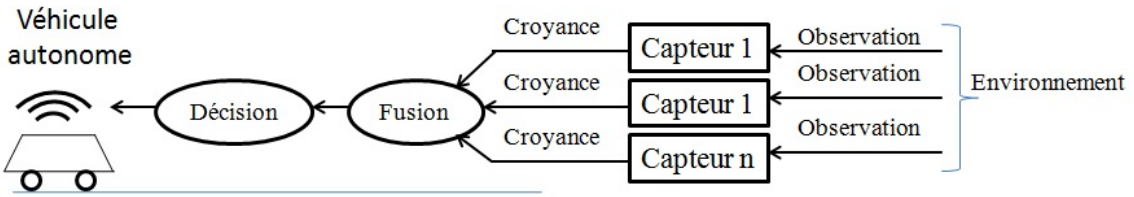

FIGURE IV.1: Interaction entre un véhicule autonome et son environnement

IV.3 Théorie des fonctions de croyance : contexte et outils

Dans cette section, nous décrivons les concepts fondamentaux de la théorie des fonctions de croyance. Également, nous présentons les opérateurs principaux de combinaison pour la fusion d'informations dans le cadre de cette théorie.

IV.3.1 Théorie des fonctions de croyance (Dempster-Shafer)

La théorie de *Dempster-Shafer* a été introduite par *Arthur P. Dempster* [137] en 1968, dans le contexte de l'inférence statistique, afin de développer un cadre général de représentation des incertitudes. En 1976, elle a attiré l'attention de *Glenn Shafer* pour la modélisation des incertitudes, ce qui lui permettait d'établir les fonctions de base de la théorie de croyance dans son livre *A Mathematical Theory of Evidence* [138]. C'est pour cette raison, que cette théorie porte le nom de *Dempster-Shafer*, aussi, nous trouvons d'autres appellations telles que la théorie des croyances ou la théorie de l'évidence.

La théorie des fonctions de croyance a pour objectif de développer des outils théoriques pour la gestion des informations imparfaites, caractérisées par des imprécisions et incertitudes. Au début des années 1990, *Philippe Smets* a développé un cadre formel générique pour la représentation et la combinaison des connaissances, basé sur la définition de fonctions de croyance, connu sous le nom de modèle de croyance transférable (TBM : Transferable Belief Model) [139]. L'un des points fondamentaux qui caractérisent le TBM est la différenciation des niveaux *Repré-*

sentation des connaissances et *Décision* [140]. Les mécanismes de raisonnement du TBM sont donc regroupés en deux niveaux [141] :

(i) **Niveau crédal** (du latin *credo* signifiant *je crois*) pour représenter et combiner les informations.

(ii) **Niveau pignistique** (du latin *pignus* signifiant *un pari*) pour assurer la prise de décision.

IV.3.2 Présentation des concepts

La théorie des fonctions de croyance suppose connu au départ un ensemble d'hypothèses Ω, appelé cadre de discernement qui se définit comme suit [139] :

$$\Omega = \{\omega_1, \omega_2, ..., \omega_N\} \tag{IV.1}$$

où ω est une hypothèse et N est le nombre d'hypothèses.
Les hypothèses sont supposées exclusives signifiant que deux hypothèses ne peuvent être vraies simultanément.

À partir du cadre de discernement Ω, nous supposons que l'*espace puissance* est composé de 2^Ω propositions de Ω [140] :

$$2^\Omega = \{\emptyset, \{\omega_1\}, \{\omega_2\}, ..., \{\omega_N\}, \{\omega_1 \cup \omega_2\}, \{\omega_1 \cup \omega_3\}, ..., \Omega\} \tag{IV.2}$$

où 2^Ω rassemble tous les sous-ensembles possibles formés des hypothèses et des unions d'hypothèses de Ω.
Nous distinguons des représentations différentes pour les fonctions de croyance. La représentation la plus communément utilisée est celle de la fonction de masse de croyance notée m^Ω, appelée aussi en anglais *Basic Belief Assignment* (BBA), et définie par [135] :

$$\begin{aligned} m^\Omega : 2^\Omega &\rightarrow [0, 1] \\ A &\longmapsto m^\Omega(A) \end{aligned} \tag{IV.3}$$

Satisfaisant :

$$\sum_{A \subseteq \Omega} m(A) = 1 \tag{IV.4}$$

où $A \subseteq \Omega$ est une proposition.

Nous citons d'autres concepts utilisés dans la théorie de croyance [135, 142] :

- La masse $m(A)$ représente le degré de croyance attribué à la proposition A et qui n'a pas pu, compte tenu de l'état de la connaissance, être affectée à un sous-ensemble plus spécifique que A (part de croyance allouée à A).
- Un élément A dont la masse est non nulle ($m_S^\Omega(B) > 0$) est appelé élément focal de la BBA.
- La masse $m(\Omega)$ représente le degré de l'ignorance et si $m(\Omega) = 1$, il s'agit de l'ignorance totale.
- $m(\emptyset)$ représente le degré de conflit. La notion de conflit dans la théorie des fonctions de croyance est ainsi principalement définie par la masse sur l'ensemble vide à l'issue de fusion des fonctions de croyance [143], nous revenons à cette notions dans la section IV.3.3.

Au niveau notation de la fonction de masse, le cadre de discernement, exemple Ω, est mis en exposant dans la fonction de masse et la source de croyance S est mise en indice, donc la fonction de masse est représentée par m_S^Ω.

Afin d'expliquer davantage ces notions, nous présentons un exemple dans la suite.

Exemple **:** Résultat de lancer d'une pièce monnaie parfaitement équilibrée, qui peut être pile ou face. L'objectif est de déterminer la probabilité d'obtenir soit pile ou face.

Nous considérons $\Omega = \{pile, face\}$, le cadre de discernement formé par l'ensemble des hypothèses. Les hypothèses sont supposées exclusives signifiant que deux hypothèses ne peuvent être vraies simultanément, autrement dit, à chaque lancer nous obtenons soit face ou pile. De plus, elle doivent être exhaustives (la réponse est obligatoirement l'une des hypothèses). Notons que le cade de discernement décrit l'ensemble des réponses possibles à un problème donné.

À partir du cadre de discernement Ω, nous déduisons l'ensemble noté 2^Ω, comprenant l'ensemble des "2^2" sous-ensembles de Ω :

$$2^{\Omega} = \{\emptyset, \{pile\}, \{face\}, \{pile \cup face\}\} \tag{IV.5}$$

où 2^{Ω} rassemble tous les sous-ensembles possibles formées des hypothèses et unions d'hypothèses de Ω.
Également, nous déterminons les probabilités (notées P) dans le monde probabiliste et les fonctions des masses de croyance (notées m^{Ω}) dans la théorie de croyance (monde crédibiliste) :

(i) Dans le monde probabiliste nous calculons les probabilités P(pile) et P(face).

(ii) Dans la la théorie de croyance nous calculons les fonctions des masses $m^{\Omega}(\emptyset)$, $m^{\Omega}(pile)$, $m^{\Omega}(face)$ et $m^{\Omega}(\{pile \cup face\})$.

IV.3.3 Opérateurs de combinaison pour la fusion des fonctions de croyance

Pour combiner les fonctions de masse, il existe différents opérateurs. Chaque opérateur a ses propres propriétés et il est utilisé dans un cas bien précis. En fait, l'emploi de ces opérateurs dépend des contraintes d'une application donnée telles que la fiabilité et la dépendance des informations fournies par les sources des croyances. Autrement dit, pour fusionner les masses de croyance, aucun modèle unique est adéquat pour toutes les situations [144], donc il est nécessaire de déterminer quel opérateur de combinaison le plus approprié pour une situation donnée.
Dans la suite, nous présentons cinq opérateurs pour combiner les masses de croyance.

IV.3.3.1 Opérateur conjonctif

Cette approche est introduite par *Dempster* et reprise par *Shafer*. La combinaison conjonctive est une règle de combinaison élémentaire consiste à prendre les sommes de toutes les possibilités de conjonction entre tous les objets [145]. Cet opérateur suppose l'indépendance et la fiabilité des sources, l'élément neutre est Ω et l'élément absorbant est l'ensemble vide. Pour combiner les masses de croyance nous utilisons la formule suivante [135] :

$$m_{12}(A) = \sum_{B \cap C = A \mid B,C \subseteq \Omega} m_1(B) m_2(C), \quad \forall A \subseteq \Omega \tag{IV.6}$$

où m_{12} représente la fonction des croyances fusionnées, m_1 est la fonction de croyance fournie par la première source de croyance et m_2 est celle de deuxième source.
Le tableau IV.1 présente la règle de fusion de deux fonctions des masses m_1 et m_2 qui ont un cadre de discernement $\Omega = \{A, B\}$, avec l'opérateur conjonctif.

m_1 \ m_2	$\emptyset$	A	B	Ω
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
A	$\emptyset$	A	$\emptyset$	A
B	$\emptyset$	$\emptyset$	B	B
Ω	$\emptyset$	A	B	Ω

TABLE IV.1: Fusion de masses avec l'opérateur conjonctif [10]

Exemple 1 : Opérateur conjonctif
Nous considérons les deux fonctions des masses m_1 et m_2 suivantes :

Propositions	$\emptyset$	A	B	Ω
m_1	0	0.4	0.6	0
m_2	0	0.1	0.9	0

TABLE IV.2: Exemple 1 : valeurs des fonctions des masses

Le tableau IV.3 présente les résultats de fusion :

m_1 \ m_2	$\emptyset$ 0	A 0.1	B 0.9	Ω 0
$\emptyset$ 0	$\emptyset$ 0	$\emptyset$ 0	$\emptyset$ 0	$\emptyset$ 0
A 0.4	$\emptyset$ 0	A 0.04	$\emptyset$ 0.36	A 0
B 0.6	$\emptyset$ 0	$\emptyset$ 0.06	B 0.54	B 0
Ω 0	$\emptyset$ 0	A 0	B 0	Ω 0

TABLE IV.3: Exemple 1 : fusion de masses avec l'opérateur conjonctif

Donc le résultat de fusion de masses m_{12} est calculé comme suit :
$m_{12}(\emptyset) = 0$

$m_{12}(A) = 0.04$

$m_{12}(B) = 0.54$

$m_{12}(\Omega) = 0.36 + 0.06 = 0.42$

IV.3.3.2 Opérateur disjonctif

L'emploi de l'opérateur conjonctif exige que toutes les sources d'informations soient fiables. Cependant, cette contrainte est souvent difficile à respecter. Pour répondre à cette contrainte, l'opérateur disjonctif combine les masses de croyance lorsqu'au moins une source est fiable, mais sans savoir laquelle. Cet opérateur suppose l'indépendance des sources, l'élément neutre est l'ensemble vide et l'élément absorbant est Ω. Pour combiner les masses de croyance nous utilisons la règle suivante [135] :

$$m_{12}(A) = \sum_{B \cup C = A \mid B,C \subseteq \Omega} m_1(B) m_2(C), \quad \forall A \subseteq \Omega \tag{IV.7}$$

Le tableau IV.4 présente la règle de fusion de deux fonctions des masses m_1 et m_2 qui ont un cadre de discernement $\Omega = \{A, B\}$, avec l'opérateur disjonctif.

m_1 \ m_2	∅	A	B	Ω
∅	∅	A	B	Ω
A	A	A	Ω	Ω
B	B	Ω	B	Ω
Ω	Ω	Ω	Ω	Ω

TABLE IV.4: Fusion de masses avec l'opérateur disjonctif [10]

Exemple 2 : Opérateur disjonctif

Nous considérons les deux fonctions des masses m_1 et m_2 suivantes :

Propositions	∅	A	B	Ω
m_1	0	0	0.6	0.4
m_2	0	0	0.9	0.1

TABLE IV.5: Exemple 2 : valeurs des fonctions des masses

Le tableau IV.6 présente les résultats de fusion :

m_2 / m_1	$\emptyset$	A	B	Ω
	0	0	0.9	0.1
$\emptyset$ 0	$\emptyset$	A	B	Ω
	0	0	0	0
A 0	A	A	Ω	Ω
	0	0	0	0
B 0.6	B	Ω	B	Ω
	0	0	0.54	0.06
Ω 0.4	Ω	Ω	Ω	Ω
	0	0	0.36	0.04

TABLE IV.6: Exemple 2 : fusion de masses avec l'opérateur disjonctif

Donc le résultat de fusion de masses m_{12} est calculé comme suit :

$m_{12}(\emptyset) = 0$

$m_{12}(A) = 0$

$m_{12}(B) = 0.54$

$m_{12}(\Omega) = 0.06 + 0.36 + 0.04 = 0.46$

IV.3.3.3 Opérateur de Dempster

Cet opérateur est défini par *Dempster* dans le cadre de la théorie des fonctions de croyance [146]. L'opérateur de *Dempster* est associatif et commutatif, il suppose l'indépendance et la fiabilité de deux sources d'informations. La règle de fusion de *Dempster* est calculée en deux étapes : la fusion conjonctive suivie de la normalisation de la masse conflictuelle $m(\emptyset)$ [10] :

$$\begin{aligned} m^{\Omega}_{1\oplus 2}(\emptyset) &= 0 \\ m^{\Omega}_{1\oplus 2}(A) = (m^{\Omega}_1 \oplus m^{\Omega}_2)(A) &= \frac{\sum\limits_{B\cap C=A} m^{\Omega}_1(B) m^{\Omega}_2(C)}{1-K}, \quad A \neq \varnothing \\ K &= \sum_{B\cap C=\varnothing} m^{\Omega}_1(B) m^{\Omega}_2(C) \end{aligned} \tag{IV.8}$$

où $\oplus$ est le symbole de l'opérateur de *Dempster* et K mesure la quantité de conflits entre les deux ensembles de masse.

Dans ce cas, la masse de conflits (ensemble vide) sera répartie sur chaque état présent proportionnellement à sa masse. Cependant, la règle de *Dempster* produit

des résultats contre-intuitifs lorsque le conflit est élevé.

Exemple 3 : Opérateur *Dempster*

Nous considérons les deux fonctions des masses m_1 et m_2 suivantes :

Propositions	$\emptyset$	A	B	Ω
m_1	0	0	0.3	0.7
m_2	0	0.4	0	0.6

TABLE IV.7: Exemple 3 : valeurs des fonctions des masses

La première étape consiste à appliquer la règle de fusion conjonctive comme suit :

m_2 / m_1	$\emptyset$ 0	A 0.4	B 0	Ω 0.6
$\emptyset$ 0	$\emptyset$ 0	$\emptyset$ 0	$\emptyset$ 0	$\emptyset$ 0
A 0	$\emptyset$ 0	A 0	$\emptyset$ 0	A 0
B 0.3	$\emptyset$ 0	$\emptyset$ 0.12	B 0	B 0.18
Ω 0.7	$\emptyset$ 0	A 0.28	B 0	Ω 0.42

TABLE IV.8: Exemple 3 : fusion de masses avec l'opérateur conjonctif

Donc le résultat de fusion de masses m_{12}, avec l'opérateur conjonctif, est calculé comme suit :

$m_{12}(\emptyset) = 0.12$

$m_{12}(A) = 0.28$

$m_{12}(B) = 0.18$

$m_{12}(\Omega) = 0.42$

La deuxième étape est la normalisation de la masse conflictuelle $m(\emptyset)$, donc nous appliquons la règle illustrée par l'équation IV.8 :

$m^{\Omega}_{1\oplus 2}(\emptyset) = 0$

$m^{\Omega}_{1\oplus 2}(A) = \frac{0.28}{1-0.12} = 0.3181$

$m^{\Omega}_{1\oplus 2}(B) = \frac{0.18}{1-0.12} = 0.2045$

$m^{\Omega}_{1\oplus 2}(\Omega) = \frac{0.42}{1-0.12} = 0.4772$

IV.3.3.4 Opérateur de Yager

La règle de combinaison de *Yager* considère que les sources d'informations ne sont pas fiables. Pour résoudre le problème de conflit de la règle de *Dempster*, l'opérateur de *Yager* déplace la masse conflictuelle vers Ω :

$$\begin{aligned} m_{12}^{\Omega}(A) &= \sum_{B\cap C=A} m_1^{\Omega}(B)m_2^{\Omega}(C) \qquad A \neq \varnothing \\ m_{12}^{\Omega}(\Omega) &= \sum_{B\cap C=\Omega} m_1^{\Omega}(B)m_2^{\Omega}(C) + K \\ K &= \sum_{B\cap C=\varnothing} m_1^{\Omega}(B)m_2^{\Omega}(C) \end{aligned} \tag{IV.9}$$

Le tableau IV.9 présente la règle de fusion de deux fonctions des masses m_1 et m_2 qui ont un cadre de discernement $\Omega = \{A, B\}$, avec l'opérateur de *Yager*.

m_1 \ m_2	$\emptyset$	A	B	Ω
$\emptyset$	Ω	Ω	Ω	Ω
A	Ω	A	Ω	A
B	Ω	Ω	B	B
Ω	Ω	A	B	Ω

TABLE IV.9: Fusion de masses avec l'opérateur de *Yager* [10]

Exemple 4 : Opérateur de *Yager*

Nous considérons les deux fonctions des masses m_1 et m_2 suivantes :

Propositions	$\emptyset$	A	B	Ω
m_1	0	0	0.3	0.7
m_2	0	0.4	0	0.6

TABLE IV.10: Exemple 4 : valeurs des fonctions des masses

Le tableau IV.11 présente les résultats de fusion :

Donc le résultat de fusion de masses m_{12} est calculé comme suit :

$m_{12}(\emptyset) = 0$

$m_{12}(A) = 0.28$

$m_{12}(B) = 0.18$

$m_{12}(\Omega) = 0.12 + 0.42 = 0.54$

m_2 / m_1	$\emptyset$ 0	A 0.4	B 0	Ω 0.6
$\emptyset$ 0	Ω 0	Ω 0	Ω 0	Ω 0
A 0	Ω 0	A 0	Ω 0	A 0
B 0.3	Ω 0	Ω 0.12	B 0	B 0.18
Ω 0.7	Ω 0	A 0.28	B 0	Ω 0.42

TABLE IV.11: Exemple 4 : fusion de masses avec l'opérateur de Yager

IV.3.3.5 Opérateur Cautious (prudent)

La différence par rapport aux règles précédentes est que cet opérateur combine les informations des sources qui ne sont pas indépendantes. L'algorithme du calcul de cet opérateur est décrit dans [147]. Cette règle est associative, commutative et idempotent [148].

IV.3.4 Raffinement et Grossissement

En pratique lorsque nous fusionnons les masses de croyance, il possible d'avoir des sources de croyance dont leurs cadres de discernement sont différents, mais compatibles. Il est nécessaire donc d'établir des relations entre les cadres discernement, connues sous les noms de raffinement et de grossissement.

Un raffinement associe à chaque hypothèse $\omega \in \Omega$ une ou plusieurs hypothèses $\theta \in \Theta$. L'application $\rho : 2^{\Omega} \longrightarrow 2^{\Theta}$ est un raffinement de Ω si et seulement si ρ vérifie les contraintes suivantes [140] :

(i) L'ensemble $\{\rho(\{\omega_k\}), \omega_k \in \Omega\} \subseteq 2^{\Theta}$ est une partition de Θ.

(ii) $\forall B \subseteq \Omega, \rho(B) = \cup_{\omega_k \in B} \rho(\{\omega_k\})$

La figure IV.2 présente un exemple de raffinement du cadre le moins informatif Ω au cadre le plus informatif Θ.

D'après l'exemple illustré par la figure IV.2, le raffinement de cadre de discernement est le suivant :

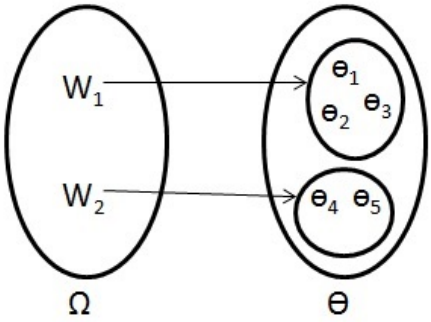

FIGURE IV.2: Raffinement de cadre de discernement : Θ est un raffinement de Ω

1. $m^{\Theta}(\{\theta_1, \theta_2, \theta_3\})=m^{\Omega}(\{\omega_1\})$
2. $m^{\Theta}(\{\theta_4, \theta_5\})=m^{\Omega}(\{\omega_2\})$
3. $m^{\Theta}(\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\})=m^{\Omega}(\{\omega_1, \omega_2\})$

D'autre côté, si Θ est un raffinement de Ω, alors Ω est un grossissement de Θ. Le grossissement est donc l'opération inverse du raffinement et permet de réduire le cadre de discernement en le rendant plus grossier.

IV.3.5 Prise de décision avec la transformation pignistique

La prise de décision dans la théorie des fonctions de croyance est basée sur la notion de probabilité pignistique issue de la transformation pignistique proposée par *Smets* [149]. Le terme "*pignistique*" provient du latin *pignus* qui signifie *un pari*. Cette transformation permet de calculer les probabilités à partir des fonctions de masse dans le but de prendre une décision. La probabilité pignistique reflète la probabilité de chaque hypothèse.

Soit m^{Ω} une BBA définie dans l'espace Ω. La fonction de probabilité pignistique $BetP^{\Omega}$ est définie comme suit [149] :

$$BetP^{\Omega}(w) = \sum_{W \subseteq \Omega, w \in W} \frac{1}{|W|} \frac{m^{\Omega}(W)}{(1 - m^{\Omega}(\oslash))}, \forall w \in \Omega \qquad \text{(IV.10)}$$

où $|W|$ est le nombre d'éléments de Ω dans W.
Cette transformation est utile pour transformer les masses de croyance en probabilités afin de prendre une décision.

Exemple : Prise de décision avec la transformation pignistique

Nous prenons le cas présenté dans l'exemple 4, celui de l'opérateur de *Yager*. Le but est de calculer les probabilités pignistiques P_A et P_B avec un cadre de discernement $\Omega = \{A, B\}$. En appliquant la règle illustrée par l'équation IV.10, nous obtenons :

$$\begin{aligned} P_A &= BetP^{\Omega}(A) = m_{12}(A) + \frac{m_{12}(\Omega)}{2} = 0.55 \\ P_B &= BetP^{\Omega}(B) = m_{12}(B) + \frac{m_{12}(\Omega)}{2} = 0.45 \end{aligned} \tag{IV.11}$$

IV.3.6 Fusion bayésienne de probabilités pignistiques

Pour combiner les probabilités pignistiques provenant de différentes sources, nous utilisons l'approche bayésienne. Pour deux probabilités pignistiques, la fusion bayésienne est définie comme suit [150] :

$$P_{Fusion(ij)} = \frac{BetP\{P_i\}BetP\{P_j\}}{\sum_{i,j=1}^{n} BetP\{P_i\}BetP\{P_j\}} \tag{IV.12}$$

où $P_{Fusion(ij)}$ est la probabilité fusionnée, P_i et P_j sont les probabilités pignistiques fournies respectivement par les sources de croyance i et j.

IV.4 Architecture proposée pour la fusion multi-capteurs : caméra et radar

Dans cette section, nous présentons les outils nécessaires pour notre architecture de fusion multi-capteurs. Ensuite, nous détaillons l'approche proposée en utilisant la théorie de croyance.

IV.4.1 Mise en place des capteurs caméra et radar

La figure IV.3 présente les champs des visions des capteurs caméra et radar. Le radar est configuré en mode *track* (voir section III.2.1.4), permet de fournir une liste d'obstacles détectés, chaque obstacle possède une identité unique ID qui lui sera attribuée, au cours du temps, pour être suivi. De plus, le radar fournit d'autres

informations sur les obstacles détectés telles que leurs positions, vitesses relatives et leurs paramètres RCS. Notons que la précision d'azimut de radar varie en fonction de son champ de vision. Par exemple, la précision d'azimut de radar est d'environ 2° pour un champ de vision de ±20°.

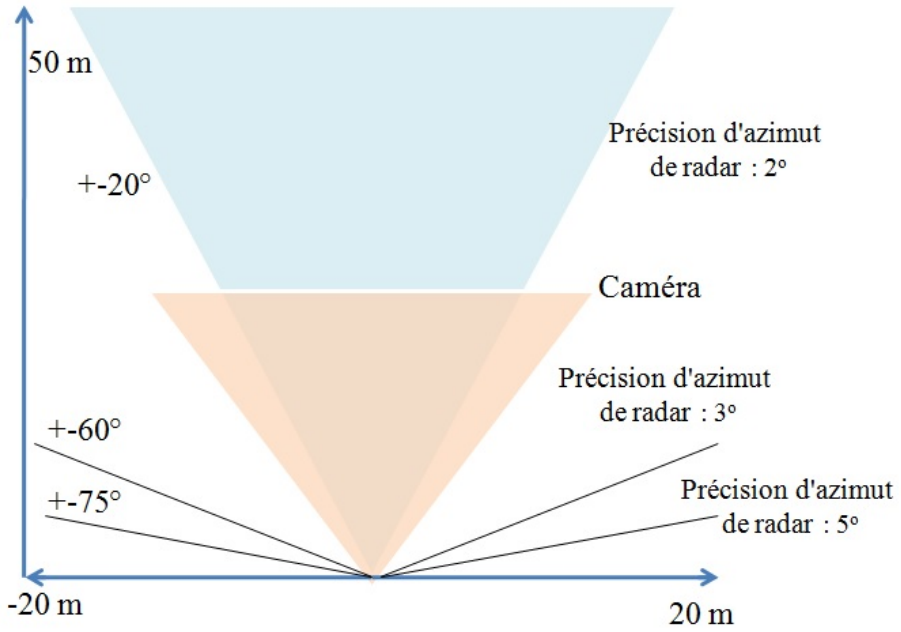

FIGURE IV.3: Champs des visions de caméra et radar

IV.4.2 Paramètre RCS

La surface équivalente radar (RCS : Radar Cross Section) caractérise le degré de réflectivité d'une cible soumise à un champ électromagnétique [151]. La valeur de RCS décrit la quantité d'énergie réfléchie par une cible donnée. Chaque cible telle que piéton, vélo, voiture ou camion a une valeur RCS. Il y a plusieurs facteurs qui influencent ce paramètre tels que le matériau et la taille de la cible, l'angle de réflexion, la fréquence de radar, etc. Pour le radar 24 GHz utilisé à courte portée et selon le document technique de fabrication, la valeur RCS du piéton est autour de $-15\ dBm^2$. La figure IV.4 montre la variation des valeurs RCS pour une détection d'un piéton mobile en fonction du temps. Empiriquement, cette valeur est négative et autour de $-15\ dBm^2$ pour un piéton. Nous utilisons cette observation comme une information pour notre approche de fusion.

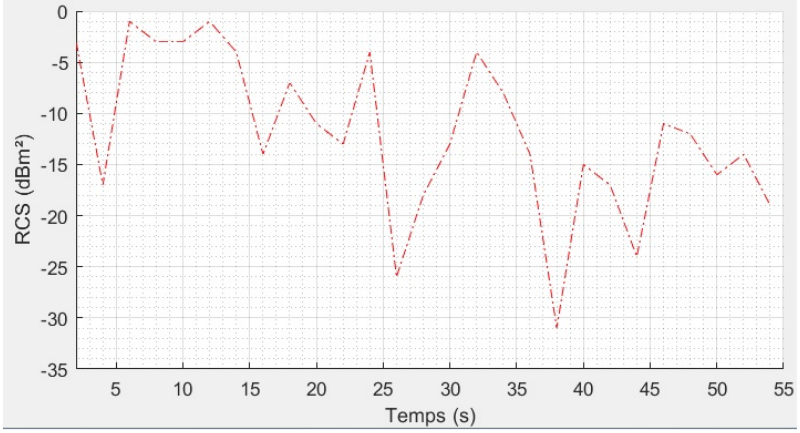

FIGURE IV.4: Variation des valeurs de *RCS* d'un même piéton mobile détecté par un radar 24 GHz à courte portée

IV.4.3 Détecteur piétons : Aggregate Channel Features

La détection d'obstacle, et en particulier celle des piétons, est une tâche délicate, d'un point de vue vision artificielle. Le manque des modèles explicites conduit à l'utilisation des techniques d'apprentissage automatique, où une représentation implicite apprend en se basant sur l'analyse des données empiriques provenant d'une base de données [152]. Au cours de ces dernières années, le nombre d'approches pour détecter les personnes ou les piétons (applications automobiles) dans les images monoculaires a augmenté régulièrement. Les problématiques actuelles restent la précision (éviter les fausses détections) et le temps nécessaire pour analyser et traiter les informations.

Parmi ces approches, nous trouvons l'algorithme Aggregate Channel Features *(ACF)*. Il s'agit d'un détecteur des personnes rapide basé sur la notion des canaux des descripteurs (Channel Features) et qui a montré d'excellentes performances sur plusieurs bases de données publiques, pour un faible temps de calcul [25]. Notons qu'un canal est une représentation d'une image dans laquelle les pixels sont obtenus en appliquant une fonction de génération de descripteurs. *ACF* utilise dix canaux de descripteurs qui sont la valeur du gradient, Histogram of Oriented Gradients *(HOG)* (sur six canaux), et les canaux couleur LUV [153]. Ces descripteurs *ACF* sont performants en termes de détection et coût du traitement **[25]**. Cet algorithme est entraîné sur le jeu de données *INRIA*. Un exemple de détection effectué par l'algorithme *ACF* est présenté dans la figure IV.5. Notons que la sortie de ce module

est les coordonnées, en pixels, des cadres entourant les objets détectés dans l'image (Bounding Boxes) ainsi que leurs scores des détections (confiances des détections).

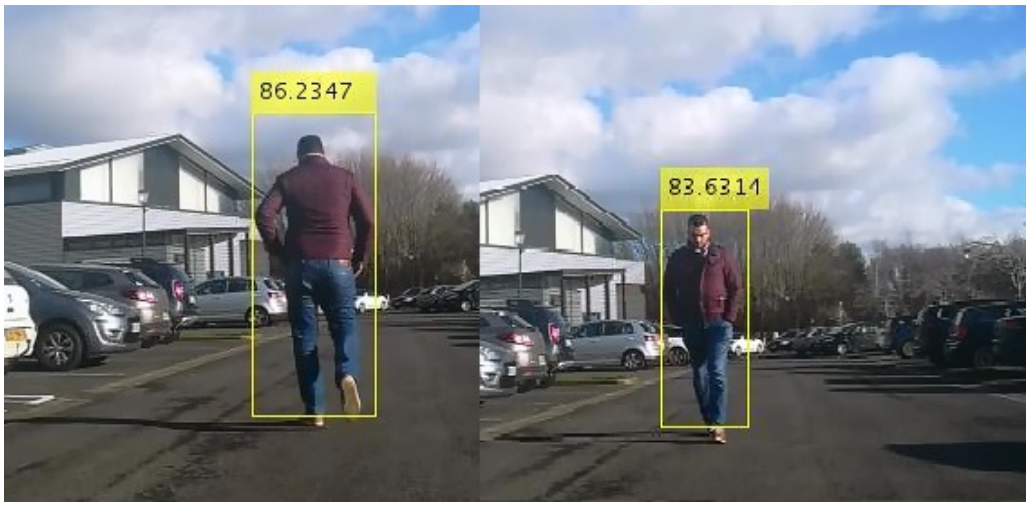

FIGURE IV.5: Détections de piéton avec le détecteur *ACF*

IV.4.4 Approche de fusion proposée

La figure IV.6 décrit l'approche proposée pour la fusion de données entre la caméra et le radar. Chaque objet détecté est représenté par un couple (Position, Indicateur de classe). La position est fournie par le radar. L'indicateur de classe spécifie la classe de détection qui peut être piéton ou véhicule (moto, voiture, camion).

Nous pouvons distinguer deux cas, si la vitesse de l'obstacle détecté est supérieure à un seuil fixé, il s'agit de la classe véhicule. Si non, nous classifions l'obstacle détecté suivant sa valeur de RCS et la classification effectuée par l'algorithme ACF.

En fait, si la valeur de RCS de l'obstacle est autour de $-15\ dBm^2$ (pour le radar utilisé), la classe est probablement un piéton. Afin d'améliorer la précision de détection, nous utilisons aussi la caméra pour détecter les piétons avec l'algorithme ACF. L'utilisation de caméra permet d'avoir une autre source pour confirmer la classe de l'objet et assurer une continuité de détection puisque le radar est sensible dans un environnement encombré.

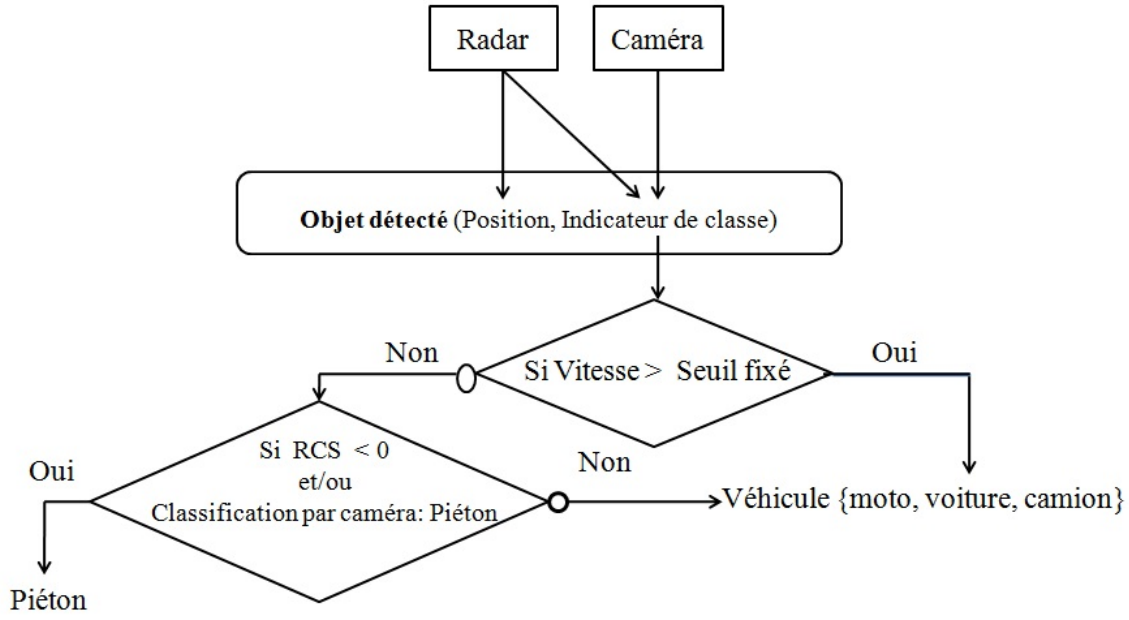

FIGURE IV.6: Approche de fusion proposée : caméra/radar

IV.4.5 Module de fusion

La figure IV.7 présente l'architecture du module de fusion proposée, le cas où la vitesse de l'obstacle détecté est inférieure au seuil fixé. Les entrées de ce module sont des listes d'objets détectés fournies par les capteurs caméra et radar. Nous détaillons chaque élément de ce module dans la suite.

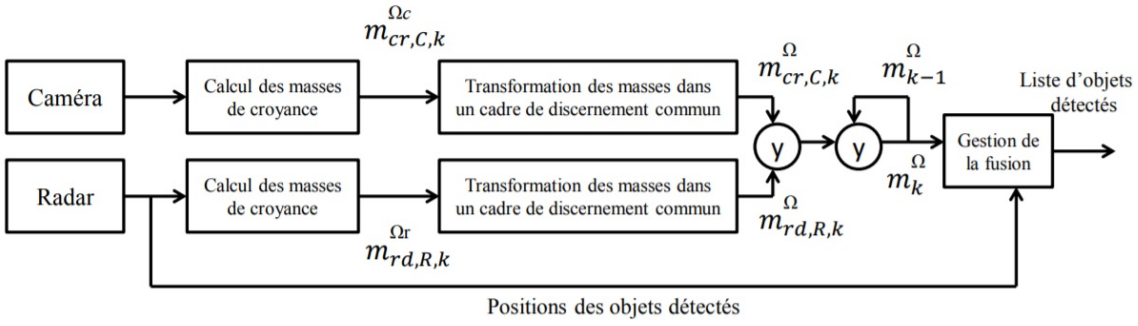

FIGURE IV.7: Architecture du module de fusion

IV.4.5.1 Cadres de discernement

La première étape consiste à définir les cadres de discernement pour chaque capteur. La caméra détecte les piétons avec l'algorithme ACF. Supposons que $\Omega_c = \{P_{cr}, NP_{cr}\}$ est le cadre de discernement de détection des piétons par la caméra, où P_{cr} représente l'hypothèse de détection des piétons et NP_{cr} est pour l'hypothèse inverse (Non-Piéton).

Quant aux objets détectés par le radar, ils peuvent être : piéton P_{dr} (selon sa valeur RCS et sa vitesse relative), ou véhicule NP_{dr} (Moto, voiture, camion), ou fausse alarme FA_{dr}. Donc, $\Omega_r = \{P_{dr}, NP_{dr}, FA_{dr}\}$ est le cadre de discernement de détection par le capteur radar.

IV.4.5.2 Calcul des masses de croyance : BBAs

La deuxième étape consiste à calculer les $BBAs$ à partir des mesures des capteurs. Soient $m^{\Omega_c}_{cr,C,k}$ et $m^{\Omega_r}_{rd,R,k}$ les masses de croyance fournies respectivement par la caméra et le radar à l'instant k. Pour la caméra, la transformée inverse de probabilité pignistique est appliquée [77, 154] aux probabilités générées par l'algorithme ACF. Cette étape est détaillée dans l'annexe C.

Pour le processus de reconnaissance et détection effectué par le radar, les $BBAs$ sont définies par des facteurs de confiance. Si la vitesse de l'obstacle détecté est inférieure au seuil fixé, nous distinguons deux cas :

(i) L'obstacle détecté est un piéton si sa valeur de RCS est autour de $-15\ dBm^2$.

(ii) Si non l'obstacle détecté est un véhicule.

IV.4.5.3 Transformation des BBAs dans un cadre de discernement commun

Afin de combiner les masses de croyance, les différentes $BBAs$ devraient être transformés dans un cadre de discernement commun :

(i) Le cadre de discernement de détection du piéton par la caméra est $\Omega_c = \{P_{cr}, NP_{cr}\}$. L'hypothèse NP_{cr} inclut les sous-hypothèses de Non-Piéton et de fausse alarme.

(ii) Le cadre de discernement du processus de reconnaissance et de détection effectué par le radar est $\Omega_r = \{P_{dr}, NP_{dr}, FA_{dr}\}$.

Selon les définitions précédentes, le cadre de discernement commun des $BBAs$ est $\Omega = \{P, NP, FA\}$. Pour transformer les $BBAs$ fournies par la caméra dans le cadre de discernement commun, nous utilisons l'opération du raffinement décrit dans la section IV.3.4 , donc nous trouvons :

(i) $m_k^{\Omega}(\{P\}) = m_{cr,C,k}^{\Omega_c}(\{P_{cr}\})$

(ii) $m_k^{\Omega}(\{NP, FA\}) = m_{cr,C,k}^{\Omega_c}(\{NP_{cr}\})$

(iii) $m_k^{\Omega}(\{\Omega\}) = m_{cr,C,k}^{\Omega_c}(\Omega_c)$

Notons que le cadre de discernement du processus de reconnaissance et de détection réalisé par le radar Ω_r est le même que celui du cadre commun Ω.

IV.4.5.4 Fusion et mise à jour des BBAs

Puisque les sources des croyances, la caméra et le radar, sont indépendantes et non-fiables, nous appliquons la règle de combinaison de *Yager* Ⓨ afin de combiner les différentes $BBAs$.
D'autre coté, pour mettre à jour une BBA donnée entre deux instants, nous devons combiner les valeurs de cette BBA aux instants $(k-1)$ et (k) en utilisant la règle de combinaison appropriée. D'abord, nous supposons l'indépendance temporelle entre les deux instants $(k-1)$ et (k), ce qui signifie l'indépendance entre les deux sources d'information. Nous supposons également que les sources d'information ne sont pas fiables. Sur la base de ces hypothèses, nous choisissons la règle de combinaison de *Yager* pour mettre à jour les $BBAs$: m_k^{Ω} et m_{k-1}^{Ω}.

IV.4.5.5 Génération et fusion des probabilités pignistiques

Pour transformer les masses de croyance en probabilités afin de prendre une décision, nous utilisons la fonction de probabilité pignistique [154] présentée dans la section IV.3.5.

Tout d'abord, les probabilités pignistiques $BetP$ devront être calculées dans les cadres de discernement originaux de caméra Ω_c et de radar Ω_r, donc nous devons

faire le processus inverse de raffinement, le grossissement. Soient m^{Ω_c} et m^{Ω_r} les masses de croyance à l'issue du processus de grossissement pour la caméra et le radar. Pour la tâche de reconnaissance de la caméra, le grossissement est déterminé comme suit :

$$\begin{aligned} m^{\Omega_c}(\{P_{cr}\}) &= m^{\Omega}(\{P\}) \\ m^{\Omega_c}(\{NP_{cr}\}) &= m^{\Omega}(\{NP\}) + m^{\Omega}(\{FA\}) + m^{\Omega}(\{NP, FA\}) \\ m^{\Omega_c}(\Omega_c) &= m^{\Omega}(\{P, NP\}) + m^{\Omega}(\{P, FA\}) + m^{\Omega}(\{\Omega\}) \end{aligned} \tag{IV.13}$$

Le processus inverse de raffinement de radar reste le même car le cadre de discernement commun Ω est le même que celui de radar Ω_r. Par conséquent, les probabilités pignistiques de détection et de reconnaissance pour la caméra et le radar sont respectivement :

$$\begin{aligned} BetP_c(P_c) &= m^{\Omega_c}(\{P_{cr}\}) + \frac{m^{\Omega_c}(\Omega_c)}{2} \\ BetP_r(P_r) &= m^{\Omega_r}(\{P_{dr}\}) + \frac{m^{\Omega_r}(\{P_{dr}, NP_{dr}\})}{2} + \frac{m^{\Omega_r}(\{P_{dr}, FA_{dr}\})}{2} + \frac{m^{\Omega_r}(\Omega_{dr})}{3} \end{aligned} \tag{IV.14}$$

La dernière étape consiste à combiner les probabilités pignistiques, pour ce faire, nous utilisons la fusion bayésienne définie par l'équation IV.12.

IV.5 Résultats

Nous utilisons la simulation pour analyser le comportement du module de fusion et explorer les situations possibles qui peuvent apparaître dans les tests sur le terrain comme montre la figure IV.8.

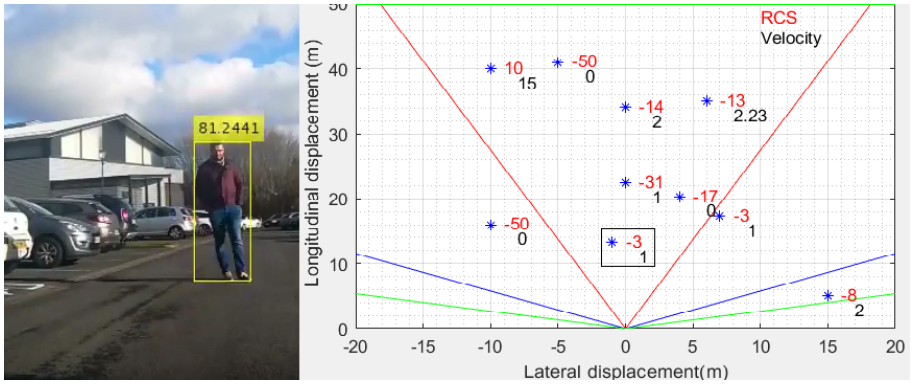

FIGURE IV.8: Reconnaissance d'un piéton à l'aide des capteurs radar et caméra

IV.5.1 Études des cas pour la détection et reconnaissance d'obstacles

Nous étudions dans cette section le comportement du module de fusion. Pour explorer les situations possibles qui peuvent apparaître dans un environnement réel, quatre cas sont considérés :

(i) Bonne détection par les deux capteurs, caméra et radar.

(ii) Les deux capteurs ne détectent pas le piéton.

(iii) Le piéton est détecté seulement par le radar.

(iv) Le piéton est détecté seulement par la caméra.

Chaque figure de IV.9 à IV.12 présente cinq courbes :

(i) Les entrées qui sont : la probabilité de détection de piéton par la caméra, la BBA de détection de piéton par la caméra (probabilité pignistique inverse) et la BBA de détection de piéton par le radar.

(ii) Les sorties qui sont les probabilités pignistiques de détection de piéton par la caméra et le radar (décisions).

(1) Bonne détection par les deux capteurs : La figure IV.9 présente les résultats de la simulation pour la reconnaissance et la détection d'un piéton. Dans ce cas, la probabilité de détection de la caméra est supérieure à 0.5 et la BBA de la détection de radar est 0.8. Par conséquent, les deux probabilités pignistiques atteignent la valeur 1 (les étoiles vertes et la ligne en pointillé noire) ce qui représente

la confiance totale que la détection est un piéton.

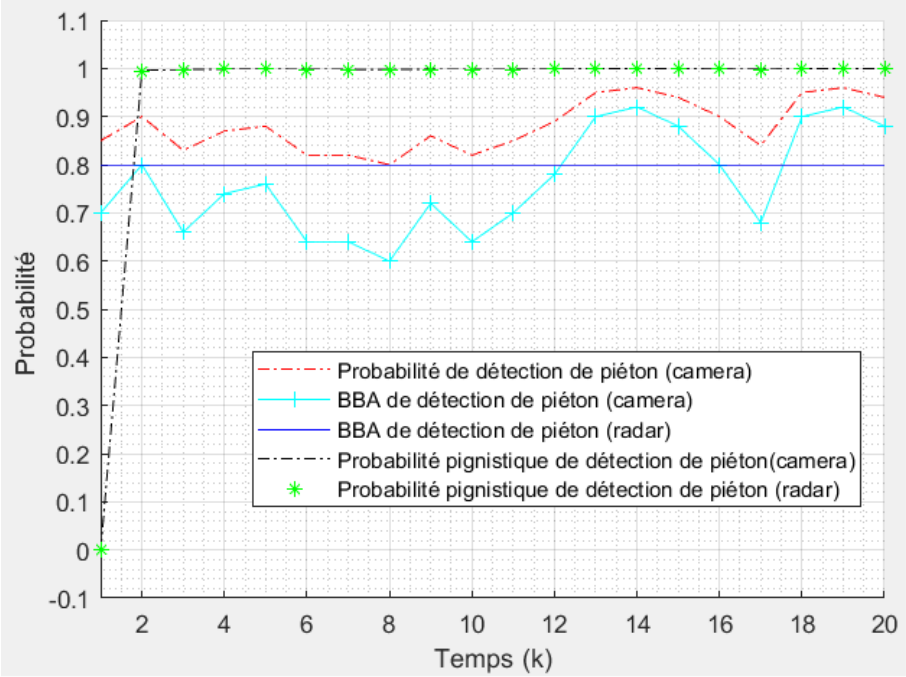

FIGURE IV.9: Simulation pour la détection et la reconnaissance d'un piéton : Détection par les deux capteurs

(2) Les deux capteurs ne détectent pas le piéton : Contrairement au premier cas, la figure IV.10 montre les résultats où la probabilité de détection de la caméra est inférieure à 0.5 et la BBA de détection par le radar est nulle, par conséquent, les probabilités pignistiques sont nulles.

(3) Le piéton est détecté seulement par le radar : La figure IV.11 présente les résultats où seul le radar détecte le piéton et la probabilité de reconnaissance par la caméra est inférieure à 0,5. Selon ces résultats, les probabilités pignistiques suivent la source qui fournit la BBA qui est supérieure à 0.5, le radar dans ce cas. La situation réelle qui correspond à cette simulation est le cas où la caméra ne fonctionne pas correctement, par exemple dans des conditions de brouillard ou de pluie.

(4) Le piéton est détecté seulement par la caméra : La figure IV.12 présente les résultats où seule la caméra détecte le piéton (probabilité supérieure à

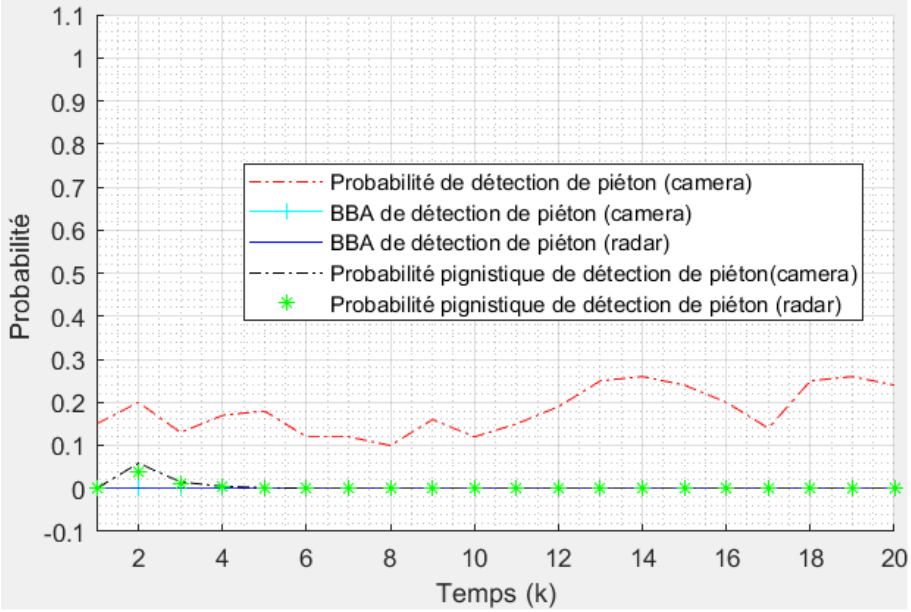

FIGURE IV.10: Simulation pour la détection et la reconnaissance d'un piéton : les deux capteurs ne détectent pas le piéton

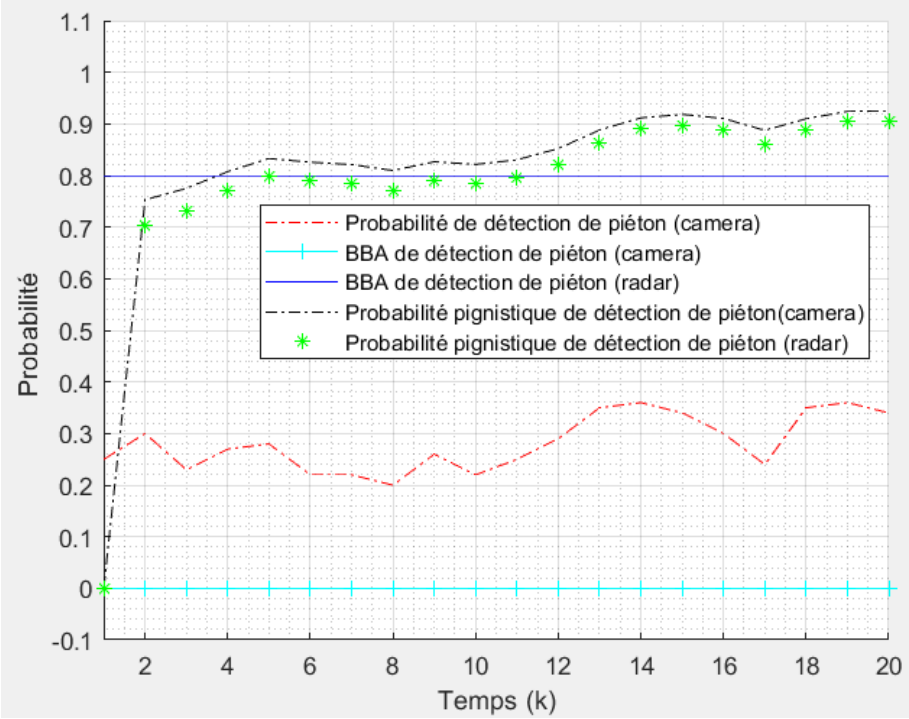

FIGURE IV.11: Le piéton est détecté seulement par le radar

0.5). Comme dans le cas précédent, les probabilités pignistiques suivent la source qui fournit la BBA la plus grande (caméra). La situation réelle qui correspond à

cette simulation est un mauvais contexte d'acquisition par le radar dans un environnement encombré.

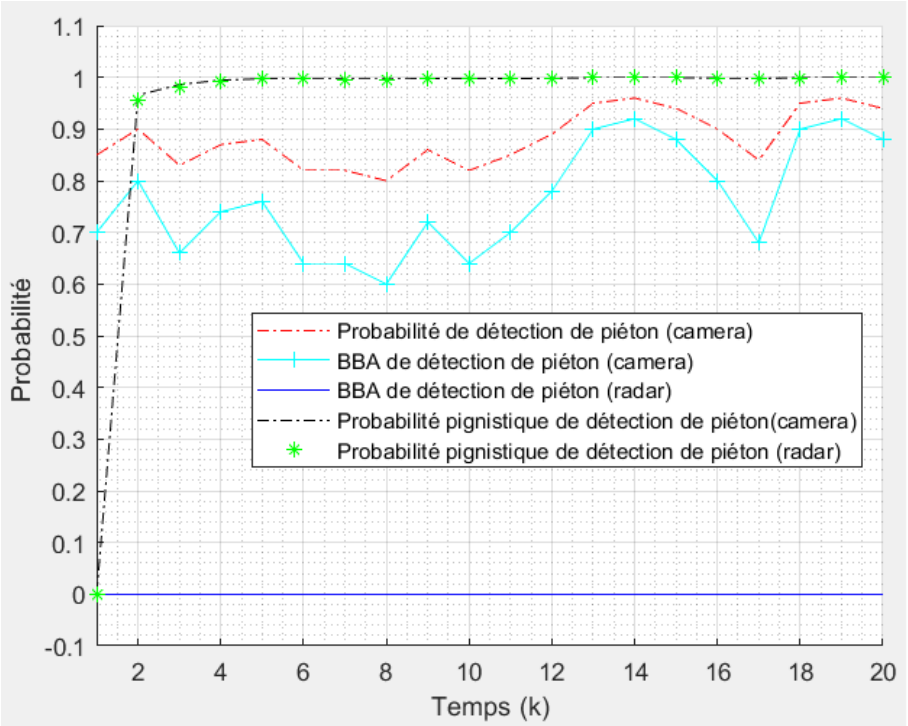

FIGURE IV.12: Le piéton est détecté seulement par la caméra

IV.5.2 Fusion des probabilités pignistiques

Afin de fusionner les deux probabilités pignistiques fournies par la camera et le radar, nous appliquons la fusion bayésienne (voir section IV.3.6). Nous prenons le cas de bonne détection par les deux capteurs pour fusionner ces probabilités. La figure IV.13 présente le résultats obtenu. Selon ces résultat, la probabilité fusionnée de détection de piéton est élevée.

Également, nous présentons le résultat de fusion bayésienne pour le cas où le piéton est détecté seulement par le radar dans la figure IV.14.

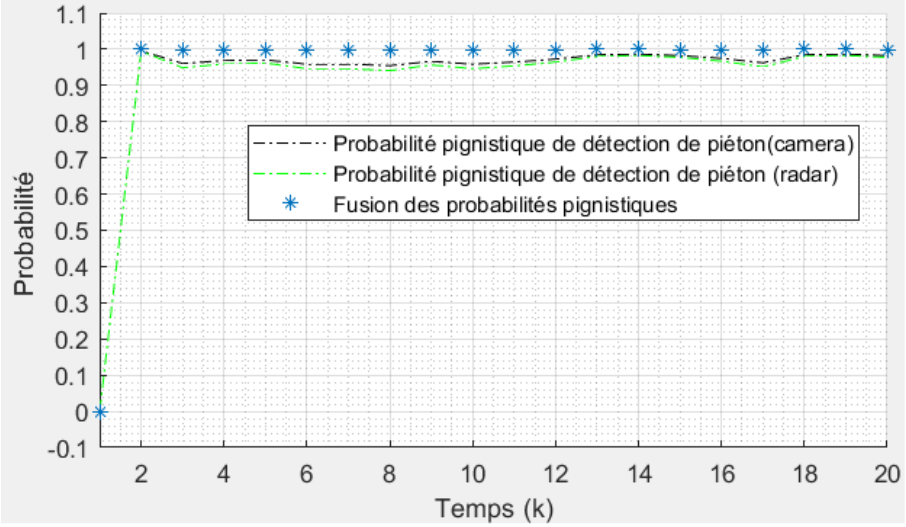

FIGURE IV.13: Fusion des probabilités pignistiques : cas de bonne détection par les deux capteurs

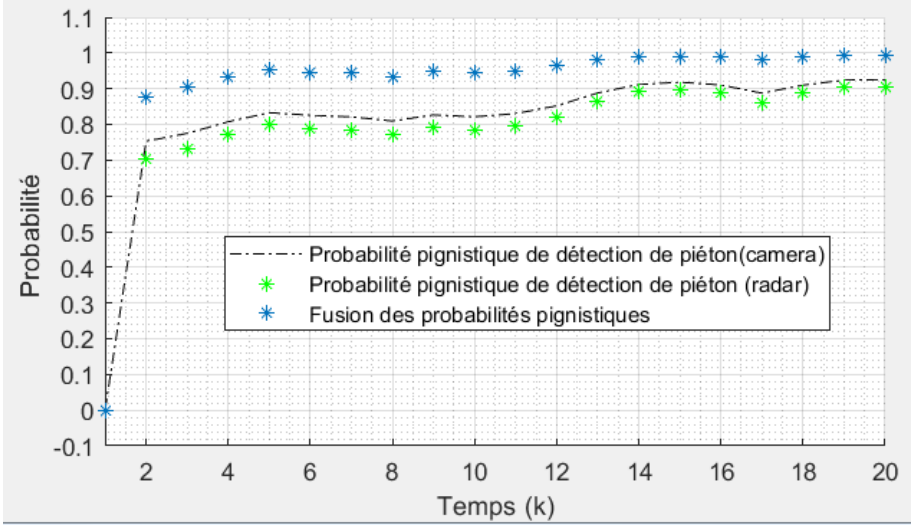

FIGURE IV.14: Fusion des probabilités pignistiques : cas où le piéton est détecté seulement par le radar

IV.5.3 Discussion

Les résultats obtenus montrent que le système conçu fonctionne comme prévu. Le module de fusion permet d'assurer la continuité de la détection et la reconnaissance de l'obstacle. En fait, en absence de détection de piéton par un capteur, le deuxième continue à effectuer cette tâche. Les fonctions de croyance permettent de redistribuer les probabilités fournies par les sources dans le but de prendre une décision plus sûre.

Par exemple, nous avons vu dans le cas où le piéton est détecté seulement par le radar, que la probabilité de détection de caméra, en entrée, est inférieure à 0.5, mais sa probabilité pignistique, en sortie, a été augmentée en s'appuyant sur la fusion des masses de croyance de deux capteurs, ce qui a permis de redistribuer les probabilités.

En comparant l'architecture de notre module de fusion par rapport à celle décrite dans [77], *Fayad* et al. ont utilisé une caméra stéréoscopique pour la reconnaissance et la détection des piétons. Ils ont utilisé une caméra stéréoscopique pour la détection et une caméra monoculaire (celle de gauche de stéréoscopique) pour la reconnaissance des pétons. Puisqu'il s'agit de la même source d'information, ils ont utilisé l'opérateur prudent de *Denœux* [148]. Dans notre cas, nous avons utilisé l'opérateur de *Yager* car il s'agit de deux sources d'informations différentes qui sont la caméra et le radar.

IV.5.4 Conclusion

Dans ce chapitre, nous avons présenté notre approche de détection et classification d'obstacles en utilisant la théorie de croyance. Puisqu'un véhicule autonome se situe souvent dans un environnement incertain, nous avons choisi le cadre *TBM* et ses outils pour modéliser et gérer les incertitudes. Nous avons proposé une architecture multi-capteurs qui s'appuie sur la fusion des preuves et des évidences provenant de caméra et de radar afin d'assurer la continuité de la détection et la reconnaissance d'un obstacle donné. Les résultats de simulation et les tests expérimentaux préliminaires ont montré le bon fonctionnement de l'approche proposé.
Dans le prochain chapitre, nous abordons les contraintes et les exigences de la conception d'une plateforme embarquée dédiée aux applications automobiles intelligentes en proposant le FPGA comme une solution candidate pour accueillir ces applications.

Chapitre V

Exploration de l'espace de conception d'une plateforme embarquée de fusion multi-capteurs

Sommaire

V.1 Introduction

La motivation principale de notre travail est la mise en œuvre d'une plateforme embarquée de fusion multi-capteurs pour la détection d'obstacles dans le but d'améliorer la précision de perception du véhicule et garantir une sécurité routière aux différents usagers. Cependant, face à l'accroissement de la complexité de calcul et des volumes des données à traiter à l'issue des capteurs, un traitement haute performance est indispensable. Parmi les technologies d'accélération du traitement, nous choisissons le FPGA permettant d'offrir une puissance de calcul important, par le moyen de ses ressources reconfigurables, avec une faible consommation d'énergie électrique tout en garantissant un coût maitrisé. De plus, pour palier aux problèmes qui sont liés à la méthodologie de conception conventionnelle basée sur le codage manuel, nous utilisons les outils de synthèse haut niveau. Ces derniers permettent aux concepteurs de tester rapidement les différents algorithmes et leurs alternatives d'implémentation dans un cycle de conception rapide.

Cet ensemble d'objectifs nous amène à explorer l'espace de conception d'une plateforme embarquée de fusion multi-capteurs dans ce chapitre. Dans la section V.2, nous abordons les défis scientifiques et industriels de conception embarquée dans le contexte du véhicule autonome. Dans la section V.3, nous présentons la plateforme cible, le *Zynq ZC*706 et les outils d'exploration utilisés. Enfin dans la section V.4, nous détaillons les différentes étapes d'exploration de l'espace de conception de la plateforme embarquée en présentant les résultats expérimentaux obtenus.

V.2 Défis scientifiques et industriels

La conception et le développement des plateformes embarquées posent des défis technologiques et scientifiques, notamment : le développement des algorithmes complexes, la vérification et le test des différentes contraintes fonctionnelles et non-fonctionnelles, la nécessité d'automatiser le processus de conception pour augmenter la productivité, la conception d'une architecture matérielle adéquate pour exploiter le parallélisme et pour satisfaire la contrainte temps-réel, réduire la puissance

consommée pour prolonger la durée de fonctionnement avant de recharger le véhicule, etc. De plus, lorsque des milliers des lignes de code seront embarquées, il faudra procéder à l'optimisation des architectures matérielles et logicielles, et sans doute concevoir des architectures matérielles dédiées.

Comme mentionné précédemment dans le chapitre II, les technologies FPGA sont adéquates pour répondre aux exigences des systèmes embarqués grâce à leurs caractéristiques qui peuvent être résumées par le calcul à haute performance à des fréquences de fonctionnement plus basses, la reconfigurabilité et leur capacité à réaliser des architectures massivement parallèles. Outre que le rôle de calcul, un FPGA peut être utilisé comme une plateforme de communication avec l'existence des ports de communication standards sur la carte tels que le bus CAN, l'interface Ethernet et l'interface FPGA Mezzanine Card (FMC). Par ailleurs, pour palier aux problèmes qui sont liés à la méthodologie de conception conventionnelle basée sur le codage manuel et afin d'accélérer le cycle du développement, nous utilisons les outils HLS. La figure V.1 présente la navette *ARMA* avec les emplacements des capteurs, la caméra stéréo et le LIDAR. Afin de générer une liste d'objets détectés par la caméra stéréo, l'implémentation *SW* de la mise en correspondance et la reconstruction stéréo n'est pas performante car le temps du traitement est de 4.7 seconde pour deux images (gauche et droite) avec une résolution *VGA* 640×480 tandis que le LIDAR fournit une liste d'objets chaque 0.04 seconde. De ce fait, l'implémentation *SW* ne répond pas à la contrainte temporelle du traitement temps réel, c'est qui nous a incité à faire appel à l'implémentation *HW* et de profiter des avantages des FPGA.

V.3 Plateforme cible et outils d'exploration

Dans cette section, nous présentons la plateforme cible, le *Zynq ZC*706 et les outils d'exploration utilisés.

La nouvelle génération des systèmes sur puce soutient les architectures hétérogènes, dynamiques avec un modèle d'exécution parallèle. Ces architectures rassemblent des processeurs multi-cores avec une partie programmable, le FPGA. Par conséquent, les concepteurs ont plusieurs choix pour leurs implémentations telles que les implé-

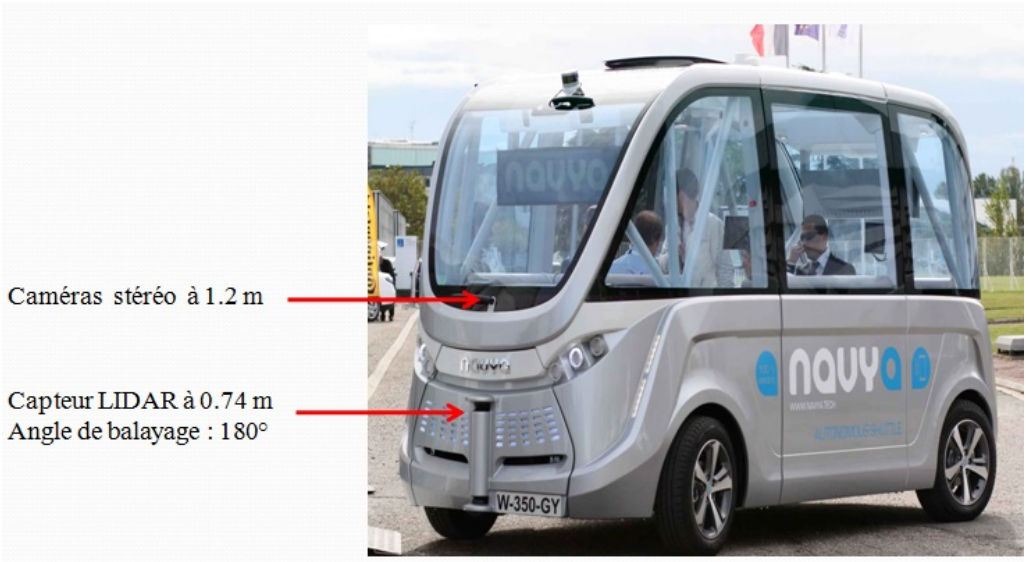

FIGURE V.1: La navette ARMA avec les emplacements des capteurs

mentations SW ou HW, séquentielle ou parallèle.

L'impact de ces architectures dans le domaine des systèmes embarqués va ainsi permettre d'utiliser la logique programmable pour gérer des interfaces spécifiques avec le monde des Entrées/Sorties, ou bien réaliser des opérations massivement parallèles. Quant à la partie processeur, elle permet de contrôler les mémoires, gérer le système d'exploitation et fournir un support de logiciels applicatifs.

Parmi ces solutions proposées dans le marché, nous utilisons dans nos travaux la carte de développement $Zynq\ ZC706$, conçue par $Xilinx$, intégrant principalement le système sur puce $Zynq-7000\ SoC$ qui repose sur la combinaison d'un processeur multi-core $ARM\ Cortex-A9$ gravé en dur (Hard) et une partie programmable, le FPGA.

V.3.1 Le système sur puce Zynq-7000

La figure V.2 présente l'architecture matérielle de SOC $Zynq-7000$ inclut :

– Processing System (PS) : intègre principalement deux cores $ARM\ Cortex-A9$ avec une fréquence maximale de 667 MHz, deux unités Floating Point Unit (FPU), deux unités Memory Mangement Unit (MMU) et d'autres éléments de calcul et mémoire.

– Programmable Logic (PL) : c'est la partie programmable de *SOC* cadencée à une fréquence maximale de 200 Mhz.

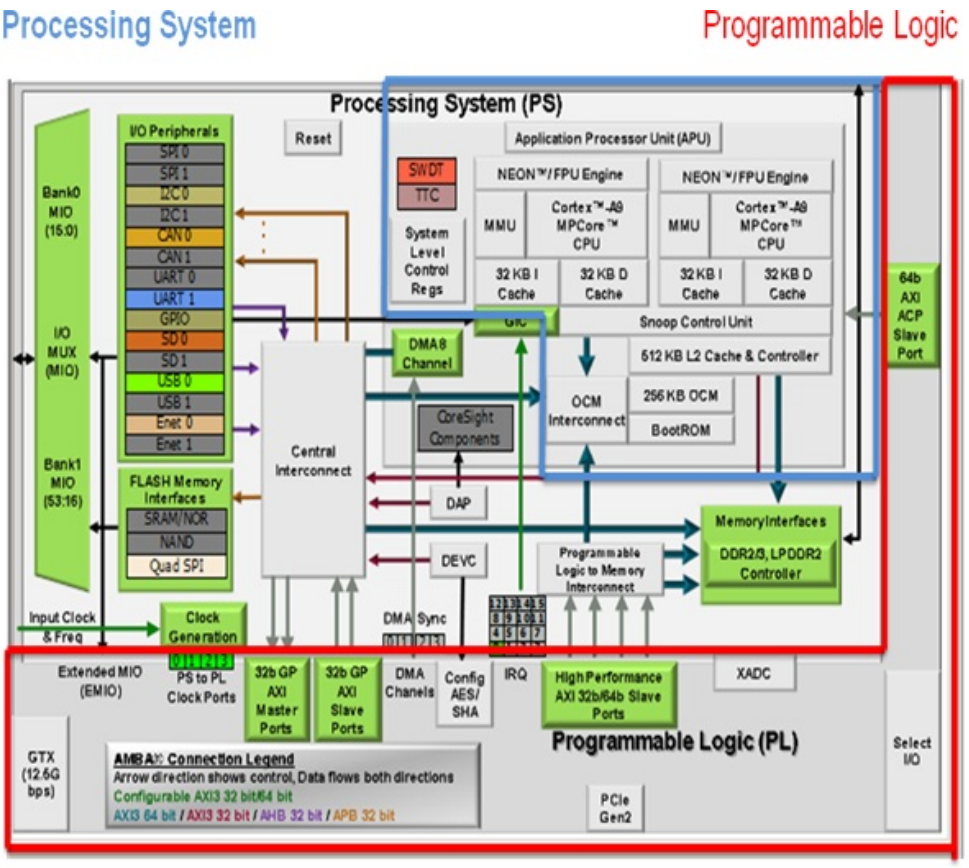

FIGURE V.2: Le système sur puce : *Zynq-7000* [7]

V.3.2 Interfaces entre PS et PL

La communication entre les parties *PS* et *PL*, au sein de *SOC Zynq* utilise le bus *AMBA* [155]. Ce dernier, fournit trois modes de communication qui sont : AXI4 Memory MAP, AXI4 Stream et AXI Lite. Chaque mode a ses propres caractéristiques et par la suite son utilisation précise. Pour résumer, le SOC $Zynq - 7000$ fournit 9 moyens de communication entre le *PS* et le *PL* qui sont :

– Deux ports Master General-Purpose (GP) 32 bits : permettent de transférer les données entre les deux parties dont le *PS* est le maître de communication, c'est-à-dire que c'est le *PS* qui déclenche et contrôle le traitement.

– Deux Ports Slave GP 32 bits : permettent de transférer les données entre les

deux parties dont le PL est le maître, c'est-à-dire que c'est le PL qui déclenche et contrôle le traitement.

– Quatre ports High Performance (HP) 32/64 bits : ces ports permettent au FPGA d'accéder directement au DDR (le PL est le maître).
– Un Port ACP 64 bits : ce port permet à la partie PL d'accéder aux mémoires caches.

V.3.3 Flot de conception de Matlab/Simulink

Matlab/Simulink offre principalement deux outils pour les implémentations embarquées. Le premier est $HDL\ Coder$, qui génère automatiquement du code HDL à partir de $MATLAB$, permettant ainsi aux utilisateurs d'implémenter les designs FPGA ou ASIC à partir du langage $MATLAB$ couramment utilisé. Quant au deuxième outil, il s'agit de *Embedded Coder* qui génère un code *C/C++* pour être utilisé sur des processeurs embarqués, des cartes de prototypage rapide sur cible ou des microprocesseurs utilisés dans la production en grande série. Ces outils permettent d'obtenir des codes *C/C++* ou *HDL* pour la même application afin d'être exécutée sur un processeur ou sur un FPGA, ce qui nous donne plus de flexibilité au niveau du choix des implémentations. La figure V.3 présente le flot de conception de Matlab/Simulink (approche Model-Based Design). Les concepteurs et les développeurs des algorithmes utilisent Matlab/Simulink dans le but de créer des modèles pour un système donné à partir des spécifications précises. Également, cet outil, inclut des bibliothèques prêt à utiliser, permet de simuler les algorithmes afin d'évaluer les concepts, les compromis et les algorithmes de partitionnement *HW/SW*. La tâche de partitionnement spécifie la partie d'algorithme qui sera exécutée par le processeur *ARM* en *SW*, et celle qui sera exécutée par le FPGA en *HW*. Les sorties de ces outils sont un fichier exécutable pour le processeur *ARM* et un *bitstream* pour le FPGA. À noter qu'il y a d'autres étapes de vérification entre ces tâches afin de produire une application qui satisfait les spécifications du concepteur. Notons que ces outils respectent la norme ISO : 2626.2 celle des applications automobiles.

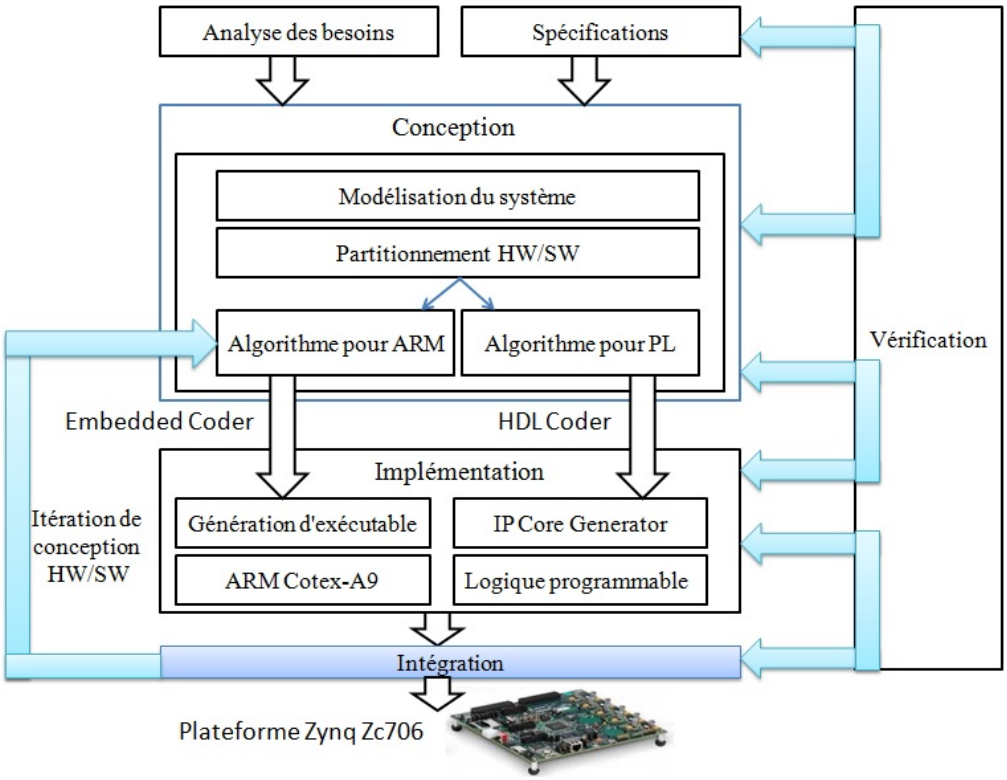

FIGURE V.3: Le flot de conception de Matlab/Simulink

V.3.4 Mesure de consommation électrique de Zynq-7000

Le SOC $Zynq-7000$ inclut une unité Power Mangement Unit (PMU) permettant de modéliser et de mesurer les signaux analogiques tels que la tension, le courant et la température. Ces grandeurs sont mesurées avec l'adaptateur Power Management Bus (PMB) de Texas Instruments [156]. L'unité PMU permet de mesurer la tension, le courant et la puissance d'entrée ainsi que ceux de sortie du SOC et d'autres grandeurs. Les figures V.4, V.5 et V.6 représentent respectivement :

- L'allure de puissance de sortie de PS au repos (sous tension mais sans activité) qui a une valeur moyenne de 0.14 W.
- L'allure de puissance de sortie de PS lors d'exécution de Linux embarqué (distribution Snapgear) qui a une valeur moyenne de 0.33 W.
- L'allure de puissance de sortie de PL au repos qui a une valeur moyenne de 0.06 W.

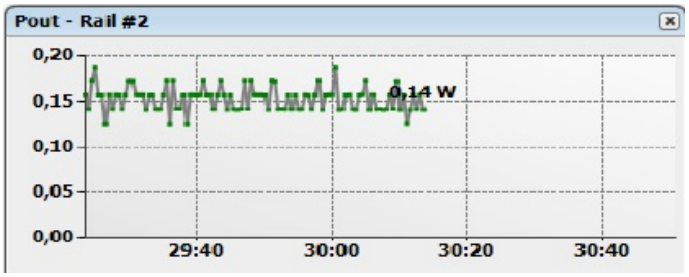

FIGURE V.4: Puissance de sortie de *PS* au repos (en Watt)

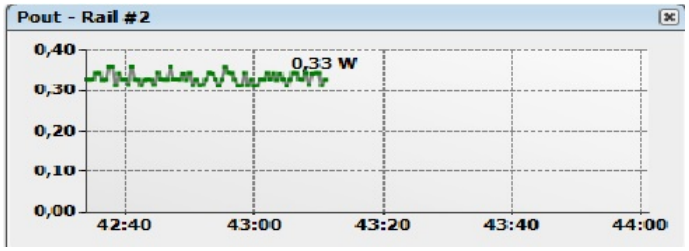

FIGURE V.5: Puissance de sortie de PS lors d'exécution de Linux embarqué (en Watt)

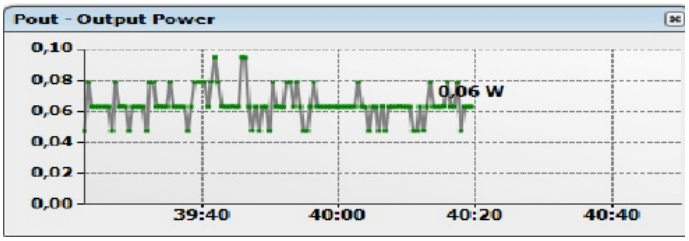

FIGURE V.6: Puissance de sortie de PL au repos (en Watt)

V.4 Exploration de l'espace de conception de plateforme embarquée

Dans cette partie nous abordons l'exploration de l'espace de conception de plateforme embarquée. Nous commençons par présenter la conception d'une architecture matérielle dédiée à la fusion de positions des obstacles détectés par une caméra stéréo et un LIDAR. Puis nous présentons les résultats expérimentaux des quelques *IP* qui sont nécessaires à la mise en œuvre de la plateforme et à travers lesquels nous explorons les performances des implémentations.

V.4.1 Architecture de fusion proposée

La figure V.7 présente l'architecture de fusion proposée pour fusionner les positions des obstacles détectés, fournies par n capteurs. Il s'agit d'une architecture générique et modulaire de fusion au niveau intermédiaire. Elle inclut trois processus principaux qui sont :

- Processus de calibration des capteurs : est déjà décrit dans le chapitre III et il est effectué en mode "hors-ligne" (l'étape d'initiation de la mise en service du véhicule). La sortie de ce processus est une transformation rigide permet d'assurer l'association des données entre les différents repères capteurs.
- Processus de détection des objets (en ligne) : chaque chaîne de détection, parmi n, fournit une liste des positions des objets détectés. Dans ce travail, nous nous intéressons aux capteurs LIDAR et caméra stéréo. Chaque capteur fournit une liste d'objets détectés, ces deux listes seront fusionnées afin d'obtenir une seule nouvelle liste d'objets fusionnés. Pour ce faire, nous utilisons la technique de fusion bayésienne.
- Processus de suivi (en ligne) : permet de suivre les objets détectés, il inclut principalement trois étapes qui sont :**(1)** Association des données entre les mesures (les nouveaux objets détectés) et les pistes existantes (les objets déjà détectés). L'objectif de cette étape est de sélectionner la mesure qui sera utilisée pour mettre à jour l'estimation d'état de la cible. Nous pouvons citer l'approche Multiple Hypotheses Tracking (MHT) [1] pour résoudre le problème d'association entre les mesures et les pistes. **(2)** La deuxième étape est la gestion des objets détectés et la prise de décision concernant les pistes, donc chaque piste peut être créée, confirmée ou supprimée. **(3)** La méthode de suivi précise comment les objets de la liste fusionnée seront suivis. Le filtre de Kalman est une solution appropriée pour cette tâche.

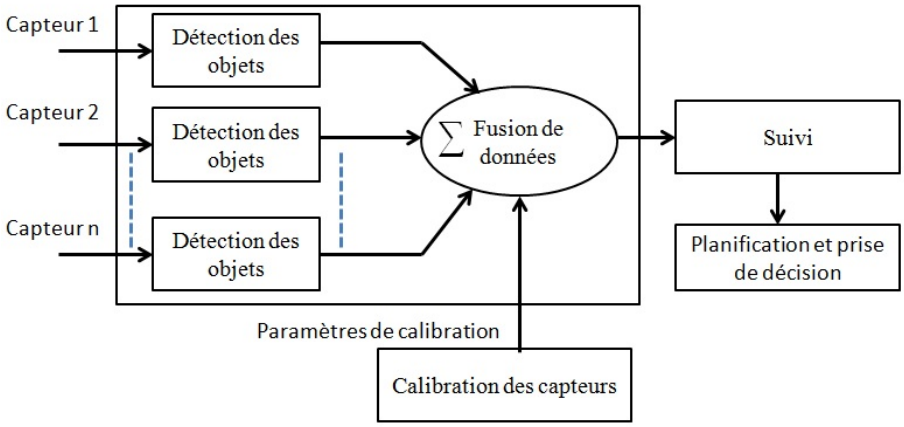

FIGURE V.7: Architecture de fusion proposée pour fusionner les positions des obstacles détectés

V.4.2 Technique bayésienne pour fusionner les positions des obstacles détectés

Nous utilisons la technique bayésienne pour fusionner les positions des objets détectés fournies par une caméra stéréo et un LIDAR. L'incertitude de position est représentée sous la forme d'une distribution gaussienne $2D$ pour les deux capteurs. La fusion bayésienne est décrite précédemment par l'équation III.30. À noter que cette technique de fusion s'appuie sur les matrices des covariances des capteurs pour pondérer les mesures afin d'estimer la positon fusionnée.

Pour expliquer le fonctionnement de l'approche bayésienne, nous simulons le comportement de deux capteurs qui détectent simultanément le même objet. Nous modélisons le déplacement de obstacle détecté par un mouvement uniforme avec une vitesse constante. Nous abordons le comportement de cette approche suivant trois cas en variant les valeurs des matrices des covariances :

- **Premier cas** (la figure V.8.a) : Le premier cas présente le résultat de fusion lorsque les deux capteurs ont des matrices des covariances similaires. Dans ce cas, les positions fusionnées estimées (les étoiles noires dans la figure) seront situées, aux milieux, entres les mesures fournies par les deux capteurs.
- **Deuxième cas** (la figure V.8.b) : Lorsque le capteur 2 présente des erreurs des mesures supérieures à ceux du capteur 1, les positions fusionnées suivent

les positions qui sont fournies par le capteur 1 (courbe bleue) car il présente des erreurs des mesures moins considérables que ceux de capteur 2. De ce fait, la position fusionnée fait "confiance" aux mesures du capteur 1 en s'appuyant sur sa matrice de covariance.

– **Troisième cas** (la figure V.8.c) : Lorsque le capteur 1 présente des erreurs des mesures supérieures à ceux du capteur 2, les positions fusionnées suivent les positions fournies par le capteur 2 (courbe rouge), contrairement au cas précédent.

Par conséquent, le résultat de la fusion bayésienne est une combinaison des mesures pondérées par les matrices des covariances de bruit.

V.4.3 Architecture matérielle proposée

La figure V.9 décrit l'architecture matérielle proposée dédiée au *SOC Zynq* − 7000. Dans la suite, nous détaillons cette architecture.

V.4.3.1 Carte Ethernet/FMC à quatre ports

Puisque les données de caméra stéréo et de LIDAR sont transmises à travers les ports Ethernet et la plateforme *Zynq* possède un seul port, alors nous utilisons la carte Ethernet/*FMC* à quatre ports [157]. Elle permet de multiplier le nombre de ports Ethernet à travers la carte d'extension *FMC*. La norme *FMC* du groupe VMEbus International Trade Association (VITA) résout partiellement le problème de l'obsolescence des Entrées/Sorties, en proposant un connecteur unique à 400 broches avec une bande passante globale potentielle de 40 Gb/s [84].

Cela signifie que l'interface des bus Entrées/Sorties d'un circuit imprimé est conçue séparément en tant qu'un module et il est couplé avec la plateforme embarquée (carte *Zynq* dans notre cas) à travers le connecteur *FMC*.

V.4.3.2 IP du traitement des données caméra

Les images provenant de la caméra sont stockées dans la mémoire *DDR* via le protocole *AXI-DMA*. Par la suite, ces images (gauche et droite) stockées seront

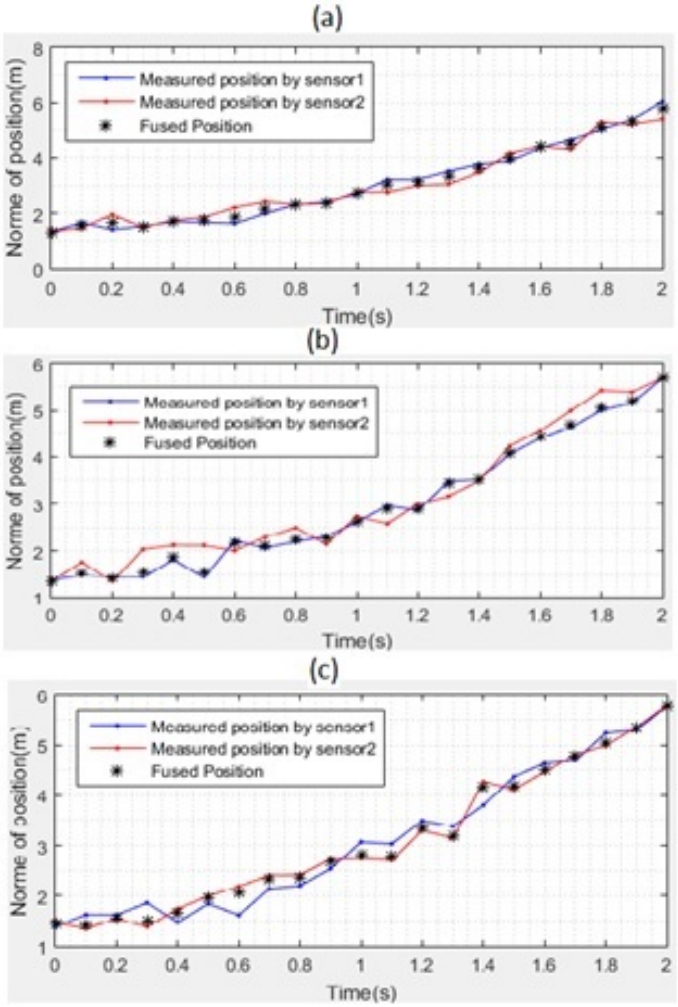

FIGURE V.8: Fusion des positions par l'approche bayésienne : (a) Les deux capteurs ont des matrices de covariance similaires (b) Le capteur 2 présente des erreurs des mesures supérieures à celles du capteur 1 (c) Le capteur 1 présente des erreurs des mesures supérieures à celles du capteur 2

envoyées séquentiellement à l'IP de traitement stéréo pour traiter la mise en correspondance et la reconstruction stéréo. Cet IP assure deux fonctionnalités principales qui sont : le calcul de disparité des objets détectés, suivi par le calcul de la profondeur pour estimer les positions des objets. La sortie de cet IP est un vecteur qui contient les cordonnées des objets détectés sous forme des *Bounding Boxes*. La tâche du calcul de disparité en utilisant l'approche de $Multi\ Window\ SAD$ est effectuée dans la thèse de Karim Ali [101].

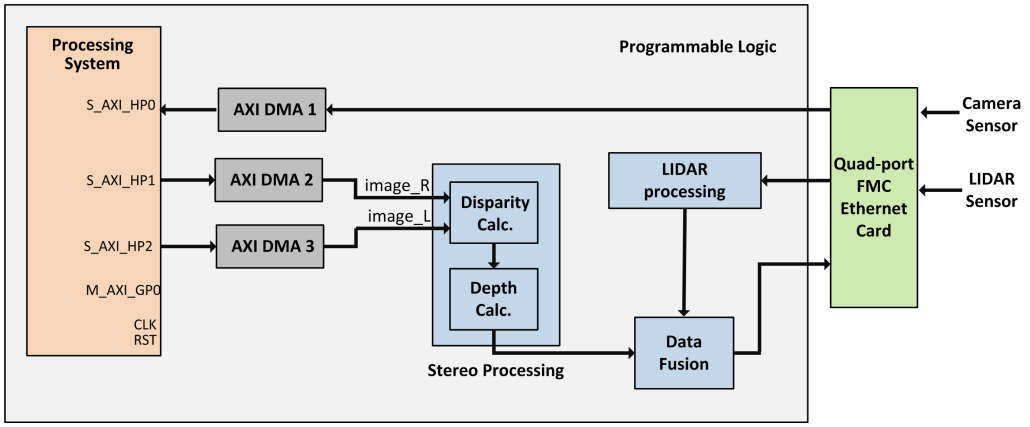

FIGURE V.9: Architecture matérielle proposée basée sur le Zynq-7000 SoC

V.4.3.3 IP du traitement des données LIDAR

L'entrée de l'*IP* de traitement de données LIDAR est un vecteur des mesures spécifiant les positions des objets détectés dans le système de coordonnées polaires. Après, le traitement et le processus de segmentation décrit dans la section III.3.3.1, la sortie de cet *IP* est un vecteur qui contient les cordonnées des segments (clusters) représentés par leurs barycentres.

V.4.3.4 IP de fusion de données

Cet *IP* permet d'assurer les fonctionnalités de fusion et suivi qui sont décrites dans la section V.4. Quant à la sortie, il s'agit d'un vecteur contenant les cordonnées des postions des objets détectés.

V.4.4 IP du filtre de Kalman

Afin d'explorer l'espace de conception de la plateforme embarquée, nous choisissons aussi de générer le code *VHDL* du filtre de Kalman. Ce filtre est indispensable dans les fonctionnalités de perception telles que la fusion ou le suivi des obstacles, et il nécessite une puissance du calcul importante à cause des opérations matricielles de son algorithme qui est décrit dans l'annexe A. Dans cette implémentation, le filtre est utilisé dans le but de suivre un obstacle détecté.

La figure V.10 décrit l'architecture matérielle de l'*IP* de filtre de Kalman. Il s'agit

d'un transfert de données en mode *stream* entre le PS et le PL (FPGA) au sein de $SoC\ Zynq-7000$. Le protocole de ce transfert utilise l'interface $AXI4-Stream$ qui permet d'avoir une transmission de données à haut débit. L'IP est couplé au contrôleur Direct Access Memory (DMA) pour transférer une grande quantité de données du PS vers le PL. Le contrôleur DMA lit les données à partir de la mémoire DDR et les envoie en mode *stream* à l'IP du filtre via l'interface $AXI4-Stream$.

Dans notre cas, le PS envoie les mesures et le PL renvoie la position estimée . À noter que le bus $AXI4-Lite$ est utilisé pour les communications simples et à faible débit, telles que la configuration des registres de contrôle ou d'état.

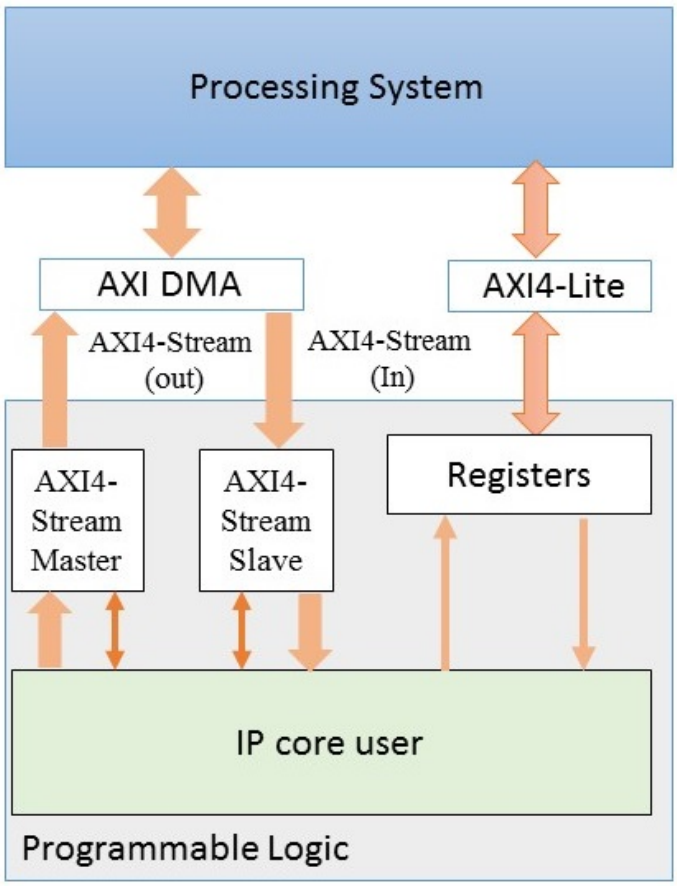

FIGURE V.10: Conception matérielle de filtre de Kalman avec l'interface AXI4-Stream

V.4.5 Implémentation du filtre de Kalman

En se basant sur la conception matérielle présentée dans la figure V.10, nous implémentons le filtre de Kalman en utilisant $HDL\ Coder$, l'outil HLS de Matlab $R2017b$. Le processeur ARM envoie les données à l'IP via l'interface $AXI4-Stream$, garantissant une transmission de données à haut débit. La figure V.11 décrit la communication entre les deux modules, maître et esclave, ainsi que leur chronogramme. À noter qu'il existe d'autres signaux d'Entrées/Sorties mais nous illustrons les principaux d'entre eux.

L'interface $AXI4-Stream$ intègre des signaux de données et de contrôle. Les principaux signaux de contrôle sont :

- **Valid signal** : Lorsque le signal de données est valide, ce signal est affirmé.
- **Ready signal** : Indique si le module esclave peut accepter de nouvelles données.
- **TLAST signal** : Ce signal indique la réception de la dernière donnée pendant un transfert.

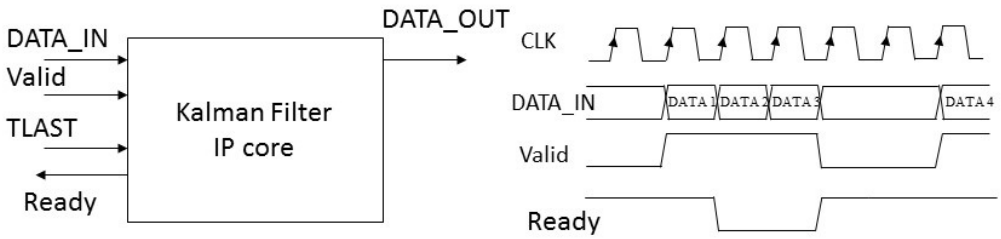

FIGURE V.11: Les Entrées/Sorties des modules maîtres/esclaves et leur chronogramme

V.4.6 Résultats expérimentaux

Le tableau V.1 présente le résultat de synthèse de l'IP de traitement de données LIDAR. Selon ce tableau, l'utilisation des ressources matérielles est faible, ce qui permet d'intégrer davantage d'autres IPs et par la suite autres fonctionnalités utiles pour la perception. En outre, la consommation électrique maximale du SoC (PS + PL) est d'environ 0.7 W.

Ressources	slices	FF	LUT	BRAM36
Utilisation HW	8445	32455	35554	23
Utilisation HW en (%)	15.45	7.42	16.26	4.15

TABLE V.1: Résultat de synthèse de l'IP de traitement de données LIDAR

Quant au tableau V.2, il présente l'occupation matérielle de l'implémentation de filtre de Kalman. Selon ce tableau, l'utilisation des ressources matérielles est faible.

Ressources	Slices	FF	LUT	BRAM36
Utilisation HW	211	10233	8435	5
Utilisation HW en (%)	1	2.34	3.85	1

TABLE V.2: Résultat de synthèse de l'IP du filtre de Kalman

Le tableau V.3, montre les temps d'exécution et la consommation de puissance maximale de différentes implémentations *SW* et *HW*. D'après ces résultats, l'implémentation *HW* a permis de réduire le temps d'exécution même avec une fréquence largement faible par rapport à celle de PC, 3 GHz. Par ailleurs, la performance de l'implémentation *HW* a été améliorée en réduisant sa consommation de 91 W vers 1.2 W.

En s'appuyant sur les résultats obtenus, l'implémentation *HW* sur le *SoC Zynq* a prouvé sa bonne performance en termes du temps d'exécution et de consommation de puissance.

Implémentations	SW (Intel Core i7, 3 GHz)	HW (FPGA, 200 MHz)
Temps en us	520	497
Consommation électrique en Watt	91	1.2

TABLE V.3: Performance temporelle et consommation de puissance de filtre de Kalman en implémentations *SW* et *HW*

Pour résumer, à travers des implémentations matérielles telles que le filtre de Kalman, des co-designs *SW/HW* dédiés à la plateforme *Zynq* sont présentés. Ces implémentations, ont montré des avantages en termes de rapidité et de consommation.

V.4.7 Conclusion

Dans ce chapitre, nous avons mis en avant une exploration de l'espace de conception d'une architecture embarquée de fusion multiplicateurs, à travers des implémentations des IPs matériels tels que l'IP de traitement de données de LIDAR et celui de filtre de Kalman. À titre d'exemple, avec une implémentation HW de filtre de Kalman, le temps d'exécution a été réduit et la consommation de puissance électrique aussi avec un facteur de *75x*.

De plus, l'outil *HLS* utilisé a permis d'avoir un temps de développement court, tout en garantissant de tester plusieurs alternatives de conception afin de générer des accélérateurs efficaces satisfaisant les contraintes de notre application.

Dans le prochain chapitre, nous conclurons ce mémoire et nous présenterons les perspectives liées à ce projet de thèse.

Chapitre VI

Conclusion et Perspectives

VI.1 Bilan

Le travail que nous avons présenté dans ce manuscrit s'inscrit dans le cadre de la conception d'une plateforme embarquée de fusion multi-capteurs pour la détection des obstacles et qui sera implémentée dans la navette autonome *ARMA* de la société *Navya*.

La conception de cette plateforme comporte deux volets, le premier est la fusion multi-capteurs. Comme mentionné précédemment, pour un véhicule autonome, un seul capteur n'est pas suffisant pour garantir une perception précise si le capteur a une grande incertitude, fournit de mesures insuffisantes et bruitées ou s'il a une faible résolution et un champ de vision limité. Il est donc nécessaire de développer des architectures multi-capteurs pour tirer profit des avantages de chaque capteur. L'emploi d'une telle architecture permet d'améliorer la précision de perception afin d'éviter les fausses alarmes et réduire les détections manquées dans un environnement incertain dans lequel se situe le véhicule. Mais avant de fusionner les données, la première étape consiste à unifier tous les repères capteurs, afin que les données issues de différents capteurs soient exprimées dans un seul référentiel. Quant au deuxième volet, c'est la mise en œuvre d'une plateforme embarquée permettant d'améliorer les performances globales du système de perception du véhicule tout en minimisant le coût de conception et de production. Ces performances s'articulent autour de trois axes principaux, à savoir, la performance temporelle, la consommation d'énergie

électrique et la sûreté de fonctionnement.

Cet ensemble d'objectifs nous a amené à diviser l'état de l'art en trois sections, qui sont la calibration multi-capteurs, en particulier celle de caméra/LIDAR, la détection des obstacles avec une architecture multi-capteurs et les implémentations embarquées dédiées à la perception du véhicule autonome. À partir de cette étude, nous avons présenté les travaux existants liés à ces trois axes, puis nous avons analysé les besoins nécessaires pour concevoir cette plateforme, enfin nous avons positionné nos travaux.

Dans une première contribution, nous avons proposé une méthodologie de calibration extrinsèque caméra/LIDAR en minimisant le nombre de poses requis tout en garantissant une bonne précision. L'utilisation d'une mire rectangulaire multi-lignes a permis de réduire le temps nécessaire pour la mise en œuvre de ce processus et d'améliorer sa précision puisque en augmentant le nombre de lignes, la répercussion des bruites des mesures a été réduite. De plus, nous avons développé un environnement d'extraction des caractéristiques de la mire afin d'automatiser le processus de calibration dans la mesure du possible et réduire l'intervention de l'utilisateur. Cet environnement inclut la détection des lignes par la caméra et les points des bords de la mire par le LIDAR. L'amélioration de la précision est démontrée en comparant la contribution proposée, tout d'abord, avec l'approche qui utilise une mire avec une seule ligne, puis avec l'approche itérative. Les résultats obtenus ont validé le bon fonctionnement de l'implémentation de l'approche proposée sur la navette autonome *ARMA*.

Dans un second temps, nous avons proposé une méthodologie de fusion entre les capteurs caméra et radar en s'appuyant sur la théorie des fonctions de croyance. Notre objectif est d'améliorer la détection et la reconnaissance des obstacles, y compris les piétons. Tout d'abord, nous avons justifié le choix de cette théorie pour la représentation des connaissances ainsi que pour modéliser et gérer les incertitudes provenant des mesures des capteurs. Puis, nous avons introduit les concepts fondamentaux de la théorie concernée en analysant les propriétés des opérateurs de combinaison pour la fusion des fonctions de croyance. Étant donné que chaque opérateur a ses propres propriétés et il est utilisé dans un cas bien précis, cette

analyse nous a permis de choisir l'opérateur adéquat pour la fusion des fonctions de croyance. Les fonctions de croyance sont conclues à partir de la détection des piétons effectuée par la caméra d'un côté, et les vitesses ainsi que les paramètres *RCS* des obstacles détectés par le radar d'autre côté. Ensuite, nous avons détaillé les différentes étapes nécessaires pour développer l'approche proposée. Les résultats de simulation obtenus ont montré que le système conçu fonctionne comme prévu. Par conséquent, le module de fusion a permis d'assurer la continuité de détection et de reconnaissance de l'obstacle. En effet, en absence de détection de piéton par un capteur, le deuxième continue à effectuer cette tâche.

Dans un troisième temps, nous avons exploré l'espace de conception de la plateforme embarquée. Parmi les technologies d'accélération du traitement, nous avons choisi la solution FPGA permettant d'offrir une puissance de calcul performante, par le moyen de ses ressources reconfigurables, avec une faible consommation de puissance tout en garantissant un coût maitrisé. De plus, pour palier aux problèmes qui sont liés à la méthodologie de la conception conventionnelle basée sur le codage manuel, nous avons utilisé les outils *HLS*. Ces derniers permettent aux concepteurs de tester rapidement les différents algorithmes et leurs alternatives d'implémentation dans un cycle de conception court.

VI.2 Perspectives

Le travaux entrepris durant cette thèse peuvent être poursuivis suivant plusieurs directions, nous en présentons ici quelques-unes :

Calibration de caméra et LIDAR :

La navette autonome ARMA utilise des capteurs LIDAR mono-napppe et multi-nappes, 16 nappes, c'est à dire 16 faisceaux lasers de balayage. Nous proposons d'étendre notre méthodologie pour qu'elle soit utilisée dans la calibration des capteurs caméra et LIDAR multi-nappes. De plus, nous proposons de développer une approche d'auto-calibration, dans ce cas, la calibration entre les capteurs sera déterminée sans l'utilisation d'une mire et sans l'intervention humaine, ce qui permettra d'automatiser la mise en service de la navette.

Fusion multi-capteurs :

En ce qui concerne la fusion, nous proposons d'étendre le développement de l'approche de fusion pour la détection multi-obstacles. Dans ce cas, un module de suivi sera développé comportant principalement trois étapes : **(1)** L'association de données entre les mesures (nouvel objet détecté) et les pistes déjà existantes. L'association de données a pour objectif de sélectionner les mesures qui seront utilisées pour mettre à jour l'estimation de l'état cible. Nous citons l'approche Multiple Hypotheses tracking (MHT) [1] qui est utilisée souvent pour résoudre ce problème. **(2)** La deuxième étape est la gestion des objets détectés et la prise de décision concernant les pistes, donc une piste sera créée, confirmée ou supprimée. **(3)** La troisième étape est le suivi des obstacles de la liste fusionnée. Le filtre de Kalman est une solution appropriée pour cette étape. Également, l'approche de fusion pourra être étendue pour qu'elle inclut les observations fournies par le LIDAR et ce qui permettra d'améliorer la précision de détection. En fait, nous avons mentionné précédemment que le LIDAR permet d'estimer les positions latérales des obstacles avec une précision plus efficace que celle du radar. Il peut aussi estimer la zone d'occupation d'un objet et fournir une représentation détaillée de la scène.

Industrialisation et mise en œuvre de la plateforme embarquée :

Une phase d'industrialisation verra le jour chez Navya dans l'objectif dévaluer le système de perception existant et changer les PC industriels par des calculateurs embarqués performants. De ce fait, notre motivation principale est la conception et la mise en œuvre d'une plateforme embarquée. À court terme, nous proposons de synthétiser et implémenter les algorithmes de fusion proposés dans ce travail de thèse sur la carte de développement *Zynq* 7000 à l'aide des outils HLS afin de valider le bon fonctionnement de la fonctionnalité *DTMO*. À long terme, le but sera que la plateforme embarquée accueillera la perception entière de la navette autonome afin d'améliorer la performance temporelle, réduire la consommation d'énergie et garantir la sûreté de fonctionnement.

Bibliographie

[1] H. Durrant-Whyte and T. C. Henderson, *Multisensor data fusion.* Springer, 2008.

[2] C.-C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects : Theory and results from a ground vehicle in crowded urban areas," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 1, pp. 842–849, IEEE, 2003.

[3] Q. B. Baig, *Multi sensor data fusion for detection and tracking of moving objects from a dynamic autonomous vehicle.* Theses, Université de Grenoble, Feb. 2012.

[4] "Frame rate vs stopping distance." https ://www.eetimes.com. Accessed : 2018-08-29.

[5] Y. Li, *Stereo vision and Lidar based dynamic occupancy grid mapping : Application to scenes analysis for intelligent vehicles.* PhD thesis, Université de Technologie de Belfort-Montbeliard, 2013.

[6] V. Lemonde, *Stéréovision embarquée sur véhicule : De l'auto-calibrage à la détection d'obstacles.* PhD thesis, INSA de Toulouse, 2005.

[7] "Le système sur puce zynq 7000." https ://www.xilinx.com/support/documentation. Accessed : 2018-12-03.

[8] G. Welch and G. Bishop, "An introduction to the kalman filter. department of computer science, university of north carolina," *ed : Chapel Hill, NC, unpublished manuscript*, 2006.

[9] "Tableau comparatif de performance des capteurs." https ://www.vdclab.kaist.ac.kr/bbs/. Accessed : 2018-10-31.

[10] J. Moras, *Grilles de perception évidentielles pour la navigation robotique en milieu urbain*. PhD thesis, Université de Technologie de Compiègne, 2013.

[11] "Les minibus autonomes de navya arma." https ://www.usinenouvelle.com/article/les-minibus-autonomes-de-navya-arma-parcourent-la-centrale-nucleaire-de-civaux.N427612. Accessed : 2018-06-29.

[12] "Navya company." http ://navya.tech. Accessed : 2018-06-27.

[13] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[14] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on robotics and automation*, vol. 17, no. 3, pp. 229–241, 2001.

[15] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2-3, pp. 123–139, 2009.

[16] R. Siegwart, I. R. Nourbakhsh, D. Scaramuzza, and R. C. Arkin, *Introduction to autonomous mobile robots*. MIT press, 2011.

[17] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[18] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv :1604.07316*, 2016.

[19] A. Y. Lam, Y.-W. Leung, and X. Chu, "Autonomous-vehicle public transportation system : Scheduling and admission control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1210–1226, 2016.

[20] D. Dubois, "Théorie des possibilités ; applications a la représentation des connaissances en informatique," tech. rep., 1988.

[21] F. Fayad, *Gestion de la confiance dans un système de fusion multisensorielle : application à la détection de piétons en situations routières*. PhD thesis, Compiègne, 2009.

[22] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion : A review of the state-of-the-art," *Information Fusion*, vol. 14, no. 1, pp. 28 – 44, 2013.

[23] T. Denœux, "Théorie des fonctions de croyance : application en reconnaissance de formes et en fusion d'informations," 2010.

[24] E. W. Weisstein, "mathworld. from mathworld–a wolfram web resource," 1999.

[25] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014.

[26] X. Wen, L. Shao, Y. Xue, and W. Fang, "A rapid learning algorithm for vehicle classification," *Information Sciences*, vol. 295, pp. 395–406, 2015.

[27] J. Ćesić, I. Marković, I. Cvišić, and I. Petrović, "Radar and stereo vision fusion for multitarget tracking on the special euclidean group," *Journal of Robotics and Autonomous Systems*, vol. vol. 83, pp. 338–348, 2016.

[28] A. Martin, "La fusion d'informations," *Polycopié de cours ENSIETA-Réf*, vol. 1484, p. 117, 2005.

[29] J. A. Stover, D. L. Hall, and R. E. Gibson, "A fuzzy-logic architecture for autonomous multisensor data fusion," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 3, pp. 403–410, 1996.

[30] B. Bouchon-Meunier, *La logique floue :«Que sais-je ?» n° 2702*. Presses universitaires de France, 2007.

[31] P. Smets, "Data fusion in the transferable belief model," in *Information Fusion, 2000. FUSIon 2000. Proceedings of the Third International Conference on*, vol. 1, pp. PS21–PS33, IEEE, 2000.

[32] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information.," in *AAAI*, 2012.

[33] Y. Park, S. Yun, C. S. Won, K. Cho, K. Um, and S. Sim, "Calibration between color camera and 3d lidar instruments with a polygonal planar board," *Journal of Sensors*, vol. vol. 14, no. no. 3, pp. 5333–5353, 2014.

[34] Y. Li, Y. Ruichek, and C. Cappelle, "Optimal extrinsic calibration between a stereoscopic system and a lidar," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 8, pp. 2258–2269, 2013.

[35] L. Zhou and Z. Deng, "A new algorithm for the establishing data association between a camera and a 2-d lidar," *Journal of Science and Technology*, vol. vol. 19, no. no. 3, pp. 314–322, 2014.

[36] S. Debattisti, L. Mazzei, and M. Panciroli, "Automated extrinsic laser and camera inter-calibration using triangular targets," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pp. 696–701, IEEE, 2013.

[37] C. X. Guo and S. I. Roumeliotis, "An analytical least-squares solution to the line scan lidar-camera extrinsic calibration problem," in *Proceedings of the International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany*, pp. 2943–2948, IEEE, 2013.

[38] V. Fremont, S. A. Rodriguez F, and P. Bonnifait, "Circular targets for 3d alignment of video and lidar sensors," *Journal of Advanced Robotics*, vol. vol. 26, no. no. 18, pp. 2087–2113, 2012.

[39] W. Dong and V. Isler, "A novel method for the extrinsic calibration of a 2-d laser-rangefinder and camera," in *Proceedings of The IEEE International Conference on Robotics and Automation (ICRA), Singapour*, pp. 5104–5109, IEEE, 2017.

[40] V. John, Q. Long, Z. Liu, and S. Mita, "Automatic calibration and registration of lidar and stereo camera without calibration objects," in *Vehicular Electronics and Safety (ICVES), 2015 IEEE International Conference on*, pp. 231–237, IEEE, 2015.

[41] K. Banerjee, D. Notz, J. Windelen, S. Gavarraju, and M. He, "Online camera lidar fusion and object detection on hybrid data for autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1632–1638, IEEE, 2018.

[42] J. Jiang, P. Xue, S. Chen, Z. Liu, X. Zhang, and N. Zheng, "Line feature based extrinsic calibration of lidar and camera," in *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 1–6, IEEE, 2018.

[43] M. Trajković and M. Hedley, "Fast corner detection," *Image and vision computing*, vol. 16, no. 2, pp. 75–87, 1998.

[44] M. A. Fischler and R. C. Bolles, "Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[45] D. P. Bertsekas, *Nonlinear programming*. Athena scientific Belmont, 1999.

[46] K. Sentz, S. Ferson, *et al.*, *Combination of evidence in Dempster-Shafer theory*, vol. 4015. Citeseer, 2002.

[47] F. Smarandache and J. Dezert, *Advances and Applications of DSmT for Information Fusion, Vol. IV : Collected Works*. Infinite Study, 2015.

[48] M. Liggins II, D. Hall, and J. Llinas, *Handbook of multisensor data fusion : theory and practice*. CRC press, 2017.

[49] R. O. Chavez-Garcia, *Multiple sensor fusion for detection, classification and tracking of moving objects in driving environments*. PhD thesis, Université de Grenoble, 2014.

[50] M. Chojnacki and V. Indelman, "Vision-based dynamic target trajectory and ego-motion estimation using incremental light bundle adjustment," *International Journal of Micro Air Vehicles*, vol. 10, no. 2, pp. 157–170, 2018.

[51] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007.

[52] C. Urmson, "Google self-driving car project," *SXSW Interactive*, 2016.

[53] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and

mapping : Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[54] O. El Hamzaoui, *Localisation et cartographie simultanées pour un robot mobile équipé d'un laser à balayage : CoreSLAM.* PhD thesis, Paris, ENMP, 2012.

[55] F. Moosmann and C. Stiller, "Velodyne slam," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 393–398, IEEE, 2011.

[56] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based slam," *The International Journal of Robotics Research*, vol. 26, no. 4, pp. 335–359, 2007.

[57] H. Carrillo, P. Dames, V. Kumar, and J. A. Castellanos, "Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 487–494, IEEE, 2015.

[58] E. Javanmardi, M. Javanmardi, Y. Gu, and S. Kamijo, "Autonomous vehicle self-localization based on probabilistic planar surface map and multi-channel lidar in urban area," in *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pp. 1–8, IEEE, 2017.

[59] K. Werber, M. Barjenbruch, J. Klappstein, J. Dickmann, and C. Waldschmidt, "Roughcough—a new image registration method for radar based vehicle self-localization," in *Information Fusion (Fusion), 2015 18th International Conference on*, pp. 1533–1541, IEEE, 2015.

[60] T. Pire, T. Fischer, J. Civera, P. De Cristóforis, and J. J. Berlles, "Stereo parallel tracking and mapping for robot localization," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 1373–1378, IEEE, 2015.

[61] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *null*, p. 1403, IEEE, 2003.

[62] J. A. Castellanos, J. Neira, and J. D. Tardós, "Multisensor fusion for simultaneous localization and map building," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 908–914, 2001.

[63] K. Walas, M. Nowicki, D. Ferstl, and P. Skrzypczyński, "Depth data fusion for simultaneous localization and mapping—rgb-dd slam," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2016 IEEE International Conference on*, pp. 9–14, IEEE, 2016.

[64] J. Cheng, Z. Xiang, T. Cao, and J. Liu, "Robust vehicle detection using 3d lidar under complex urban environment," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 691–696, IEEE, 2014.

[65] C. Tzomakas and W. von Seelen, "Vehicle detection in traffic scenes using shadows," in *Ir-Ini, Institut fur Nueroinformatik, Ruhr-Universitat*, Citeseer, 1998.

[66] S. Nedevschi, S. Bota, and C. Tomiuc, "Stereo-based pedestrian detection for collision-avoidance applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 380–391, 2009.

[67] M. Bertozzi, L. Castangia, S. Cattani, A. Prioletti, and P. Versari, "360 detection and tracking algorithm of both pedestrian and vehicle using fisheye images," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pp. 132–137, IEEE, 2015.

[68] H. Liu and S. Niar, "Radar signature in multiple target tracking system for driver assistant application," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 887–892, EDA Consortium, 2013.

[69] M. Oliveira, V. Santos, A. D. Sappa, and P. Dias, "Scene representations for autonomous driving : An approach based on polygonal primitives," in *Robot 2015 : Second Iberian Robotics Conference*, pp. 503–515, Springer, 2016.

[70] A. Mukhtar, L. Xia, and T. B. Tang, "Vehicle detection techniques for collision avoidance systems : A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2318–2338, 2015.

[71] C. Blanc, L. Trassoudaine, Y. Le Guilloux, and R. Moreira, "Track to track fusion method applied to road obstacle detection," in *International Conference on Information Fusion*, 2004.

[72] Q. Baig, O. Aycard, T. D. Vu, and T. Fraichard, "Fusion between laser and stereo vision data for moving objects tracking in intersection like scenario," in *Proceedings of Intelligent Vehicles Symposium (IV)*, pp. 362–367, IEEE, 2011.

[73] B. Salina and P. Malathi, "Fpga implementation of data fusion algorithm for object localization," in *Recent Advances and Innovations in Engineering (ICRAIE), 2014*, pp. 1–5, IEEE, 2014.

[74] R. Durrett, *Probability : theory and examples*. Cambridge university press, 2010.

[75] H. Cho, Y.-W. Seo, B. V. Kumar, and R. R. Rajkumar, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 1836–1843, IEEE, 2014.

[76] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto, "A lidar and vision-based approach for pedestrian and vehicle detection and tracking," in *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pp. 1044–1049, IEEE, 2007.

[77] F. Fayad and V. Cherfaoui, "Detection and recognition confidences update in a multi-sensor pedestrian tracking system," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 409–416, 2008.

[78] R. O. Chavez-Garcia and O. Aycard, "Multiple sensor fusion and classification for moving object detection and tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 525–534, 2016.

[79] Q. Baig and O. Aycard, "Low level data fusion of laser and monocular color camera using occupancy grid framework," in *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, pp. 905–910, IEEE, 2010.

[80] M. Aeberhard and N. Kaempchen, "High-level sensor data fusion architecture for vehicle surround environment perception," in *Proc. 8th Int. Workshop Intell. Transp*, 2011.

[81] A. Teichman, J. Levinson, and S. Thrun, "Towards 3d object recognition via classification of arbitrary object tracks," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4034–4041, IEEE, 2011.

[82] "Vehicle detector." https ://www.mathworks.com/help/driving/examples/. Accessed : 2018-12-13.

[83] J. H. Gawron, G. A. Keoleian, R. D. De Kleine, T. J. Wallington, and H. C. Kim, "Life cycle assessment of connected and automated vehicles : Sensing and computing subsystem and vehicle level effects," *Environmental science & technology*, vol. 52, no. 5, pp. 3249–3256, 2018.

[84] M. Bouain, V. Viswanathan, R. B. Atitallah, and J.-L. Dekeyser, "Communication-centric design for fmc based i/o system," in *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2014 9th International Symposium on*, pp. 1–8, IEEE, 2014.

[85] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "Caad : Computer architecture for autonomous driving," *arXiv preprint arXiv :1702.01894*, 2017.

[86] "La norme iso 26262 :2011." https ://www.iso.org/standard/43464.html. Accessed : 2018-06-27.

[87] D. D. Ward and S. E. Crozier, "The uses and abuses of asil decomposition in iso 26262," in *System Safety, incorporating the Cyber Security Conference 2012, 7th IET International Conference on*, pp. 1–6, IET, 2012.

[88] Y. R. Qu, H. H. Zhang, S. Zhou, and V. K. Prasanna, "Optimizing many-field packet classification on fpga, multi-core general purpose processor, and gpu," in *Architectures for Networking and Communications Systems (ANCS), 2015 ACM/IEEE Symposium on*, pp. 87–98, IEEE, 2015.

[89] S. Baruah, M. Bertogna, and G. Buttazzo, *Multiprocessor scheduling for real-time systems*. Springer, 2015.

[90] P. Guillou, *Compilation efficace d'applications de traitement d'images pour processeurs manycore*. PhD thesis, PSL Research University, 2016.

[91] N. Einspruch, *Application specific integrated circuit (ASIC) technology*, vol. 23. Academic Press, 2012.

[92] M. Daga, A. M. Aji, and W.-c. Feng, "On the efficacy of a fused cpu+ gpu processor (or apu) for parallel computing," in *Application Accelerators in High-Performance Computing (SAAHPC), 2011 Symposium on*, pp. 141–149, IEEE, 2011.

[93] C. A. Navarro, N. Hitschfeld-Kahler, and L. Mateu, "A survey on parallel computing and its applications in data-parallel problems using gpu architectures," *Communications in Computational Physics*, vol. 15, no. 2, pp. 285–329, 2014.

[94] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 1–12, ACM, 2017.

[95] "Soc intel fpgas." https ://www.altera.com/products/soc/overview.html. Accessed : 2018-07-23.

[96] S. Bauer, S. Köhler, K. Doll, and U. Brunsmann, "Fpga-gpu architecture for kernel svm pedestrian detection," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pp. 61–68, IEEE, 2010.

[97] "Le processeur leon de la société gaisler." https ://www.gaisler.com. Accessed : 2018-07-03.

[98] F. Sun, H. Wang, F. Fu, and X. Li, "Survey of fpga low power design," in *Intelligent Control and Information Processing (ICICIP), 2010 International Conference on*, pp. 547–550, IEEE, 2010.

[99] Y. Li, K. Gai, M. Qiu, W. Dai, and M. Liu, "Adaptive human detection approach using fpga-based parallel architecture in reconfigurable hardware," *Concurrency and Computation : Practice and Experience*, vol. 29, no. 14, 2017.

[100] L. Puglia, M. Vigliar, and G. Raiconi, "Real-time low-power fpga architecture for stereo vision," *IEEE Transactions on Circuits and Systems II : Express Briefs*, vol. 64, no. 11, pp. 1307–1311, 2017.

[101] K. M. A. Ali, *Architectures parallèles reconfigurables pour le traitement vidéo temps-réel*. PhD thesis, Valenciennes, 2018.

[102] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "Computer architectures for autonomous driving," *Computer*, vol. 50, no. 8, pp. 18–25, 2017.

[103] N. Abel, S. Manz, F. Grull, and U. Kebschull, "Increasing design changeability using dynamical partial reconfiguration," in *Real Time Conference, 2009. RT'09. 16th IEEE-NPSS*, pp. 381–387, IEEE, 2009.

[104] J. Ahmad and A. Warren, "Fpga based deterministic latency image acquisition and processing system for automated driving systems," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*, pp. 1–5, IEEE, 2018.

[105] K. Uetsuhara, H. Nagayama, Y. Shibata, and K. Oguri, "Discussion on high level synthesis fpga design of camera calibration," in *Conference on Complex, Intelligent, and Software Intensive Systems*, pp. 538–549, Springer, 2018.

[106] Y. Lyu, L. Bai, and X. Huang, "Real-time road segmentation using lidar data processing on an fpga," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*, pp. 1–5, IEEE, 2018.

[107] F. Schwiegelshohn, L. Gierke, and M. Hübner, "Fpga based traffic sign detection for automotive camera systems," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2015 10th International Symposium on*, pp. 1–6, IEEE, 2015.

[108] R. Nane, V.-M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi, *et al.*, "A survey and evaluation of fpga high-level synthesis tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 10, pp. 1591–1604, 2016.

[109] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 161–170, ACM, 2015.

[110] G. Zhong, S. Niar, A. Prakash, and T. Mitra, "Design of multiple-target tracking system on heterogeneous system-on-chip devices," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4802–4812, 2016.

[111] A. Gamatié, S. Le Beux, É. Piel, R. Ben Atitallah, A. Etien, P. Marquet, and J.-L. Dekeyser, "A model-driven design framework for massively parallel embedded systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 10, no. 4, p. 39, 2011.

[112] L. Zhang, M. Glaß, N. Ballmann, and J. Teich, "Bridging algorithm and esl design : Matlab/simulink model transformation and validation," in *Languages, Design Methods, and Tools for Electronic System Design*, pp. 189–206, Springer, 2015.

[113] C. Khongprasongsiri, P. Kumhom, W. Suwansantisuk, T. Chotikawanid, S. Chumpol, and M. Ikura, "A hardware implementation for real-time lane detection using high-level synthesis," in *Advanced Image Technology (IWAIT), 2018 International Workshop on*, pp. 1–4, IEEE, 2018.

[114] A. Dine, *Localisation et cartographie simultanées par optimisation de graphe sur architectures hétérogènes pour l'embarqué.* PhD thesis, Université Paris-Saclay, 2016.

[115] A. Qamar, F. B. Muslim, and L. Lavagno, "Analysis and implementation of the semi-global matching 3d vision algorithm using code transformations and high-level synthesis," in *Proceedings of the 81st Vehicular Technology Conference (VTC Spring), Glasgow, United Kingdom*, pp. 1–5, IEEE, 2015.

[116] "Vision par ordinateur." https ://team.inria.fr/steep/files/2015/03/poly. Accessed : 2018-12-18.

[117] "Capteur lidar." http ://news.voyage.auto/. Accessed : 2018-12-19.

[118] B. A. Garcia, *Conception d'un radar d'aide à la conduite automobile utilisant un système discriminateur de fréquence type" six-port".* PhD thesis, Télécom ParisTech, 2002.

[119] G. Bradski and A. Kaehler, "Opencv," *Dr. Dobb's journal of software tools*, vol. 3, 2000.

[120] J.-Y. Bouguet, "Complete camera calibration toolbox for matlab (1999)," *URL http ://www. vision. caltech. edu/bouguetj.*

[121] L. A. Rosero and F. S. Osório, "Calibration and multi-sensor fusion for on-road obstacle detection," in *Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR), 2017 Latin American*, pp. 1–6, IEEE, 2017.

[122] S. Li, C. Xu, and M. Xie, "A robust o (n) solution to the perspective-n-point problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1444–1450, 2012.

[123] R. Horaud and O. Monga, *Vision par ordinateur : outils fondamentaux*. Editions Hermès, 1995.

[124] N. Trawny and S. I. Roumeliotis, "Indirect kalman filter for 3d attitude estimation," *University of Minnesota, Dept. of Comp. Sci. & Eng., Technical Report , Number 2005-002, Rev. 57*, vol. 2, p. 2005, 2005.

[125] R. Fisher, S. Perkins, A. Walker, and E. Wolfart, "Spatial filters-laplacian of gaussian," 2003.

[126] *Maïtine Bergounioux. Quelques méthodes de filtrage en Traitement d'Image. Cours donné dans le cadre d'une école CIMPA - en attente de publication dans les actes. 2010. <hal-00512280v1>.*

[127] W. Rong, Z. Li, W. Zhang, and L. Sun, "An improved canny edge detection algorithm," in *Proceedings of the International Conference on Mechatronics and Automation (ICMA), Tianjin, China*, pp. 577–582, IEEE, 2014.

[128] W. Gao, X. Zhang, L. Yang, and H. Liu, "An improved sobel edge detection," in *Proceedings of the 3rd IEEE International Computer Science and Information Technology (ICCSIT), Chengdu, China*, vol. 5, pp. 67–71, IEEE, 2010.

[129] R. Maini and H. Aggarwal, "Study and comparison of various image edge detection techniques," *International journal of image processing (IJIP)*, vol. vol. 3, no. no. 1, pp. 1–11, 2009.

[130] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," in *Readings in computer vision*, pp. 714–725, Elsevier, 1987.

[131] Y. Li, Y. Ruichek, and C. Cappelle, "Optimal extrinsic calibration between a stereoscopic system and a lidar," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, pp. 2258–2269, Aug 2013.

[132] E. Shang, X. An, M. Shi, D. Meng, J. Li, and T. Wu, "An efficient calibration approach for arbitrary equipped 3-d lidar based on an orthogonal normal vector pair," *Journal of Intelligent & Robotic Systems*, vol. vol. 79, no. no. 1, pp. 21–36, 2015.

[133] C. Sanderson, "Armadillo : An open source c++ linear algebra library for fast prototyping and computationally intensive experiments," *Technical report, NICTA*, 2010.

[134] Y. Lemeret, E. Lefevre, and D. Jolly, "Fusion de données provenant d'un laser et d'un radar en utilisant la théorie de dempster-shafer," *MAJECSTIC'04, France, 2004*, 2004.

[135] P. Vannoorenberghe, "Un état de l'art sur les fonctions de croyance appliquées au traitement de l'information," *Revue I3*, vol. 3, no. 2, pp. 9–45, 2003.

[136] A. Samet, *Théorie des fonctions de croyance : application des outils de data mining pour le traitement des données imparfaites.* PhD thesis, Artois, 2014.

[137] G. Shafer, "Dempster-shafer theory," *Encyclopedia of artificial intelligence*, pp. 330–331, 1992.

[138] G. Shafer, *A mathematical theory of evidence*, vol. 42. Princeton university press, 1976.

[139] P. Smets and R. Kennes, "The transferable belief model," *Artificial intelligence*, vol. 66, no. 2, pp. 191–234, 1994.

[140] E. Ramasso, M. Rombaut, and D. Pellerin, "Modèle des croyances transférables : Représentation des connaissances, fusion d'informations, décision," 2007.

[141] E. Ramasso, *Reconnaissance de séquences d'états par le Modèle des Croyances Transférables. Application à l'analyse de vidéos d'athlétisme.* PhD thesis, Université Joseph-Fourier-Grenoble I, 2007.

[142] E. Lefevre, O. Colot, and P. Vannoorenberghe, "Belief function combination and conflict management," *Information fusion*, vol. 3, no. 2, pp. 149–162, 2002.

[143] A. Martin, "Le conflit dans la théorie des fonctions de croyance.," in *EGC*, pp. 655–666, 2010.

[144] A. Jøsang, "Categories of belief fusion," *JOURNAL OF ADVANCES IN INFORMATION FUSION*, 2018.

[145] P. Djiknavorian, D. Grenier, and P. Valin, "Analysis of information fusion combining rules under the dsm theory using esm inputs," in *Information Fusion, 2007 10th International Conference on*, pp. 1–8, IEEE, 2007.

[146] A. P. Dempster, "A generalization of bayesian inference," in *Classic works of the dempster-shafer theory of belief functions*, pp. 73–104, Springer, 2008.

[147] T. Denceux, "The cautious rule of combination for belief functions and some extensions," in *Information Fusion, 2006 9th International Conference on*, pp. 1–8, IEEE, 2006.

[148] T. Denœux, "Conjunctive and disjunctive combination of belief functions induced by nondistinct bodies of evidence," *Artificial Intelligence*, vol. 172, no. 2-3, pp. 234–264, 2008.

[149] P. Smets, "Decision making in the tbm : the necessity of the pignistic transformation," *International Journal of Approximate Reasoning*, vol. 38, no. 2, pp. 133–148, 2005.

[150] S. J. J, "Yet another paradigm illustrating evidence fusion (yapief)," in *Information Fusion, 2006 9th International Conference on*, pp. 1–7, IEEE, 2006.

[151] E. F. Knott, *Radar cross section measurements*. Springer Science & Business Media, 2012.

[152] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection : Survey and experiments," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 12, pp. 2179–2195, 2008.

[153] G. Marion, A. A. Mekonnen, and F. Lerasle, "Pertinence des combinaisons traqueur-détecteur pour le suivi-par-détection," in *Congrès Reconnaissance des Formes et Intelligence Artificielle*, 2016.

[154] J. J. Sudano, "Inverse pignistic probability transforms," in *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, vol. 2, pp. 763–768, IEEE, 2002.

[155] "Protocole d'interface amba axi4." https ://www.xilinx.com/products/intellectual-property/axi.html. Accessed : 2018-12-03.

[156] "Power management bus." http ://www.ti.com/power-management/digital-power/tools-software.html. Accessed : 2018-12-03.

[157] "carte ethernet/fmc." https ://ethernetfmc.com/. Accessed : 2018-12-03.

Annexes

A Filtre de Kalman

Le filtre de Kalman résout le problème d'estimation d'état $x \in \Re^n$ d'un processus contrôlé en temps discret qui est modélisé par une équation différentielle stochastique linéaire :

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{A.1}$$

avec $z \in \Re^m$ est le vecteur de mesure :

$$z_k = Hx_k + v_k \tag{A.2}$$

avec :

- $x \in \Re^n$: Vecteur d'état
- $u \in \Re^l$: Vecteur de commande (entrée optionnelle)
- $z \in \Re^m$: Vecteur de mesure
- $A \in \Re^{n \times n}$: Matrice d'état
- $B \in \Re^{n \times l}$: Matrice de commande
- $H \in \Re^{m \times n}$: Matrice d'observation
- w_k : bruit de processus
- v_k : bruit de mesure
- Q : Covariance de bruit de processus
- R : Covariance de bruit de mesure

La figure A.1 décrit l'algorithme de filtre de Kalman qui comporte deux phases : prédiction et correction (mise à jour).

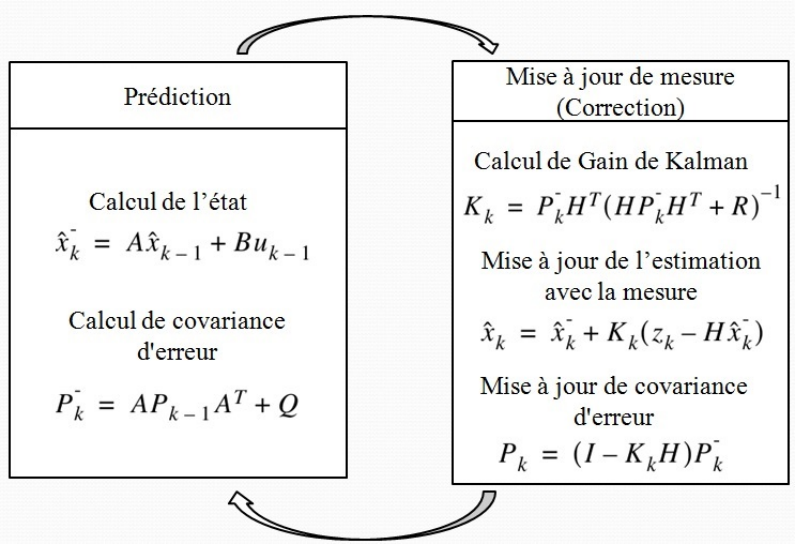

FIGURE A.1: Algorithme du filtre de Kalman [8] : prédiction et correction

B Propriétés des quaternions

- Le quaternion est généralement défini comme suit :

$$\bar{q} = q_4 + q_1 i + q_2 j + q_3 k \tag{B.1}$$

où i, j et k sont des nombres hyper-complexes et q_4 est la partie réelle ou scalaire du quaternion.

-Pour convertir un vecteur *3D* $\overrightarrow{p}$ en quaternion, nous utilisons :

$$\bar{p} = [\overrightarrow{p} \quad 0]^T \tag{B.2}$$

-Le produit $\overrightarrow{p}_2 = R\overrightarrow{p}_1$, où $\overrightarrow{p}_1$, $\overrightarrow{p}_2$ sont des vecteurs, R est une matrice de rotation et q son quaternion équivalent, peut s'écrire comme suit :

$$\bar{p}_2 = q \otimes \bar{p}_1 \otimes q^{-1} \tag{B.3}$$

avec $\otimes$ représente l'opérateur de multiplication de quaternion, q^{-1} est l'inverse

de quaternion défini par $q^{-1} = [-q_1 \ -q_2 \ -q_3 \ q_4]^T$.

-Pour deux quaternions q_a et q_b, le produit : $q_a \otimes q_b$ est défini par :

$$q_a \otimes q_b \triangleq \mathcal{L}(q_a)q_b = \mathcal{R}(q_b)q_a \tag{B.4}$$

avec :

$$\mathcal{L}(q) = \begin{bmatrix} q_4 & -q_3 & q_2 & q_1 \\ q_3 & q_4 & -q_1 & q_2 \\ -q_2 & q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \quad \mathcal{R}(q) = \begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix}$$

-Nous avons également les propriétés suivantes :

$$\begin{aligned} &\mathcal{L}(q_a)\mathcal{R}(q_b) = \mathcal{R}(q_b)\mathcal{L}(q_a) \\ &q_b^T\mathcal{L}(q_a)^T = q_b^T\mathcal{R}(q_a)^T \\ &\mathcal{L}(q^{-1}) = \mathcal{L}(q)^T \\ &\mathcal{R}(q^{-1}) = \mathcal{R}(q)^T \end{aligned}$$

-Pour plus de détails, le lecteur intéressé est référé à [124].

C Détermination des *BBAs* avec la transformation de probabilité pignistique inverse

Pour trouver le *BBA* de chaque détection fournie par le processus de reconnaissance, nous avons utilisé la méthode décrite dans [77]. La caméra fournit une liste de détection des piétons en utilisant le détecteur d'objets *ACF*. La première étape consiste à s'assurer que la sortie de l'algorithme *ACF* fournit des fonctions des probabilités bayésiennes. Ensuite, il faut utiliser la transformation de probabilité pignistique inverse proposée par *Sudano* [154] pour déterminer les *BBAs*.

Supposons que P_{rec} est la fonction de probabilité donnée par le processus de reconnaissance et p_{ped} est la probabilité de reconnaissance :

$$P_{rec}(P_{cr}) = p_{ped} \qquad P_{rec}(NP_{cr}) = 1 - p_{ped} \tag{C.1}$$

TABLE C.1: Détermination des *BBAs*

2^{Ω_c}	P_{rec}	$m^{\Omega_c}_{cr,C,k}$ $(p_{ped} \leqslant \frac{1}{2})$	$m^{\Omega_c}_{cr,C,k}$ $(p_{ped} \geqslant \frac{1}{2})$
$\oslash$	•	0	0
$\{P_{cr}\}$	p_{ped}	0	$2p_{ped}$-1
$\{NP_{cr}\}$	$1 - p_{ped}$	1-$2p_{ped}$	0
Ω_c	•	$2p_{ped}$	2-$2p_{ped}$

Table C.1 décrit le calcul des *BBAs* pour le processus de reconnaissance. La première colonne présente les sous-ensembles de Ω_c tandis que la seconde montre les distributions des probabilités initiales fournies par le processus de reconnaissance. Les troisième et quatrième colonnes sont les *BBAs* calculées pour les deux cas : $p_{ped} \leqslant \frac{1}{2}$ (cas de Non-Piéton) et $p_{ped} \geqslant \frac{1}{2}$ (cas de détection de piéton).