



## "Conception d'un senseur intégré multimodal pour l'observation des routes"

Duflot, Alexis

### ABSTRACT

L'essor de l'intelligence artificielle, porté par le développement des systèmes embarqués, permet l'analyse de situations réelles de plus en plus complexes. Dans le domaine de l'observation des routes, l'étude multimodale de l'interaction entre les véhicules et les piétons, nécessite de définir une stratégie de détection, de classification et enfin de suivi de cibles multiples, sur base de la prise de données de plusieurs capteurs, pouvant être hétérogènes. Afin d'enregistrer des situations d'interaction entre des piétons et des véhicules, la conception d'un dispositif composé d'une caméra et d'un radar est réalisée. La détection et la classification des usagers de la route se fait à l'aide d'algorithmes tiny-YOLO, entraînés avec les données capturées sur le lieu d'étude. Enfin, la fusion des données et le suivi des cibles sont rendus possible par un filtre probabiliste d'association de données, dérivé du filtre de Kalman. Les données prises sur le lieu d'étude favorisent un entraînement efficace des réseaux tiny-YOLO. Lors de l'étape de détection et de classification, une précision de 88.1 % et 88.2 % est atteinte avec la caméra et le radar, respectivement. La fusion des données sur base d'une estimation de la distance des objets par la caméra, est une limitation majeure de la performance du suivi des cibles. Malgré des résultats de fusion des données et de suivi des cibles encourageants, l'algorithme implémenté ne permet pas d'étudier de manière fiable l'interaction entre les véhicules et les piétons. L'optimisation d...

### CITE THIS VERSION

Duflot, Alexis. *Conception d'un senseur intégré multimodal pour l'observation des routes*. Ecole polytechnique de Louvain, Université catholique de Louvain, 2020. Prom. : Craeye, Christophe ; Macq, Benoît. <http://hdl.handle.net/2078.1/thesis:23057>

Le dépôt institutionnel DIAL est destiné au dépôt et à la diffusion de documents scientifiques émanants des membres de l'UCLouvain. Toute utilisation de ce document à des fin lucratives ou commerciales est strictement interdite. L'utilisateur s'engage à respecter les droits d'auteur lié à ce document, principalement le droit à l'intégrité de l'oeuvre et le droit à la paternité. La politique complète de copyright est disponible sur la page [Copyright policy](#)

DIAL is an institutional repository for the deposit and dissemination of scientific documents from UCLouvain members. Usage of this document for profit or commercial purposes is strictly prohibited. User agrees to respect copyright about this document, mainly text integrity and source mention. Full content of copyright policy is available at [Copyright policy](#)

École polytechnique de Louvain

# Conception d'un senseur intégré multimodal pour l'observation des routes

Auteur: **Alexis DUFLOT**

Promoteurs: **Benoît MACQ, Christophe CRAEYE**

Lecteur: **Dani MANJAH**

Année académique 2018–2019

Master [120] : ingénieur civil électromécanicien



# Preface

Ce travail de fin d'études vient compléter ma formation d'ingénieur civil électromécanicien à l'École Polytechnique de Louvain. C'est aussi pour moi l'aboutissement de mon parcours dans l'enseignement supérieur, et le début d'une nouvelle étape dans ma vie professionnelle.

Tout au long de mes études, j'ai eu l'occasion de me spécialiser au travers de cours de mécanique et d'électronique. Il m'a cependant parut important de développer mes compétences en informatique et en intelligence artificielle, car de plus en plus de projets d'ingénierie incluent une forte composante numérique. Par ailleurs, l'utilisation de plusieurs capteurs est très courante en robotique, et m'a passionné depuis le projet en mécatronique réalisé durant ma première année de Master.

L'aboutissement de ce projet aura nécessité beaucoup d'effort et d'implication. La manipulation de jeux de données volumineux, l'entraînement de réseaux de neurones artificiels ainsi que l'optimisation de paramètres sont des processus nécessitant du temps et de la rigueur. La compréhension du fonctionnement des capteurs, de l'architecture des réseaux de neurones et des principes de fusion des données, demandent quand à eux des efforts de compréhension, et de la persévérance.

Malgré des résultats finaux un peu en dessous de mes ambitions, ce travail de fin d'études a plus que dépassé mes attentes en terme de découverte et d'approfondissement des disciplines nécessaires à la conception d'un senseur intégré multimodal pour l'observation des routes.

Je tiens à remercier mes superviseurs, Christophe Craeye et Benoît Macq, qui m'ont guidé et soutenu tout au long de ce projet. Dani Manjah pour avoir accepté de relire mon travail et d'apporter un regard extérieur au projet. Je suis par ailleurs reconnaissant envers Thomas Feuillen et Thomas Peron pour leurs explications concernant le fonctionnement du radar, ainsi qu'envers Jean Léger pour son aide par rapport au traitement des données radar. Je remercie également Regis Lomba de m'avoir permis d'accéder au Makilab. Merci aussi à ma famille de m'avoir soutenu dans les moments difficiles, et à mon frère pour m'avoir laissé accéder à son matériel d'impression 3D.

*Louvain-la-Neuve, 12 janvier 2020*

Alexis Duflot



# Abstract

The rise of artificial intelligence, driven by the development of embedded systems, enables the analysis of real and increasingly complex situations. In the field of road observation, the multimodal study of pedestrians and vehicles interactions imposes to choose a detection and classification strategy as well as a multi-target tracking method, based on the synchronised data capture of potentially heterogeneous sensors.

In order to record the situations of interaction between pedestrians and vehicles, a device equipped with the appropriate camera and radar is created. Road users are detected and classified by means of tiny-YOLO algorithms, trained with data collected on the site under investigation. Data fusion and target tracking are made possible by a probabilistic data association filter, derived from the Kalman filter.

The data collected allow for an efficient training of the tiny-YOLO networks. During the detection and classification stages, an accuracy of 88.1 % and 88.2 % is achieved for the camera and the radar, respectively. The data fusion - which is based on an estimation of the distance by the camera - is a major limitation of the target tracking performance. Despite encouraging results of the data fusion and the target tracking, the implemented algorithm does not seem to be a reliable measure for the study of the interactions between pedestrians and vehicles.

The Kalman parameters optimisation with an unsupervised learning method is a promising way of improving the obtained results. Nevertheless, the design of a single neural network algorithm, working from the data sensors to target tracking, seems even more promising in the long term.

Key words : Embedded system, Camera, Radar, *Tiny-YOLO* algorithm, Kalman filter, Multimodal sensor, Multiple target tracking, Traffic analysis



# Résumé

L'essor de l'intelligence artificielle, porté par le développement des systèmes embarqués, permet l'analyse de situations réelles de plus en plus complexes. Dans le domaine de l'observation des routes, l'étude multimodale de l'interaction entre les véhicules et les piétons, nécessite de définir une stratégie de détection, de classification et enfin de suivi de cibles multiples, sur base de la prise des données de plusieurs capteurs, pouvant être hétérogènes. Afin d'enregistrer des situations d'interaction entre des piétons et des véhicules, la conception d'un dispositif composé d'une caméra et d'un radar est réalisée. La détection et la classification des usagers de la route se fait à l'aide d'algorithme *tiny-YOLO*, entraînés avec les données capturées sur le lieu d'étude. Enfin, la fusion des données et le suivi des cibles sont rendus possible par un filtre probabiliste d'association de données, dérivé du filtre de Kalman.

Les données prises sur le lieu d'étude favorisent un entraînement efficace des réseaux *tiny-YOLO*. Lors de l'étape de détection et de classification, une précision de 88.1 % et 88.2 % est atteinte avec la caméra et le radar, respectivement. La fusion des données sur base d'une estimation de la distance des objets par la caméra, est une limitation majeure de la performance du suivi des cibles. Malgré des résultats de fusion des données et de suivi des cibles encourageants, l'algorithme implémenté ne permet pas d'étudier de manière fiable l'interaction entre les véhicules et les piétons.

L'optimisation des paramètres du filtre de Kalman par une méthode d'entraînement non supervisée représente une piste d'amélioration prometteuse. La conception d'un algorithme basée sur un seul réseau de neurones artificiels, partant des données des capteurs et permettant le suivi des cibles, paraît pourtant être une méthode plus prometteuse à long terme.

Mots clefs : Système embarqué, Caméra, Radar, Algorithme *tiny-YOLO*, Filtre de Kalman, Senseur multimodal, Suivi de cibles multiples, Analyse du trafic



# Table des matières

<b>Preface</b>	i
<b>Abstract</b>	iii
<b>Table des figures</b>	xi
<b>Liste des tableaux</b>	xiii
<b>Introduction</b>	1
<b>1 Description du dispositif de capture des données</b>	3
1.1 Spécifications techniques de la caméra . . . . .	4
1.2 Spécifications techniques du radar . . . . .	4
1.2.1 Principe de fonctionnement du radar . . . . .	6
1.2.2 Choix des paramètres du radar . . . . .	15
1.3 Description du micro-ordinateur embarqué . . . . .	16
1.4 Alimentation et connexion des modules . . . . .	17
1.5 Conception d'un boîtier pour le dispositif de capture . . . . .	19
<b>2 Algorithme de capture des données</b>	23
2.1 Acquisition et conversion des données radar . . . . .	23
2.2 Acquisition des données vidéo . . . . .	25
2.3 Synchronisation du radar et de la caméra . . . . .	26
2.3.1 Calcul de la latence et du délai des capteurs . . . . .	27
2.4 Interface de contrôle de la capture de données . . . . .	28
2.5 Capture des données sur le lieu d'étude . . . . .	29
<b>3 Aspect théorique des méthodes de détection et classification des objets</b>	31
3.1 Critères d'évaluation des algorithmes de classification . . . . .	31
3.2 Choix de la méthode de détection et de classification des objets . . . . .	33
3.2.1 Détecteurs en deux étapes . . . . .	34
3.2.2 Détecteurs en une étape . . . . .	35
3.3 <i>You Only Look Once</i> : principe de fonctionnement de l'algorithme . . . . .	36
3.3.1 Principe de fonctionnement de la détection et de la classification . . . . .	37
3.3.2 Architecture du réseau de neurones . . . . .	39

## Table des matières

---

3.3.3	Définition de la fonction de pertes . . . . .	41
3.3.4	Une version optimisée pour la vitesse : <i>tiny-YOLO</i> . . . . .	42
<b>4</b>	<b>Entraînement du réseau de détection et de classification des objets</b>	<b>43</b>
4.1	Stratégie d'entraînement avec les données vidéo . . . . .	43
4.1.1	Élaboration des jeux de données d'entraînement . . . . .	43
4.1.2	Choix des paramètres d'entraînement . . . . .	47
4.2	Analyse des résultats de détection pour la vidéo . . . . .	50
4.2.1	Entraînement avec les images du jeu de données <i>COCO</i> . . . . .	50
4.2.2	Entraînement avec les images du jeu de données du cas d'étude . . . . .	52
4.3	Optimisation des paramètres de seuil pour la vidéo . . . . .	55
4.3.1	Définition des paramètres de seuil . . . . .	55
4.3.2	Influence des paramètres de seuil . . . . .	57
4.3.3	Choix des paramètres de seuil . . . . .	59
4.4	Stratégie d'entraînement avec les données radar . . . . .	61
4.4.1	Choix des paramètres d'entraînement . . . . .	62
4.5	Analyse des résultats de détection pour le radar . . . . .	62
4.6	Optimisation du paramètre de seuil pour les détections radar . . . . .	64
<b>5</b>	<b>Stratégie de fusion des données et de suivi des cibles</b>	<b>67</b>
5.1	Définition de l'état des objets . . . . .	67
5.2	Description théoriques des étapes du filtre de Kalman . . . . .	68
5.2.1	Étape de prédiction . . . . .	69
5.2.2	Étape d'association . . . . .	70
5.2.3	Étape d'actualisation . . . . .	71
5.2.4	Étape de filtrage des pistes . . . . .	73
5.3	Définition des matrices de covariance et des paramètres de seuil . . . . .	73
5.3.1	Erreur intrinsèque des capteurs . . . . .	74
5.3.2	Autres sources d'erreur . . . . .	74
5.3.3	Erreur de prédiction du modèle . . . . .	75
5.3.4	Valeurs des paramètres de seuils . . . . .	75
5.3.5	Estimation de la distance des objets . . . . .	75
5.4	Résultats préliminaires . . . . .	76
5.4.1	Validation des mesures des deux capteurs . . . . .	77
5.4.2	Suivi d'une cible unique . . . . .	78
5.4.3	Association des pistes lors du suivi de cibles multiples . . . . .	79
5.4.4	Influence des paramètres sur les résultats . . . . .	80
<b>6</b>	<b>Améliorations de la précision du filtre probabiliste d'association</b>	<b>81</b>
6.1	Rectification du déséquilibre des composantes en quadrature du radar . . . . .	81
6.2	Prise en compte de la classe lors de l'étape d'association . . . . .	82
6.3	Détection des situations d'obstruction . . . . .	83
6.4	Association d'une unique détection par piste . . . . .	83

---

**Table des matières**

6.5	Imposition d'une distance minimum pour la création d'une piste . . . . .	84
6.6	Filtrage des pistes à l'affichage . . . . .	85
6.7	Discussion des résultats . . . . .	85
<b>7</b>	<b>Analyse statistique du lieu d'étude</b>	<b>87</b>
7.1	Comptage des véhicules et des piétons . . . . .	87
7.2	Détection des situations impliquant des véhicules et des piétons . . . . .	88
7.3	Discussion des résultats . . . . .	88
<b>8</b>	<b>Améliorations proposées</b>	<b>91</b>
8.1	Synchronisation physique des deux capteurs . . . . .	91
8.2	Amélioration de la détection et de la classification . . . . .	92
8.3	Fusion des données et suivi des cibles multiples . . . . .	92
8.3.1	Utilisation des angles mesurés par le radar . . . . .	92
8.3.2	Optimisation par apprentissage des paramètres du filtre de Kalman . . .	93
8.3.3	Faire de la classe des objets une variable d'état des pistes . . . . .	94
8.4	Proposition d'une stratégie alternative . . . . .	94
8.5	Discussion de l'impact potentiel de la technologie visée . . . . .	95
	<b>Conclusion</b>	<b>97</b>
<b>A</b>	<b>Annexes</b>	<b>99</b>
A.1	Influence des paramètres de seuil sur le score-F1 . . . . .	99
A.2	Optimisation des paramètres de seuil par intervalles . . . . .	101
A.3	Interactions sur le lieu d'étude . . . . .	103
	<b>Bibliographie</b>	<b>107</b>



# Table des figures

1.1	Lieu d'étude de l'interaction des véhicules et des piétons. . . . .	3
1.2	Modulation du signal émis par rampes de fréquences. . . . .	6
1.3	Disposition des antennes radar. . . . .	7
1.4	Exemple de graphique vitesse-distance avec une cible de grande taille. . . . .	12
1.5	Schéma d'alimentation et de connexion des modules. . . . .	18
1.6	Fixation des modules dans le boîtier du dispositif de capture. . . . .	19
1.7	Système de refroidissement et interrupteurs du dispositif. . . . .	20
1.8	Vue du dispositif dans son ensemble. . . . .	21
2.1	Exemple de pré-traitement des données radar . . . . .	25
2.2	Effet de l'asymétrie des composantes en phase et en quadrature . . . . .	26
2.3	Mesure de la latence et du délai des capteurs. . . . .	27
2.4	Architecture logicielle de l'algorithme de capture des données . . . . .	29
3.1	Calcul de la précision moyenne par interpolation de la courbe rappel-précision	33
3.2	Définition des coordonnées de position des boîtes détectées. . . . .	38
3.3	Représentation simplifiée de l'architecture du réseau <i>YOLOv3</i> . . . . .	40
4.1	Un objet représentatifs de chaque classe . . . . .	44
4.2	Représentation des <i>anchor boxes</i> pour l'entraînement du réseau <i>Yolo</i> . . . . .	49
4.3	Évolution du <i>mAP</i> lors de l'entraînement avec le jeu de données <i>COCO</i> . . . . .	51
4.4	Évolution des pertes moyennes lors de l'entraînement avec le jeu de données <i>COCO</i> . . . . .	51
4.5	Évolution de la précision et du rappel lors de l'entraînement avec le jeu de données <i>COCO</i> . . . . .	52
4.6	Évolution du score F1 lors de l'entraînement avec le jeu de données <i>COCO</i> . . . . .	52
4.7	Évolution du <i>mAP</i> lors de l'entraînement avec le jeu de données du cas d'étude. . . . .	53
4.8	Évolution des pertes moyennes lors de l'entraînement avec le jeu de données du cas d'étude . . . . .	53
4.9	Évolution de la précision et du rappel lors de l'entraînement avec le jeu de données du cas d'étude . . . . .	54
4.10	Évolution du score F1 lors de l'entraînement avec le jeu de données du cas d'étude	54
4.11	Situations de recouvrement des boîtes à corriger . . . . .	56
4.12	Influence des valeurs d' <i>indices de Jaccard</i> sur la précision et le rappel . . . . .	58

## Table des figures

---

4.13 Influence des valeurs d' <i>indices de Jaccard</i> et des valeurs d'indices de confiance sur le score-F1 . . . . .	58
4.14 Valeurs possibles du score-F1 en fonction du seuil de détection . . . . .	60
4.15 Valeurs possibles du score-F1 sur un intervalle réduit du seuil de détection . . . . .	60
4.16 Évolution du <i>mAP</i> et des pertes moyennes lors de l'entraînement avec les données radar . . . . .	63
4.17 Évolution de la précision et du rappel lors de l'entraînement avec les données radar . . . . .	63
4.18 Évolution du score F1 lors de l'entraînement avec les données radar . . . . .	64
4.19 Évolution de la précision en fonction du rappel . . . . .	65
5.1 Zone d'intérêt du carrefour étudié . . . . .	77
5.2 Suivie de la trajectoire d'un piéton venant de la route secondaire droite . . . . .	78
5.3 Suivie d'une voiture avec et sans l'utilisation des données radar . . . . .	79
6.1 Représentation de la zone interdite . . . . .	82
6.2 Calcul de la zone cachée par un objet . . . . .	83
6.3 Cas d'une obstruction détectée lors du passage de deux voitures . . . . .	84
8.1 Évolution de l'information entre chaque étape de l'algorithme implémenté . . . . .	94
8.2 Exemple de l'architecture d'un réseau de neurones, partant de l'information des capteurs au suivi des pistes . . . . .	95
A.1 Évolution de score-F1 en fonction du paramètre d'IoU objet unique . . . . .	99
A.2 Évolution de score-F1 en fonction de l'influence accordée aux fenêtres . . . . .	100
A.3 Influence des paramètres de seuil sur le score-F1 . . . . .	101
A.4 Influence des paramètres d'indices de Jaccard sur le score-F1 . . . . .	102

# Liste des tableaux

1.1	Spécifications techniques de la caméra . . . . .	5
1.2	Spécifications techniques du radar . . . . .	5
1.3	Paramètres du radar et limitations de la vitesse et de la distance. . . . .	15
1.4	Spécifications techniques du micro-ordinateur. . . . .	16
2.1	Valeurs moyennes des latences et du délai des capteurs . . . . .	28
2.2	Statistiques de capture des données . . . . .	30
4.1	Nombre d'objet de chaque classe dans le jeu de données d'entraînement pour la caméra . . . . .	46
4.2	Paramètres de simulation lors de l'optimisation des seuils . . . . .	57
4.3	Résultats de l'optimisation des paramètres de détection pour la caméra . . . . .	61
4.4	Paramètres de détection des objets avec le radar . . . . .	65
5.1	Définition des paramètres des pistes . . . . .	68
5.2	Variances des erreurs sur la valeur des variables d'état mesurées du système . . . . .	75
5.3	Valeurs initiales des paramètres du filtre de Kalman . . . . .	76
5.4	Hauteur moyenne des objets pour chaque classe . . . . .	76
7.1	Nombre d'objets détectés, et nombre d'objets passant réellement devant le dispositif. . . . .	87
A.1	Résultats de l'analyse de la distance entre les usagers vulnérables et les véhicules. . . . .	103



# Introduction

En 2017, d'après un rapport de recherche de l’Institut Belge pour la Sécurité Routière (IBSR), 12 % des accidents de la route en Belgique impliquaient des piétons. Au sein d'une agglomération dense telle que Bruxelles, 25 % de ce type d'accident mène au décès du piéton, dans les 30 jours après l'évènement. D'autre part, les systèmes embarqués, ainsi que l'intelligence artificielle, sont toujours plus présents au sein de la société moderne. Cet essor offre la possibilité d'analyser des évènements se déroulant dans des environnements complexes. Afin d'appréhender cette complexité, les technologies fonctionnant avec plusieurs capteurs complémentaires tendent à devenir incontournables. Il est ainsi pertinent de concevoir un senseur multimodal, capable d'étudier les interactions entre les véhicules et les piétons.

La conception d'un senseur analysant l'interaction entre les différents usagers de la route, nécessite l'observation de situations typiques. Un dispositif de capture des données doit donc être conçu, et permettre d'enregistrer l'information accessible par les différents capteurs, de manière synchronisée. Une fois les données accessibles, la détection des objets dans le champ de capture du dispositif doit être réalisée. Il est alors possible de reconnaître les différents usagers de la route, en pratiquant une classification des détections. L'utilisation de différents capteurs, mène à la nécessité de fusionner les données, qui peuvent être hétérogènes si les senseurs ne mesurent pas les mêmes grandeurs. En effet, les grandeurs accessibles par le radar et la caméra ne sont pas du même type, ce qui représente un défi majeur dans la mise en place d'un tel système. Afin de pouvoir analyser les interactions entre les différents objets, il est indispensable de suivre les trajectoires des objets dans le temps. L'étude des interactions est donc intimement liée au suivi de cibles multiples.

L'étude multimodale de l'interaction entre les véhicules et les piétons, nécessite par conséquent de définir une stratégie de détection, de classification et enfin de suivi de cibles multiples, sur base de l'enregistrement synchronisé des données de plusieurs capteurs, pouvant être hétérogènes. Dans ce but, la conception d'un dispositif permettant la prise d'informations sur un lieu d'étude est réalisée. Le lieu d'étude choisi est un endroit de passage, fréquenté par différents types d'usagers, non loin des bâtiments de l'Université catholique de Louvain. Le dispositif se base sur deux capteurs différents, à savoir une caméra et un radar. À l'aide des données récoltées, un algorithme de détection et de classification des objets peut être implanté et testé. Un entraînement non supervisé est réalisé, afin d'optimiser les performances sur le lieu d'étude. L'information des deux capteurs à un instant donné, est ensuite fusionnée

## **Introduction**

---

et l'évolution de la trajectoire des objets dans le temps peut être suivie.

Par la conception d'un dispositif de capture de données et l'utilisation d'un algorithme d'intelligence artificielle, l'objectif premier réside dans la capacité à détecter et différencier les usagers empruntant la route. Dans un second temps, le suivi des cibles, ainsi que le calcul des trajectoires des différents objets est recherché. Finalement, il est ambitionné de réaliser l'étude statistique de la fréquentation du lieu d'étude, ainsi que l'évaluation du danger des situations observées par le dispositif.

Dans un premier temps, deux Chapitres sont consacrés respectivement à la description du dispositif de prise des données, et à l'implémentation de l'algorithme de capture, respectivement. Le Chapitre 3 décrit l'aspect théorique des méthodes de détection et de classification des objets, ainsi que le choix de la méthode utilisée dans le cadre de ce travail de fin d'études. La stratégie d'optimisation de l'algorithme choisi est exposée au Chapitre 4. La fusion des données des deux capteurs, ainsi que la méthode de suivi des cibles, définissent le fil conducteur des Chapitres 5 et 6. Enfin, une rapide analyse du lieu d'étude ainsi que les pistes d'amélioration du projet, sont exposées dans les Chapitres 7 et 8.

# 1 Description du dispositif de capture des données

Il est proposé d'exploiter les atouts combinés d'une caméra et d'un radar afin de calculer les trajectoires d'objets se déplaçant dans l'espace. Plus précisément, les objets sur lesquels l'étude se concentre sont les véhicules et les piétons interagissant sur une route et ses abords. La section de route choisie impose une gamme de distances de 0 m à 80 m (voir FIGURE 1.1), avec un intérêt particulier pour les interactions ayant lieu à environ 45 m du dispositif de capture. Les véhicules circulent dans le deux sens, et se déplacent généralement à moins de  $80 \text{ km h}^{-1}$  dans la zone, la limite légale étant de  $50 \text{ km h}^{-1}$ .

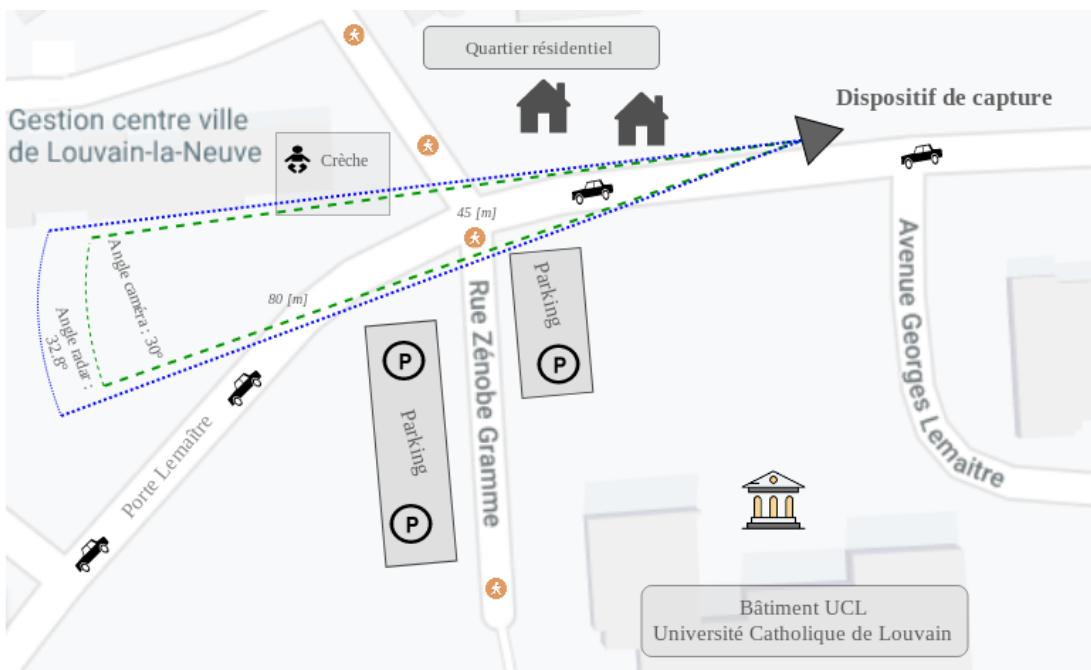


FIGURE 1.1 – Lieu d'étude de l'interaction des véhicules et des piétons. Le lieu se situe à Louvain-la-Neuve, avenue George Lemaître. La proximité d'une crèche, de l'université et d'un quartier résidentiel en font un endroit de passage, fréquenté par beaucoup de piétons et de véhicules. Aucun passage piéton n'existe cependant.

## Chapitre 1. Description du dispositif de capture des données

---

Afin de récolter des données exploitables, il est nécessaire de sélectionner un radar et une caméra avec des performances adaptées au cas d'étude. Trois aspects sont particulièrement importants. Premièrement la résolution et le rapport signal-sur-bruit des données doivent être suffisants pour détecter et reconnaître les objets dans la gamme de distances désirée. Deuxièmement, l'angle de capture doit permettre l'enregistrement des données sur la route et ses abords. Finalement, la fréquence de capture des données doit être suffisante pour qu'une hypothèse de vitesse constante entre les images soit valide. En considérant qu'un véhicule classique n'aura pas d'accélération supérieure à  $1.2 \times g \text{ m s}^{-2}$  [1][2] (avec  $g$  l'accélération de la pesanteur valant  $9.81 \text{ m s}^{-2}$ ) et en cherchant à avoir une variation de vitesse de  $1 \text{ km}^{-1}$  au maximum entre deux captures, il est nécessaire de faire au moins 11.8 prises de données par seconde.

Pour finir, l'objectif étant aussi de proposer un système à bas coût, il conviendra de choisir les différents modules dans une gamme de prix raisonnable.

Les exigences à remplir sont donc les suivantes :

- Une résolution suffisante pour identifier un piéton à environ 80 m
- Un angle de capture suffisant pour enregistrer la route et ses alentours (environ 30°)
- Une fréquence de capture d'au moins 12 images par seconde
- Un prix abordable

### 1.1 Spécifications techniques de la caméra

La caméra utilisée pour la prise des données vidéo se base sur le capteur d'images digital **AR0230** [3] en technologie CMOS. Le fabricant propose cette caméra sous la forme d'un module prêt à l'emploi [4]. Les paramètres utiles dans le cadre de l'étude sont repris dans le TABLEAU 1.1.

Le choix de la caméra est justifié par sa simplicité d'utilisation ainsi que par ses bonnes performances à moyenne distance, c'est-à-dire à des distances de l'ordre de 40 m. De plus, l'exposition automatique permet de travailler en extérieur malgré une large gamme de luminosités ambiantes.

En plus de présenter des performances rencontrant les objectifs de la prise de données, le capteur est proposé à un prix très abordable.

### 1.2 Spécifications techniques du radar

Le radar utilisé dans le cadre de l'étude est manufacturé par *RFbeam Microwave GmbH*, et permet la prise de données radar selon trois dimensions. L'appareil se charge de la conversion analogique à digitale des signaux, avec la sélection de 260 échantillons par rampe de fréquence. Comme expliqué dans la Section 1.2.1, et dans le but de détecter la vitesse et la distance des

## 1.2. Spécifications techniques du radar

Caméra ELP 1/3" CMOS AR0330	
Résolution	320 × 180, 320 × 240, 640 × 360, 640 × 480, 1280 × 720, 1920 × 1080
Format vidéo	H.264/MJPEG/YUY2(YUYV)
Images par secondes	30 fps, 25 fps, 20 fps, 10 fps, 5 fps
Exposition automatique	Oui
Interface	USB 2.0
Alimentation	5 [V] via USB / 150 [mA]
Lentille	12 [mm]
Angle d'ouverture	30 [°]
Distance focale	5695.8 pixels
Erreur moyenne de re-projection	0.78 pixels
Dimensions	38 × 38 × 32 [mm <sup>3</sup> ]
Prix	45 €

TABLEAU 1.1 – Spécifications techniques de la caméra [4][5]. Les paramètres de distance focale et d'erreur moyenne de re-projection ont été calculés dans le cadre d'un travail similaire réalisé en 2018 par Sébastien Cardinael et Arthur Malcourant [5], et sur base d'une calibration géométrique.

objets, le radar envoie des rampes de fréquence via son unique antenne émettrice. Trois antennes réceptrices se chargent de capter l'écho du signal émis. Les spécificités techniques complètes du dispositif ainsi que les instructions d'utilisation peuvent être trouvées sur le site du fabricant [6].

Radar K-MD2-RFB-01F-01	
Résolution	256 × 256 (vitesse × distance)
Format des données de sortie	Données brutes (signal démodulé), Graphique distance-vitesse (doppler), Cibles non filtrées, Cibles traquées
Images par secondes	20 [fps]
Gain de l'émetteur	24 [GHz]
Portée	0-250 [m], 1 [m] de résolution
Intervalle de vitesse détecté	±130 [km.h <sup>-1</sup> ], 1 [km.h <sup>-1</sup> ] de résolution
Angle d'ouverture en élévation	± 18.2 [°], 0.1 [°] de résolution
Angle d'ouverture en azimut	± 32.8 [°], 0.1 [°] de résolution
Interface	Ethernet ou Connections série
Alimentation	10-16 [V] / 600 [mA]
Dimensions	120 × 72 × 15 [mm <sup>3</sup> ]
Prix	1.336 €

TABLEAU 1.2 – Spécifications techniques du radar (sur base de la fiche technique [6]).

### 1.2.1 Principe de fonctionnement du radar

Le radar fonctionne sur le principe des radars à ondes continues modulées en fréquence. Les informations de vitesse, de distance, d'angle vertical et d'angle azimuthal peuvent être obtenues pour toutes les cibles dans le champ de détection, dans des gammes de résolutions propres au *hardware*.

Les explications qui suivent se basent principalement sur la thèse d'Elio Guerrero-Menéndez intitulée *Frequency-modulated continuous-wave radar in automotive applications* [7], et en particulier sur la section 2.2 : *Frequency-modulated continuous-wave radar*.

Le radar envoie des ondes modulées en fréquence qui vont se refléter sur les objets présents dans le champs de détection. Plus précisément, 257 rampes de fréquence modulent le signal envoyé pour chaque image (voir FIGURE 1.2). La première rampe est ignorée, car le saut à la fréquence porteuse occasionne des perturbations [8]. Ce sont donc 256 rampes, ou *chirps*, qui sont utilisées pour chaque image.

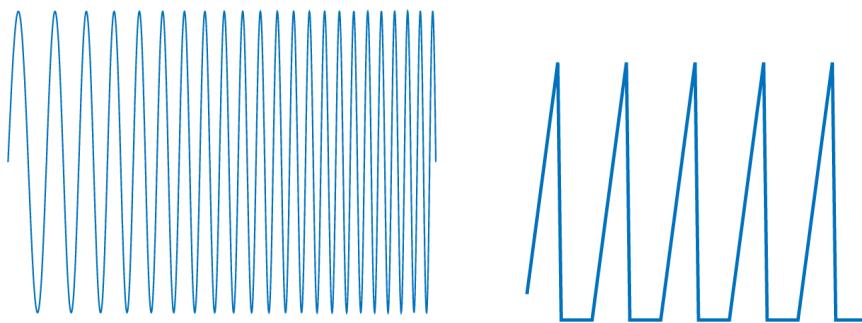


FIGURE 1.2 – Modulation du signal émis par rampes de fréquences (*chirps*). À gauche le signal émis, à droite 5 rampes de fréquence.

Lors de la réception de l'écho du signal émis, le radar opère une démodulation afin de retrouver le signal modulant, c'est-à-dire les rampes de fréquence. Du fait du temps de propagation des ondes jusqu'aux cibles, les rampes (et donc les fréquences) contenues dans le signal reçu sont décalées par rapport au signal émis. En superposition au délai du à la distance, le décalage associé à l'effet "Doppler" modifie les fréquences reçues. En prenant comme convention que la vitesse est positive dans le sens de propagation des ondes, un objet s'approchant va réfléchir des fréquences légèrement plus élevées, et donc engendrer un décalage des rampes similaire à une distance plus importante. En effet, la fréquence reçue est proportionnelle au rapport de la célérité de l'onde, avec la différence entre cette célérité et la vitesse de la cible. Dans le cas présent, la célérité de l'onde vaut la vitesse de la lumière. Il n'est cependant pas possible de retrouver à la fois la vitesse et la distance des objets, avec comme seule information le décalage des signaux émis et reçus.

Afin de résoudre l'ambiguïté sur la distance et la vitesse, la phase du signal reçu peut être étudiée. En effet, la vitesse de l'onde est très élevée car elle est égale à la vitesse de la lumière.

## 1.2. Spécifications techniques du radar

Dès lors, le délai temporel entre deux rampes de fréquences ne va quasiment pas changer au sein d'une image. En revanche, la longueur d'onde étant très petite, la phase entre deux *chirps* va changer proportionnellement à la vitesse de l'objet. De ce point de vue, il est donc avantageux d'avoir une petite longueur d'onde, et par conséquent une fréquence de porteuse élevée.

Grâce à la disposition particulière des antennes réceptrices (voir FIGURE 1.3), l'angle azimutal et l'angle d'élévation des cibles peuvent être calculés. De même que pour la vitesse, la phase du signal reçu renseigne sur la petite différence de parcours de l'onde en fonction de l'antenne réceptrice. La distance donne ensuite l'information d'angle d'incidence de l'onde par rapport au radar. Il convient cependant de garder à l'esprit que la différence de parcours entre les antennes peut être grande, et même supérieure à une longueur d'onde. Dès lors, il existe une ambiguïté sur les angles, car un angle trop important peut être compté comme un petit angle, si le déphasage dépasse légèrement 180°.

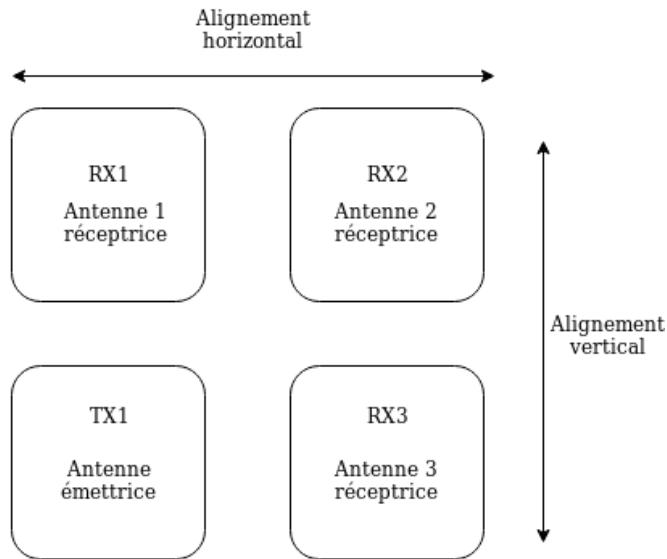


FIGURE 1.3 – Disposition des antennes radar. L'alignement horizontal permet de calculer l'angle azimutal et l'alignement vertical de calculer l'angle dans le plan vertical.

Le principe de fonctionnement du radar ayant été expliqué de manière qualitative, les équations mathématiques permettant d'obtenir le graphique distance-vitesse vont maintenant être introduites. Certains raisonnements sont repris de la thèse précédemment citée : *Frequency-modulated continuous-wave radar in automotive applications* [7]. La généralisation à plus d'une rampe de fréquence et plusieurs objets est reprise de l'article *A Low-Complexity FMCW Surveillance Radar Algorithm Using Two Random Beat Signals* [9]. Pour finir, l'article *Quantity over Quality : Dithered Quantization for Compressive Radar Systems* [10] a servi de base de compréhension.

La porteuse est un signal sinusoïdal de fréquence  $f_p$ , avec  $f_p$  valant environ 24 GHz. Le signal porteur  $s_p(t)$  s'écrit donc sous la forme :

## Chapitre 1. Description du dispositif de capture des données

---

$$s_p(t) = A_p \times \cos(\omega_p t + \phi_p)$$

où  $A_p$  est l'amplitude du signal,  $\omega_p = 2\pi f_p$  est la pulsation et  $\phi_p$  la phase au temps zéro.

D'autre part, la fonction mathématique décrivant les rampes de fréquence s'écrit sous la forme :

$$f_r(t) = f_0 + \frac{B}{T} \cdot t \quad t \in [0, T] \quad [Hz]$$

avec  $B$  la bande passante,  $T$  la période d'une rampe de fréquence et  $f_0$  la fréquence de départ des rampes.

Le signal modulé en fréquence  $s(t)$  est alors donné par :

$$s(t) = A_p \times \cos\left(2\pi \int_0^t (f_p + f_r(t')) dt + \phi_p\right)$$

Afin de simplifier les calculs, les signaux analytiques seront utilisés dans la suite du développement :

$$\bar{s}(t) = A_p \times \exp\left(2\pi i \int_0^t (f_p + f_r(t')) dt + i\phi_p\right) \quad (1.1)$$

Le signal reçu par une des antennes réceptrices,  $r(t)$ , est semblable au signal émis avec un délai temporel  $\tau$ . En négligeant le bruit, il s'écrit sous la forme :

$$\bar{r}(t) = \bar{s}(t - \tau) = A_p \times \exp\left(2\pi i \int_0^{t-\tau} (f_p + f_r(t')) dt + i\phi_p\right)$$

Afin de mettre en évidence le décalage de phase dû à l'effet doppler, il est nécessaire de prendre en compte plusieurs rampes de fréquence. En considérant  $M$  cibles,  $L$  rampes de fréquence et  $K$  antennes réceptrices, le signal reçu à l'antenne  $k$  pour la rampe  $l$  s'écrit :

$$\bar{r}_{l,k}(t) = \sum_{m=0}^{M-1} \tilde{a}_m \bar{s}(t - \tau_m) \times \exp(2\pi i f_D T l) \times \exp\left(\frac{2\pi i}{\lambda} \cdot d_s \cdot \sin(\theta_m)\right) \quad (1.2)$$

où  $f_D$  est le déphasage doppler,  $\lambda$  la longueur d'onde de la porteuse,  $d_s$  la distance entre l'antenne  $k$  et l'antenne de référence, et  $\theta_m$  l'angle d'incidence sur l'antenne  $k$ . Le paramètre  $\tilde{a}_m$  permet de modéliser le rapport entre les puissances émise et reçue.

Le premier facteur décrit le décalage temporel  $\tau_m$  comme dans le cas d'un seul *chirp*. Le

## 1.2. Spécifications techniques du radar

---

second facteur fait apparaître la fréquence doppler  $f_D$ , qui joue ici le rôle d'un déphasage entre les rampes de fréquences. En effet, le déphase entre deux rampes s'écrit :

$$\phi = 2\pi \cdot \frac{2d_m}{\lambda} \quad [rad]$$

avec  $d_m$  la différence de parcours jusqu'à la cible entre deux signaux émis et  $\lambda$  la longueur d'onde. La vitesse radiale de la cible,  $v_m$ , est liée à la distance  $d_m$  par :

$$d_m = T v_m \quad [m]$$

avec  $T$  la période entre deux *chirp*. En remplaçant la distance par la vitesse et en sachant que la vitesse de la lumière  $c$  peut s'écrire sous la forme  $c = \lambda f$ , le déphase s'exprime de la manière suivante :

$$\phi = 4\pi \cdot \frac{T v_m}{\lambda} = 2\pi T \cdot \frac{2f v_m}{c} = 2\pi T f_D \quad [rad] \quad (1.3)$$

La fréquence Doppler  $f_D$  est donc bien retrouvée sous forme d'un déphase :

$$f_D = \frac{2f v_m}{c} \quad [Hz] \quad (1.4)$$

Dans le cas d'un radar *FMCW*, l'effet Doppler s'exprime via d'un déphasage et non directement d'un décalage en fréquence. En réalité, un décalage fréquentiel est pris en compte dans le décalage temporel  $\tau$  du premier facteur de l'équation 1.2, venant se superposer à celui du temps de propagation. À noter qu'il n'est donc pas possible de faire la différence entre le décalage dû à l'effet Doppler et celui dû à la distance, sur base du premier terme de l'équation 1.2 uniquement.

Le troisième facteur de l'équation 1.2 fait apparaître l'angle de la cible par rapport au radar. En effet, de même que pour le déphase Doppler, une différence de parcours  $d_k$  va impliquer un déphasage  $\phi_k$  valant :

$$\phi_k = 2\pi \cdot \frac{d_k}{\lambda} \quad [rad]$$

La différence de parcourt  $d_k$  est simplement liée à l'angle d'incidence par l'équation :

$$d_k = d_s \cdot \sin(\theta_m) \quad [m]$$

Le radar utilisé est donc capable de calculer les angles azimutaux et verticaux grâce à ses trois

## Chapitre 1. Description du dispositif de capture des données

---

antennes réceptrices (voir FIGURE 1.3).

Après réception du signal  $\bar{r}_{l,k}(t)$ , ce dernier est démodulé en bande de base. Pour cela, il est multiplié par le conjugué du signal envoyé :  $\bar{s}^*(t)$ . Le signal conjugué du signal émis après résolution de l'intégrale dans l'équation 1.1 s'écrit :

$$\bar{s}^*(t) = A_p \times \exp\left(-2\pi i \left[(f_p + f_0)t + \frac{B}{T} \frac{t^2}{2}\right] - i\phi_p\right)$$

Le signal  $\bar{s}(t - \tau)$  présent dans l'expression de  $\bar{r}_{l,k}(t)$  (l'équation 1.2) peut être développé de la manière suivante :

$$\begin{aligned} \bar{s}(t - \tau) &= A_p \exp(2\pi i [(f_p + f_0)(t - \tau) + \frac{B}{T} \frac{(t - \tau)^2}{2}] + i\phi_p) \\ &= A_p \exp(2\pi i [(f_p + f_0)t + \frac{B}{T} \frac{t^2}{2} + i\phi_p]) \times \exp(-2\pi i [(f_p + f_0)\tau + \frac{B}{T} [\tau t - \frac{\tau^2}{2}]]) \\ &= \bar{s}(t) \times \exp(-2\pi i [(f_p + f_0)\tau + \frac{B}{T} [\tau t - \frac{\tau^2}{2}]]) \\ &= \bar{s}(t) \times \exp(-2\pi i [(f_p + f_0)\tau - \frac{B}{T} \frac{\tau^2}{2}]) \times \exp(-2\pi i \frac{B\tau}{T} t) \end{aligned}$$

Lorsque la bande relative du signal est étroite, le signal  $\bar{s}(t - \tau)$  peut donc être exprimé sous la forme du signal  $\bar{s}(t)$  multiplié par un gain indépendant du temps et par une composante qui varie au cours du temps.

Le signal démodulé  $\bar{y}_{l,k}(t)$  est donc égal à :

$$\bar{y}_{l,k}(t) = \bar{r}_{l,k}(t) \times \bar{s}^*(t) \quad (1.5)$$

$$= \sum_{m=0}^{M-1} \tilde{a}_m \bar{s}(t - \tau) \times \exp(2\pi i f_D T l) \times \exp\left(\frac{2\pi i}{\lambda} \cdot d_s \cdot \sin(\theta_m)\right) \times \bar{s}^*(t) \quad (1.6)$$

$$= \sum_{m=0}^{M-1} a_m \exp(-2\pi i f_{\tau,m} t) \times \exp(2\pi i f_D T l) \times \exp\left(\frac{2\pi i}{\lambda} d_s \sin(\theta_m)\right) \quad (1.7)$$

où  $a_m$  est le terme complexe qui prend en compte le gain statique apparaissant lors de la multiplication de  $\bar{s}(t)$  avec  $\bar{s}(t - \tau)$  et le gain  $\tilde{a}_m$ . Par définition, le terme  $f_{\tau,m}$  est égal à :

$$f_{\tau,m} = \frac{B\tau}{T} \quad [Hz] \quad (1.8)$$

et correspond au décalage en fréquence du au temps de propagation de l'onde, cette fois-ci sur le signal démodulé.

## 1.2. Spécifications techniques du radar

---

L'équation 1.7 peut être écrite sous la forme suivante :

$$\bar{y}_{l,k}(t) = \sum_{m=0}^{M-1} a_m \eta_m(t) v_m z_m \quad (1.9)$$

Grâce au convertisseur analogique à digital du radar, le signal donné par l'équation 1.9 est discrétisé avec une période  $T_s$ , en prenant N échantillons (N vaut 260 dans le cas présent) :

$$\bar{y}_{l,k}[n] = \bar{y}_{l,k}(n T_s) = \sum_{m=0}^{M-1} a_m \eta_m[n] v_m z_m \quad (1.10)$$

Afin de calculer la vitesse et la distance de chaque cible, il est possible d'arranger les données sous la forme d'une matrice de dimension  $n \times l$ ,  $n$  étant le nombre d'échantillons temporels et  $l$  le nombre de rampes de fréquence. En pratiquant la FFT (Fast Fourier Transform) sur les lignes de la matrice, les fréquences  $f_{\tau,m}$  et  $f_D$  apparaissent. Le graphique Doppler distance-vitesse obtenu est visible sur la FIGURE 1.4.

Ayant obtenu la fréquence  $f_D$ , la vitesse radiale peut être calculée à l'aide de l'équation 1.4. La distance est calculée sur base de la fréquence  $f_{\tau,m}$  et du délai temporel  $\tau$ . En effet, le délai temporel est lié à un décalage en fréquence qui résulte de la superposition de l'effet Doppler et du phénomène de battement dû à la distance :

$$f_{\tau,m} = f_{\tau_d,m} + f_{\tau_D,m} = \frac{B\tau_d}{T} + f_D \quad [Hz]$$

Connaissant  $f_D$  et  $f_{\tau,m}$ , le délai causé par la distance à parcourir par l'onde,  $\tau_d$ , peut être retrouvé et la distance est donnée par :

$$R = \frac{c\tau_d}{2} \quad [m] \quad (1.11)$$

Afin de régler les paramètres du radar, il est nécessaire de les lier à la résolution et aux valeurs maximales de la vitesse et de la distance.

Comme exprimé dans [7], la résolution en distance est limitée par le fait que les ondes envoyées sont finies dans le temps. Idéalement, un sinus dans le domaine temporel est équivalent à un pic de Dirac dans le domaine fréquentiel, à la fréquence fondamentale du sinus. En revanche, un sinus cardinal apparaît dans le domaine fréquentiel si le sinus est finit dans le temps. La largeur du lobe principal est déterminée par la période de temps durant laquelle le sinus est échantillonné, donc par la différence de temps entre le premier et le dernier échantillon. En effet, les premiers zéros du sinus cardinal se trouvent à une fréquence inversement proportionnelle à la durée de la fenêtre d'échantillonnage du sinus. La plus petite différence de fréquence observable vaut dès lors cette même fréquence de fenêtre de capture. Autrement

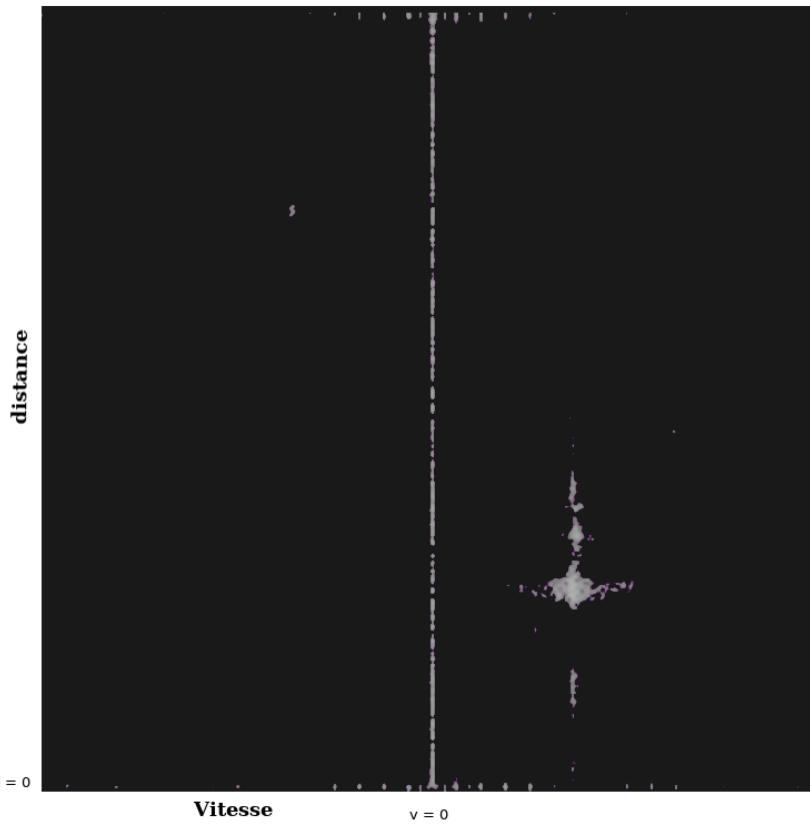


FIGURE 1.4 – Exemple de graphique vitesse-distance avec une cible de grande taille (un camion). La distance est représentée en ordonnée et la vitesse en abscisse. Une filtration par soustraction du fond moyen de bruit a été pratiquée.

dit, il ne sera pas possible de faire la différence entre deux fréquences trop proches, car les pics dans le domaine fréquentiel se toucheront. En combinant l'équation 1.11 et l'équation 1.8 il vient :

$$\Delta R = \frac{c\tau_{d,min}}{2} = \frac{c\Delta f T}{2B} = \frac{c}{2B} \times \frac{T}{T_f} \quad [m] \quad (1.12)$$

où  $T_f$  est la période de la fenêtre de capture du sinus. Le terme  $\tau_{d,min}$  donne en fait le plus petit décalage temporel perceptible de manière non ambiguë.

Dans le cas du radar utilisé, la formule donnée dans la fiche technique [8] est :

$$\Delta R = 150 \cdot 10^6 \cdot \left( \frac{N+M}{NB} \right) \quad [m]$$

## 1.2. Spécifications techniques du radar

---

avec  $N$  le nombre d'échantillons utilisés et  $M$  le nombre d'échantillons ignorés. Il convient ici de se souvenir que  $c \approx 2 \cdot 150 \cdot 10^6 \text{ m s}^{-1}$ . Pour que l'équation 1.2.1 soit identique à celle donnée par la fabricant du radar, il est nécessaire qde prendre en compte que certains échantillons sont ignorés :

$$\frac{T}{T_f} = \frac{N+M}{N} \neq 1$$

La formule précédemment dérivée et l'équation donnée dans la fiche technique du radar donnent donc une valeur de résolution en distance identique.

Concernant la distance maximale mesurable, la fiche technique donne simplement la distance maximale possible en considérant le nombre d'échantillons et la résolution :

$$R_{max} = (N - 1)\Delta R \quad [m]$$

En théorie, le critère de Nyquist doit être respecté lors de l'échantillonnage et la différence de fréquence maximale  $\Delta f_{max}$  détectable vaut la moitié de la fréquence d'échantillonage  $f_s$  :

$$R_{max} = \frac{c f_s T}{4B} \quad [m]$$

Pour la vitesse, le critère de Nyquist détermine encore une fois la valeur maximale mesurable. En effet, il est possible de voir chaque *chirp* comme un échantillon de la phase dans le temps. Par conséquent, le décalage de phase maximal détectable vaut :

$$\phi_{max} = \frac{\lambda}{4} \quad [rad]$$

Le facteur 4 vient de la multiplication du facteur 2 du critère de Nyquist, et du fait que l'onde fait un aller-retour. Autrement dit, « un changement de position de la cible de  $\lambda/4$  résultera en un changement de  $\lambda/2$  dans le module du radar » d'après [7].

Dès lors :

$$v_{max} = \frac{\lambda}{4T} \quad [m.s^{-1}]$$

À noter que si la période  $T$  diminue, la vitesse maximale détectable augmente, mais la distance maximale diminue.

En reprenant l'équation 1.3, le décalage de phase peut être écrit sous la forme :

## Chapitre 1. Description du dispositif de capture des données

---

$$\Delta\phi = \frac{4\pi v_r T}{\lambda} \quad [rad]$$

où  $v_r = v_m$  est la vitesse radiale de l'objet m dans le référentiel du radar.

Le décalage de phase est limité par le nombre de rampes par images  $N_p$  :

$$\Delta\phi_{max} = \frac{2\pi}{N_p} \quad [rad]$$

Et ainsi :

$$\Delta v_r = \frac{\lambda}{2N_p T} \quad [m.s^{-1}]$$

La fiche technique du radar donne la formule suivante :

$$\Delta v_r = \frac{3.6 \cdot \lambda \cdot clk}{2N(S_{clk} \cdot (N + M) + delay)} \quad [km.h^{-1}]$$

où  $clk$  est la fréquence d'horloge,  $S_{clk}$  le nombre de périodes d'horloge par échantillon, et  $delay$  le temps de délai entre les rampes, exprimé en fréquence d'horloge. Le facteur 3.6 vient de la conversion de mètre par seconde à kilomètre par heure. Les formules sont cohérentes car :

$$T = \frac{(S_{clk}(N + M) + delay)}{clk}$$

En effet, le nombre total de périodes d'horloge d'une rampe vaut le nombre d'échantillons multiplié par le nombre de périodes d'horloge par échantillon, auquel vient s'ajouter le délai. Pour trouver la période en seconde, il suffit alors de diviser par la fréquence d'horloge.

La vitesse maximale est donnée dans la fiche technique sous la forme :

$$v_{r,max} = \Delta v_r \left( \frac{N}{2} - 1 \right) \quad [m.s^{-1}] \quad (1.13)$$

La division par 2 vient du fait que les vitesses positives et les vitesses négatives sont prises en compte.

### 1.2.2 Choix des paramètres du radar

En considérant que les objets se déplacent entre  $0 \text{ km h}^{-1}$  et  $80 \text{ km h}^{-1}$ , et sur base de l'équation 1.13, le délai entre rampes à imposer au radar vaut 2214 fréquences d'horloge, avec la fréquence du processeur valant environ 38,5 MHz. Les autres paramètres de l'équation 1.13 sont la longueur d'onde de la porteuse, le nombre de périodes d'horloge par échantillon, et le nombre d'échantillons. Ces paramètres sont imposés par le radar et sont repris dans la TABLEAU 1.3. La résolution en vitesse vaut dès lors  $0.630 \text{ kmh}^{-1}$ .

Afin d'avoir une bonne résolution à la distance d'intérêt, c'est-à-dire à 50 m, la bande passante est fixée à 554 MHz et la fréquence initiale de rampe vaut environ 23.8 GHz. La distance maximale des objets détectables est dès lors de 70 m et la résolution de 0.27 m environ.

Grandeur	Notation	Valeur
Bandé passante de rampe	B [MHz]	554
Bandé passante d'échantillonage	$B_s$ [MHz]	545.5
Fréquence de départ	$f_0$ [GHz]	23.8
Échantillons utilisés	N	256
Échantillons ignorés	M	4
Longueur d'onde	$\lambda$ [m]	0.012426
Fréquence d'horloge	clk [MHz]	38,5
Période d'horloge par échantillon	$S_{clk}$	12
Délai des rampes	delay [ $clk^{-1}$ ]	2214
Résolution en distance	$\Delta R$ [m]	0.274
Distance maximale	$R_{max}$ [m]	70.122
Résolution en vitesse	$\Delta v_r$ [ $km.h^{-1}$ ]	0.630
Vitesse absolue maximale	$v_{r,max}$ [ $km.h^{-1}$ ]	80.014

TABLEAU 1.3 – Paramètres du radar et limitations de la vitesse et de la distance.

Le choix du radar est justifié par ses bonnes performances techniques pour les distances et les vitesses d'intérêt. La capacité à calculer la vitesse, la distance et les angles, ouvre le champ des possibilités. L'angle d'ouverture repris au TABLEAU 1.2 s'accorde bien avec celui de la caméra (TABLEAU 1.1), ce qui permettra la capture de scènes similaires avec les deux capteurs. Le prix du radar est cependant significatif et représente une part très importante du coût du dispositif, car il est environ 30 fois plus cher que la caméra.

Les paramètres intrinsèques et choisis du radar ainsi que les limitations des valeurs de vitesse et de distance sont repris dans le TABLEAU 1.3.

### 1.3 Description du micro-ordinateur embarqué

Le choix du micro-ordinateur embarqué s'est porté sur le *Jetson-Nano*® [11]. La version *Nano* du Jetson est une adaptation des versions *Xavier*® et *TX2*® pour les systèmes embarqués. L'ordinateur est fabriqué par Nvidia®, qui a concentré ses efforts sur la consommation d'énergie et le coût. Le *Nano* a donc une consommation d'énergie très faible (entre 5 W et 30 W) et des dimensions compactes pour le prix très abordable de 110 € environ. Les performances de l'ordinateur sont dopées par une carte graphique Nvidia *Maxwell*®, ce qui permet de développer des applications temps-réel d'intelligence artificielle. Les caractéristiques techniques du *Jetson Nano*® sont reprises dans le TABLEAU 1.4.

<b>Nvidia Jetson Nano</b>	
Carte graphique (GPU)	Arch. Maxwell 128 cœurs CUDA
Processeur (CPU)	Quad-core ARM Cortex-A57 MPCore
Mémoire	4 Go 64-bit LPDDR4
Mémoire de la carte SD	128 Go
Stockage	16 Go eMMC 5.1 Flash
Puissance de calcul	472 GFlops
Interfaces	Logement micro-SD, baie 40 broches multi-usage, port micro-USB, port Ethernet, 4 x port USB3.0, sortie HDMI, connecteur DisplayPort, connecteur caméra MIPI CSI, connecteur DC Barrel 5V
Alimentation	5V/2A via le connecteur micro-USB, 5V/4A via le connecteur DC Barrel, 5V/6A via la baie 40 broches
Dimensions*	110 x 90 x 48 [mm <sup>3</sup> ]
Prix**	178 €

\* En comprenant le boîtier et le ventilateur

\*\* Le prix comprend tous les modules, c'est-à-dire le boîtier, l'extension WIFI, le ventilateur, la carte SD et l'ordinateur lui-même

TABLEAU 1.4 – Spécifications techniques du micro-ordinateur (sur base de [11]).

À l'inverse d'un *Raspberry-PI*® [12], le *Jetson Nano*® n'a pas d'interface WIFI. Il est cependant très facile d'ajouter une carte WIFI *Intel 8265NGW*® [13] et deux antennes<sup>1</sup>. Il a été décidé d'ajouter également un petit ventilateur pour éviter l'échauffement, et un boîtier en plastique pour faciliter la fixation dans le dispositif de capture et protéger l'ordinateur.

Le système d'exploitation du *Jetson*® est une version compatible de *Linux Ubuntu 18.04* qu'il faut installer sur une carte SD (*Secure Digital*). L'ordinateur étant développé directement par Nvidia®, les pilotes de la carte graphique et les outils d'accélération sont installés par défaut. Afin de pourvoir enregistrer des données pendant une période de temps suffisante, de l'ordre

1. Les instructions de fixation de la carte WIFI peuvent être trouvées ici : [https://www.youtube.com/watch?v=v\\_neNpfQ38Q](https://www.youtube.com/watch?v=v_neNpfQ38Q)

de deux heures, la carte SD doit avoir une mémoire importante. En considérant 1 Go par heure d'enregistrement et 50 Go pour le système d'exploitation il faut au minimum 110 Go. Une carte SD de 128 Go est donc choisie.

Le choix du micro-ordinateur embarqué est justifié par sa faible consommation d'énergie, ses bonnes performances et ses options d'interfaçage variées. De plus, ses dimensions permettent de l'insérer facilement dans un boîtier et de le transporter. L'accélération des calculs par la carte graphique représente un gros avantage dans le cas de l'utilisation d'algorithmes d'intelligence artificielle. Le coût est raisonnable au regard du prix du dispositif de capture, et des alternatives disponibles sur le marché.

### 1.4 Alimentation et connexion des modules

Afin de parvenir à un système totalement embarqué, il est nécessaire de connecter et d'alimenter tous les modules. La caméra est connectée au micro-ordinateur via USB 3.0 et le radar par câble Ethernet, comme le montre la FIGURE 1.5. L'échange des données entre les capteurs et l'ordinateur ne pose donc aucun problème.

Concernant l'alimentation, il est important d'être attentif à certains paramètres, en particulier les tensions et les courants. La caméra est alimentée via le câble USB et ne nécessite aucun autre élément pour fonctionner. En revanche le radar nécessite une source de tension délivrant entre 10 V et 16 V. Une simple batterie de 12 V est donc suffisante, même si la tension varie autour de la tension nominale. L'élément de stockage choisi est un batterie *Lithium-Polymer* (LIPO) ayant trois cellules. Chaque cellule a une tension entre 3 V et 4.2 V. La tension de la batterie varie donc entre 9 V et 12.6 V, ce qui permet une utilisation du radar quasiment jusqu'à la décharge de la batterie.

Le dernier élément à alimenter est le micro-ordinateur lui-même. Il est important qu'il reçoive une tension de bonne qualité pour qu'il ne s'éteigne pas. En effet, le constructeur a fait le choix de protéger le *Jetson Nano* en implémentant une mise hors tension de sécurité si la tension ne correspond pas aux spécifications. Afin d'avoir une tension de 5 V sans trop d'oscillations, un convertisseur buck (abaisseur de tension) de bonne qualité est choisi. La tension d'entrée acceptée varie entre 4 V et 40 V et la tension de sortie délivrée peut être réglée entre 1.25 V et 36 V. Le courant de sortie est de maximum 8 A, mais le constructeur conseille 5 A pour une utilisation à long terme. Le *Jetson Nano*® acceptant au maximum 6 A et 5 V, le convertisseur est bien adapté à la situation.

Le circuit d'alimentation ayant été choisi, il reste à dimensionner les batteries. La batterie réservée au radar doit délivrer au moins 0.6 A. Pour un fonctionnement du radar pendant 2 heures, il est possible de calculer rapidement la capacité de batterie nécessaire :

$$C = I_{moyen} \times temps \quad [Ah]$$

## Chapitre 1. Description du dispositif de capture des données

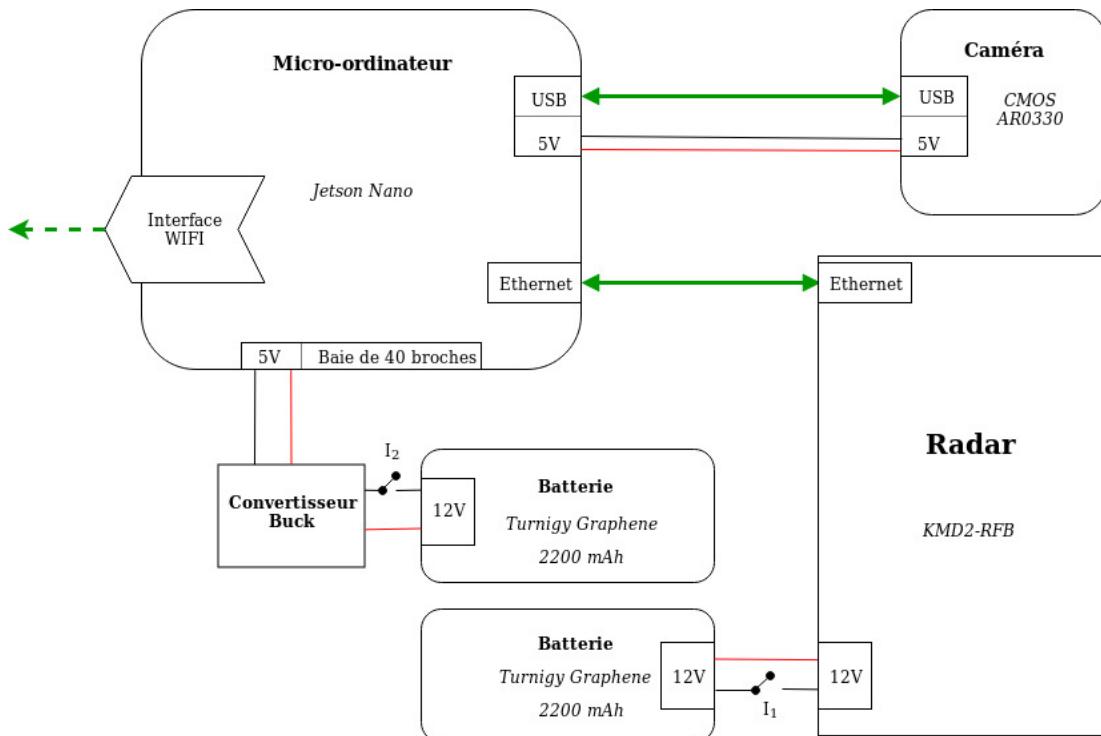


FIGURE 1.5 – Schéma d'alimentation et de connexion des modules.

Le courant de fonctionnement du radar, renseigné par la fiche technique, est de 0.6 A qui sera considéré comme valant 1 A par sécurité. Dès lors la capacité de la batterie doit être de 2 Ah au moins.

La deuxième batterie est dimensionnée sur base du courant utilisé par l'ordinateur. En sortie du convertisseur de puissance, le courant peut valoir au maximum 6 A et la tension 5 V. La puissance vaut donc 30 W au maximum. En considérant une puissance moyenne consommée valant un tiers de la puissance maximum et en négligeant les pertes dans le buck, la puissance du côté de la batterie vaut 10 W. Le courant délivré par la batterie est donc de 0.9 A environ lorsque la tension de la batterie est de 11 V, cette dernière pouvant varier entre 9 V et 12.6 V. Même si la variation de tension lors de la décharge n'est pas linéaire, une tension moyenne de 11 V a ici été choisie. La batterie doit donc avoir une capacité de 1.8 Ah au moins.

Deux batteries de 12 V et 2.2 Ah sont donc choisies et le dispositif devrait pouvoir être utilisé environ deux heures sans recharge. Le temps d'utilisation dépend en réalité de beaucoup de paramètres. Les calculs précédents donnent simplement une idée du temps d'utilisation tout en permettant de choisir une capacité de batterie adaptée. Les batteries choisies sont prévues pour alimenter des moteurs de drones et le courant maximum délivré est par conséquent largement suffisant.

Pour finir, un interrupteur est ajouté en série à chaque batterie afin de contrôler l'allumage des modules. Le schéma complet d'alimentation et de connexion peut être trouvé FIGURE 1.5.

## 1.5 Conception d'un boîtier pour le dispositif de capture

Un boîtier est nécessaire afin de maintenir tous les modules ensemble et de garantir l'alignement du radar et de la caméra. Pour le réaliser, les modules sont tout d'abord modélisés à l'aide d'un logiciel 3D.

Afin d'optimiser la fabrication du boîtier, seule la face avant est réalisée par fabrication additive. Les autres faces ne comportent pas de géométrie 3D et peuvent donc être découpées au laser. La face avant est plus complexe du fait de la fixation particulière de la caméra, comme le montre la Figure 1.6. Le bord extérieur de la caméra est inséré entre deux pièces en plastique maintenues par quatre petites vis.

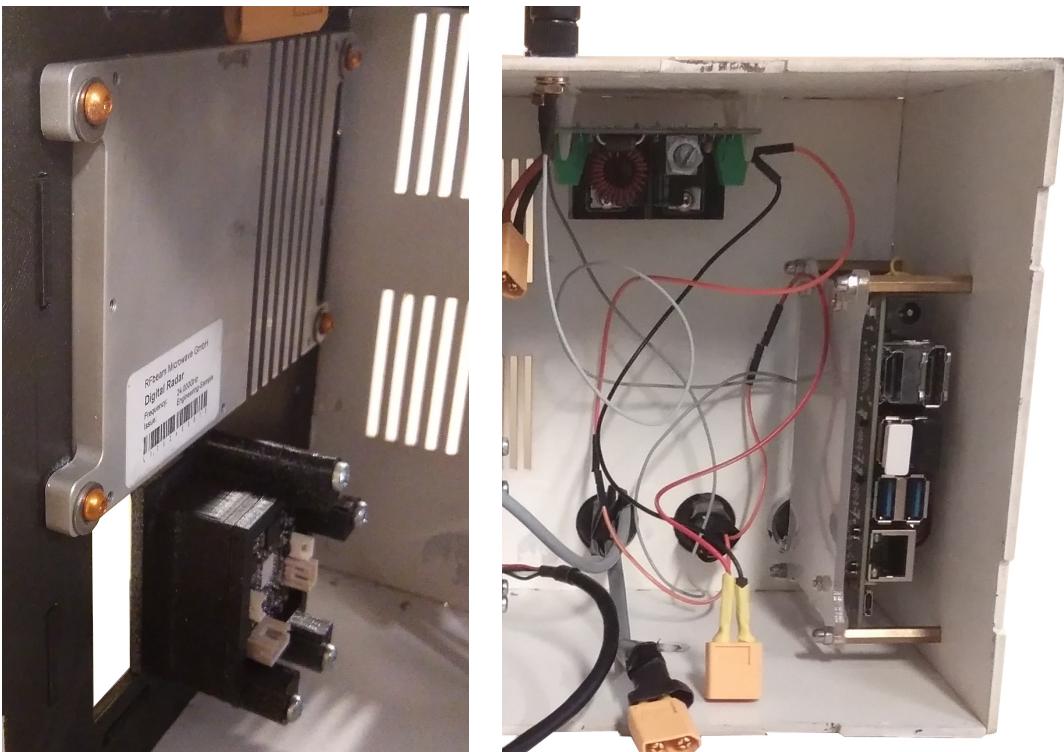


FIGURE 1.6 – Fixation des modules dans le boîtier du dispositif de capture. À gauche le radar et la caméra, à droite l'ordinateur et le convertisseur buck. Le buck est fixé au dessus, en dessous des antennes WIFI. Les ports de communication de l'ordinateur sont accessibles sans retirer le *Jetson*<sup>®</sup>.

Le radar est quand à lui directement fixé par quatre vis, les écrous étant insérés dans la face avant du boîtier. Un orifice est laissé dans la face en plastique afin de pouvoir brancher le câble Ethernet nécessaire à la communication avec l'ordinateur.

Le *Jetson Nano*<sup>®</sup> est fixé à l'aide de quatre entretoises. La FIGURE 1.6 montre l'accessibilité des ports communication lors de l'ouverture du dispositif. Sur la face supérieure du boîtier, deux trous sont prévus pour la fixation des antennes WIFI et le passage des fils. Le convertisseur

## **Chapitre 1. Description du dispositif de capture des données**

---

buck est collé à la face supérieure par quatre pièces en plastique.

Sur le côté non-amovible du dispositif, dix fentes ont été découpées près des radiateurs du radar et du buck (FIGURE 1.7). Un trou dans la face arrière permet également de faire communiquer le ventilateur de l'ordinateur avec l'extérieur. Pour finir, trois orifices circulaires permettent la fixation des boutons d'allumage du dispositif. Pour rappel, un bouton contrôle l'alimentation du micro-ordinateur et un celle du radar, comme indiqué sur la FIGURE 1.5.



FIGURE 1.7 – Système de refroidissement et interrupteurs du dispositif. À gauche les deux boutons et les 10 fentes de refroidissement. Un troisième emplacement de boutons comblé par un cache est disponible. À droite le ventilateur du *Jetson Nano*®.

Le dernier détail de conception du boîtier est la couleur. Afin d'éviter l'échauffement et de ne pas être trop visible, une peinture blanche n'absorbant pas beaucoup la lumière est choisie. Le boîtier est placé sur un trépied (voir FIGURE 1.8), afin de permettre la prise de donnée sur le lieu d'étude.

## **1.5. Conception d'un boîtier pour le dispositif de capture**

---



FIGURE 1.8 – Vue du dispositif dans son ensemble.



## 2 Algorithme de capture des données

Le dispositif de capture étant fabriqué, il est à présent nécessaire d'implémenter un algorithme permettant d'enregistrer le passage des objets en temps réel. L'alignement de la caméra et du radar est assuré par le support imprimé en 3D, mais il n'y a aucune synchronisation temporelle entre les images des deux senseurs. Une forme de synchronisation doit donc être implémentée, au travers de l'algorithme de capture. La capture des données peut sembler être une tâche évidente, mais certaines limitations matérielles amènent en pratique des contraintes, auxquelles il faut répondre.

Afin de faciliter le développement des outils informatiques, l'algorithme de capture a été implémenté en langage *Python*. Cependant, l'utilisation du langage *C* a été nécessaire pour une partie du logiciel, afin de permettre la capture en temps réel des images vidéo.

L'algorithme de capture des données doit rendre possible l'enregistrement des images vidéo, ainsi que des informations envoyées par le radar, le tout en temps réel. Les données radar sont converties en graphique distance-vitesse, et un filtrage par soustraction du bruit moyen est effectué. Les graphiques sont ensuite convertis en images, et enregistrés sous format vidéo. Les images de la caméra sont enregistrées en parallèle aux données radar, et sauvegardées sous un format vidéo. Afin de pouvoir replacer les mesures dans le temps, l'instant précis de capture des données est enregistré. Le délai entre le radar et la caméra a de plus été mesuré, afin de pouvoir synchroniser les données provenant des deux capteurs. Pour finir, une interface accessible via quelques pages *web* a été imaginée, afin que l'opérateur du dispositif puisse choisir d'enregistrer ou non les objets. Les différentes étapes de l'implémentation *software* du dispositif, ainsi que les mesures du délai entre les capteurs, sont décrites en détails dans la suite de ce chapitre.

### 2.1 Acquisition et conversion des données radar

Dans le cadre de ce travail, il a été décidé d'exploiter directement les données brutes provenant du radar. En effet, le capteur peut renvoyer les graphiques distance-vitesse, directement

## Chapitre 2. Algorithme de capture des données

---

calculés par le radar. Cependant, il a été choisi de faire la conversion sur l'ordinateur, à partir des données brutes, afin d'avoir plus de contrôle sur les paramètres de calcul des matrices Doppler (voir Section 1.2.1). Le choix de quelles antennes utiliser peut par exemple être fait. Les données brutes envoyées par le radar à l'ordinateur sont les signaux démodulés en bande de base. Ils se présentent sous la forme d'une suite de valeurs complexes, comportant une partie réelle et une partie imaginaire. C'est en réalité les composantes de *rice*, ou composantes en quadrature, des signaux démodulés des trois antennes réceptrices. Une valeur est donnée pour chaque échantillon temporel  $n$ , et pour chaque *chrip*  $l$ . Une fois les données remises sous forme d'une matrice  $n \times l$  de valeurs complexes, une FFT peut être réalisée afin de retrouver le graphique distance-vitesse, comme expliqué dans la Section 1.2.1.

Dans le graphique Doppler, les objets ressortent sous la forme de plusieurs maxima locaux, ou pics. Les données étant superposées à du bruit, il est parfois nécessaire d'adopter une stratégie de filtrage. Dans le cadre de ce mémoire, deux stratégies différentes ont été utilisées. Dans un premier temps, une valeur seuil a été définie, afin de ne représenter que les pics de valeurs importantes (FIGURE 2.1 (b)). Cette méthode a l'avantage de permettre à un opérateur de vérifier rapidement que les objets soient bien visibles et correctement enregistrés. Cependant, beaucoup d'information est perdue, et il a donc été décidé de ne pas appliquer de filtrage par valeur seuil, excepté pour l'affichage lors de la capture des données.

Dans un deuxième temps, après l'enregistrement des données, un filtrage par soustraction de l'arrière plan a été réalisé (FIGURE 2.1 (c)). En effet, deux composantes de bruit peuvent être définies. Premièrement, un bruit blanc gaussien aléatoire sur chaque pixel, de valeur moyenne nulle. Deuxièmement, un signal de bruit de valeur moyenne non nulle et qui dépend de chaque pixel. Ce bruit peut être dû à divers objets renvoyant les ondes du radar, à des interférences externes, ou encore à une fréquence de couplage interne au dispositif. En enregistrant le bruit d'arrière plan du lieu d'étude pendant une période de temps suffisamment longue, et en faisant la moyenne de ce bruit pour chaque pixel, seule la valeur moyenne non nulle du bruit de fond reste présente. Cette dernière est ensuite soustraite, pixel par pixel, pour chaque image<sup>1</sup> radar. Cette méthode marche remarquablement bien, et moins d'information est perdue, comparé à une méthode par seuil.

L'image (a) de la FIGURE 2.1 est l'image bruitée, sans aucun procédé de filtrage. En haut à droite de l'image, un ensemble de pics représentent une voiture captée par le radar. Deux pics symétriques sont présents, en haut à gauche et en bas à droite de l'image. Ces pics correspondent à un couplage interne du dispositif, et sont présents sur chaque image radar. L'image (b) montre le résultat d'un filtrage par valeur seuil, qui conserve tous les pics importants. Le véhicule, ainsi que les deux pics de bruit statique, sont bien visibles. Finalement, l'image (c) montre une réduction du bruit par soustraction de l'arrière plan. Le bruit statique a été retiré, et seul du bruit aléatoire de moyenne nulle subsiste. Les pics correspondants à la voiture sont toujours bien visibles. Il est à noter que, lors du filtrage par soustraction de l'arrière plan,

---

1. Les graphiques distance-vitesse Doppler seront souvent qualifiés d'images dans la suite de ce travail, même si ce sont en réalité des matrices représentant des signaux.

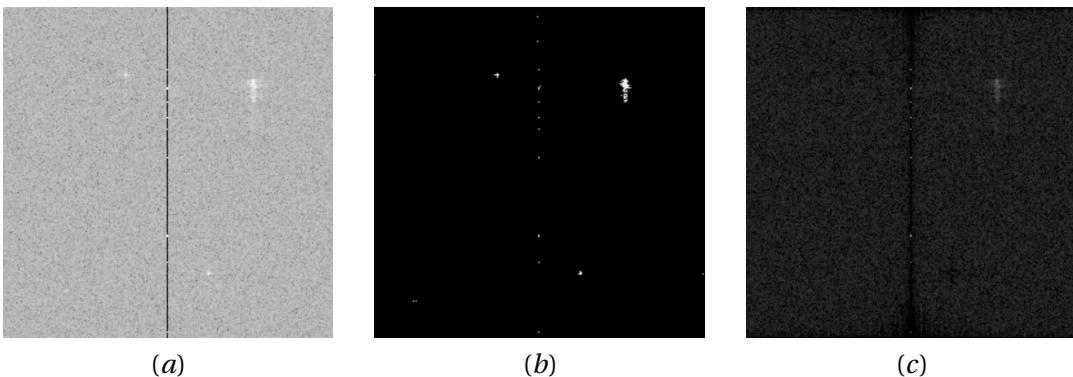


FIGURE 2.1 – Exemple de pré-traitement des données radar. L'image (a) est l'image avant filtrage, comportant beaucoup de bruit. En (b), un seuil a été appliqué, et seul les pics importants restent. L'image (c) est filtrée par soustraction de l'arrière plan.

les zones avec une moyenne de bruit supérieure subissent un filtrage plus fort, et les objets passants au travers de ces zones risquent d'être masqués.

En théorie, les deux composantes de *rice* envoyées par le radar sont parfaitement déphasées de  $90^\circ$ , et leur amplitude a été augmentée par un gain similaire. En réalité, les signaux produits par l'oscillateur local du radar ne sont pas déphasés de exactement  $90^\circ$ , et les deux sections de la chaîne de réception introduisent un gain légèrement asymétrique. Dès lors, les signaux en phase et en quadrature sont distordus [14]. En pratique, cela se manifeste par des pics "fantômes" de fréquence de Doppler inverse. Un objet fantôme avec une vitesse opposée et une distance en miroir, par rapport à la distance moyenne de détection, apparaît alors, comme le montre la FIGURE 2.2. L'asymétrie des composantes de *rice* peut être corrigée grâce à une calibration, comme celle décrite en annexe de [15].

Afin de minimiser l'espace de stockage, les données sont toutes converties sous forme d'image et stockées dans un format vidéo. Cela permet de plus d'avoir un format similaire à celui des données prises par la caméra.

## 2.2 Acquisition des données vidéo

Malgré une apparente simplicité d'utilisation de la caméra, il n'a pas été facile de remplir les exigences temps-réel de l'application. Le principal problème a été que l'utilisation de la librairie *Python OpenCV* mène à des défauts d'optimisation très importants. Il a donc été nécessaire de capturer les données vidéo par l'intermédiaire d'un code écrit en *C*. Grâce au langage *C*, il a été possible de communiquer avec la caméra à bas niveau, en s'aidant de l'API *Video4Linux*<sup>2</sup>, et des exemples de code source partagés par les membres de la communauté Linux [16]. Les images ont été enregistrées avec une résolution de  $1280 \times 720$  pixels, et

2. Plus d'informations peuvent être trouvées ici : <https://01.org/linuxgraphics/gfx-docs/drm/media/kapi/v4l2-intro.html>

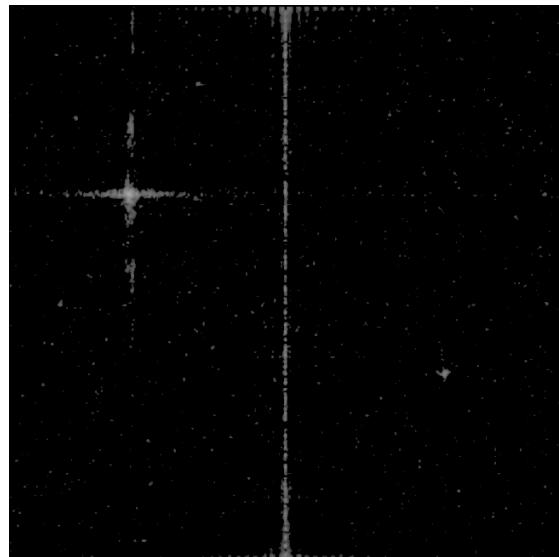


FIGURE 2.2 – Effet de l’asymétrie des composantes en phase et en quadrature. À gauche, le pic correspond à la détection d’une voiture. À droite, un pic fantôme, plus faible, apparaît

environ 30 images par seconde. L’utilisation de deux langages de programmation permet plus de parallélisme, et des meilleures performances, mais complique la synchronisation et la communication entre les processus.

Après leur capture, les images sont enregistrées sous forme de vidéos afin de minimiser l’espace de stockage, et de permettre à l’utilisateur de les visionner facilement.

### 2.3 Synchronisation du radar et de la caméra

Afin de pouvoir faire correspondre les données radar et les données vidéo, il est nécessaire de pouvoir les replacer dans le temps, au moins relativement par rapport à une référence commune. Tout d’abord, le lancement de la capture des données doit se faire plus ou moins en même temps. Pour cela, un système de consigne (*flag*) a été implémenté. Lorsque la capture des données radar commence, la consigne est mise à 1. Dès que le programme *C*, responsable de la capture des données vidéo, détecte la consigne, la vidéo commence à être enregistrée. À l’inverse, lorsque la consigne est égale à 0, les données ne sont pas enregistrées. L’exécution du programme *C* est donc, en quelque sorte, asservie à celle du code *Python*. Ce choix de conception permet de gérer les interactions avec l’utilisateur uniquement au travers d’une interface *Python*.

À défaut de pouvoir imposer une synchronisation lors de la capture, il a été décidé d’enregistrer l’instant de capture de chaque image, sur base du temps propre de l’ordinateur. Le temps pris en compte pour les données vidéo est celui correspondant à l’acquisition d’une nouvelle image par le programme *C*. Concernant les données radar, le temps de référence est la réception, par

## 2.3. Synchronisation du radar et de la caméra

le code *Python*, d'un paquet via le connecteur Ethernet. Le seul délai restant entre les deux senseurs est celui dû à la latence d'acquisition des données.

### 2.3.1 Calcul de la latence et du délai des capteurs

Une simple procédure de test a été mise en place afin de calculer la latence et le délai des deux senseurs. D'une part, la latence de la caméra a été mesurée, et d'autre part le délai entre le radar et la caméra a été estimé. À l'aide de ces deux informations, la latence du radar peut être trouvée.

Afin de mesurer la latence de la caméra, la capacité du capteur à détecter un changement peut être déterminée facilement. Pour cela, la caméra filme le moniteur de l'ordinateur, qui peut afficher une couleur parmi deux. Dès que le programme détecte que l'écran est d'une certaine couleur, le fond d'écran est immédiatement changé pour afficher la couleur opposée. Si la caméra et l'ordinateur avaient une latence nulle, l'écran changerait immédiatement de couleur, selon le délai entre les images prises par la caméra. La latence avant les changements de couleurs de l'écran est principalement due à la caméra<sup>3</sup>, et non au temps nécessaire à l'ordinateur pour changer l'écran de couleur. Celle-ci mesure donc le temps entre la capture de l'image par la caméra, et le traitement de l'information par l'ordinateur. Les résultats des mesures de latence sont repris sur la FIGURE 2.3 (a). En moyenne, un changement de l'environnement est détecté en environ 250 [ms].

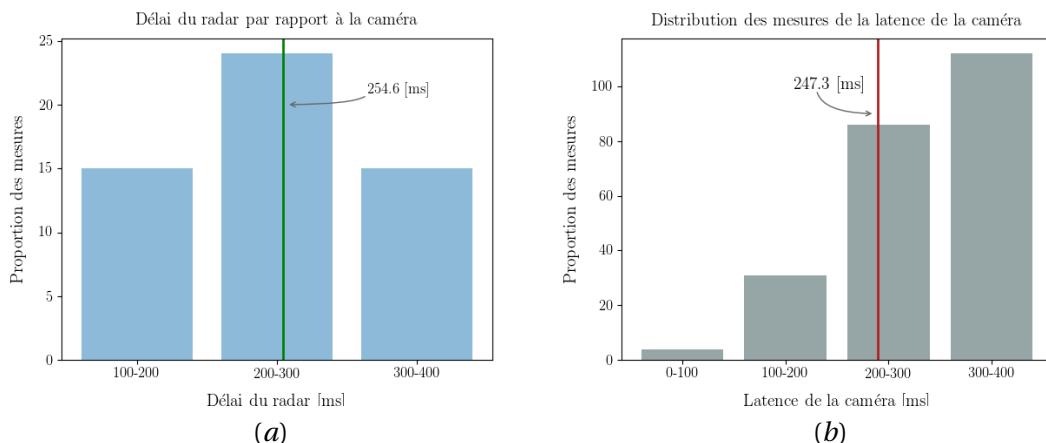


FIGURE 2.3 – Mesure de la latence et du délai des capteurs. En a., le délai du radar par rapport à la caméra est autour de 255 ms. En b, la latence de capture de la caméra a tendance à se situer entre 200 ms et 400 ms, en ayant une moyenne d'environ 247 ms.

Pour ce qui est du délai entre le radar et la caméra, il est nécessaire de mesurer un évènement ayant lieu à un instant connu, dans le référentiel de chacun des deux capteurs. Dans le cadre de ce travail, il a été décidé de se baser sur la distance des objets. En effet, la mesure de distance

<sup>3</sup>. La procédure complète ayant été suivie est décrite ici : <https://hackernoon.com/how-to-measure-the-latency-of-a-webcam-with-opencv-1a3d4a86558>

## Chapitre 2. Algorithme de capture des données

---

est directe avec le radar, et peut être déduite à l'aide de repères dans le référentiel de la caméra. En mesurant à quel instant une cible passe un certain repère placé à une distance connue, le délai entre les deux capteurs peut être calculé. La fréquence de capture de la caméra et du radar étant finie, il est important de prendre plusieurs mesures, pour palier au fait que la cible n'est pas capturée exactement au même endroit. Les résultats des mesures sont repris sur la FIGURE 2.3 (b), et montrent un délai moyen d'environ 250 [ms].

Les résultats de latence et de délai des capteurs sont repris dans le TABLEAU 2.1. La latence du radar a été calculée sur base de la latence de la caméra, et du délai entre les deux capteurs. La latence du radar peut sembler relativement longue, mais la latence des systèmes de détection radio-fréquence se situe généralement entre 200 [ms] et 500 [ms] selon [17].

<i>Latence moyenne de la caméra</i>	247.3 [ms]
<i>Latence moyenne du radar</i>	501.9 [ms]
<i>Délai moyen entre les deux capteurs</i>	254.6 [ms]

TABLEAU 2.1 – Valeurs moyennes des latences et du délai des capteurs

## 2.4 Interface de contrôle de la capture de données

Afin de permettre aux utilisateurs de contrôler la prise des données, un système de communication via WIFI a été mis en place, au moyen d'un serveur *Flask*<sup>4</sup> implémenté sur le micro-ordinateur du dispositif. Le principe des serveurs *Flask* est très simple, mais la solution choisie est suffisante pour le but recherché. Le serveur consiste en un certain nombre de pages web *HTML*, affichant les données capturées, ainsi que quelques options de contrôle de la capture. Le *Jetson Nano*<sup>®</sup> génère un hotspot WIFI, auquel n'importe quel type d'appareil peut se connecter très facilement. L'opérateur peut, par exemple, utiliser une simple tablette, voir même un GSM. La FIGURE 2.4 reprend l'architecture logicielle du système de capture et d'enregistrement des données.

Lors de la connexion au dispositif, une première page propose l'initialisation des senseurs. Principalement, cette étape consiste à envoyer tous les paramètres de fonctionnement au radar. Si quelque chose se passe mal, un message d'erreur est renvoyé à l'utilisateur. Lorsque tout est prêt, la capture des données peut être lancée. Par défaut, les données ne sont pas enregistrées en mémoire, afin de laisser le choix à l'opérateur de sauver uniquement les objets intéressants. L'enregistrement peut donc être activé ou désactivé à l'aide d'un simple bouton, présent sur la page *web* principale. Cela permet, de ne pas sauver les données lorsqu'il n'y a pas de circulation, et donc d'économiser de la mémoire.

Pour finir, une sécurité pour éviter la saturation de la mémoire a été mise en place. Lorsque la taille des données enregistrées dépasse 80 Go, la sauvegarde ne peut plus avoir lieu, et l'opérateur est prévenu via un message d'erreur.

---

4. Plus d'informations sont disponibles ici : <http://flask.palletsprojects.com/en/1.1.x/>

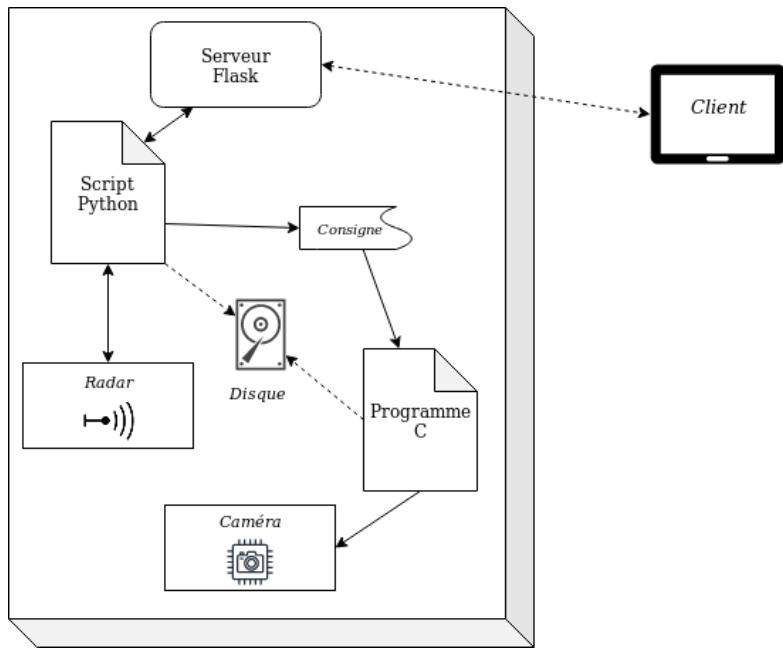


FIGURE 2.4 – Architecture logicielle de l'algorithme de capture des données

La communication distante via WIFI permet beaucoup de flexibilité, et permet à l'opérateur de ne pas rester à côté du dispositif. Ainsi, les usagers de la route voient moins le dispositif, et ne ralentissent pas systématiquement. Lors de la capture des données, il a en effet été observé la curiosité de certains conducteurs, et même l'arrêt de certains piétons. Il est donc avantageux de ne pas trop attirer l'attention sur le dispositif, afin de ne pas introduire de biais dans les résultats de l'analyse du lieu d'étude.

## 2.5 Capture des données sur le lieu d'étude

Grâce au dispositif de capture, des données de circulation ont pu être récoltées à différents moments de la journée. Le TABLEAU 2.2 reprend les informations liées à la capture des données, et un décompte approximatif des objets observés.

Entre les différentes prises captures, le placement du dispositif a été soigneusement gardé constant, à l'aide de repères au sol. L'angle de la caméra et du radar ont été bloqués sur base de repères visibles sur les images vidéo, de manière approximative. La distance par rapport au carrefour est donc supposée constante, mais l'angle du dispositif peut varier très légèrement.

## Chapitre 2. Algorithme de capture des données

---

Tranche d'heures	Véhicules	Piétons	Deux-roues	Fréquentation relative
7h50 à 9h10	339	54	22	0.87
10h20 à 12h20	369	68	28	0.65
11h à 12h20	290	39	10	0.71
14h à 15h45	387	105	6	0.80
17h à 18h45	518	62	46	1.00
20h30 à 21h30	90	20	4	0.32

TABLEAU 2.2 – Statistiques de capture des données. La fréquentation relative est calculée par rapport à la tranche d'heures durant laquelle le passage est maximal, et correspond à la fréquentation de tous les usagers confondus. Les tranches d'heures correspondent à différentes prises de données en continu, n'ayant pas forcément eu lieu le même jour.

# **3 Aspect théorique des méthodes de détection et classification des objets**

Dans le cadre de ce travail de fin d'études, il a été décidé d'utiliser un algorithme opérant image par image. Dès lors, l'information temporelle ne sera utilisée que pendant l'étape de fusion des données, comme expliqué au Chapitre 5. Les objets sont donc tout d'abord identifiés sur chaque image, en n'utilisant aucune autre information que celle disponible à un instant donné. Cependant, avant même de pouvoir calculer la position des objets dans l'espace et de déterminer leur type, il est nécessaire de les isoler de l'arrière plan de l'image. Cette étape de détection n'est pas évidente, et peut prendre un certain temps. Lorsque la détection est faite, l'étape de classification permet de différencier les objets entre eux.

## **3.1 Critères d'évaluation des algorithmes de classification**

Chaque algorithme ayant des performances différentes, il faut définir sur quel critère se baser, lors de la comparaison des différentes méthodes. Il est courant de parler de précision au quotient, mais il est parfois nécessaire de définir le terme de manière plus rigoureuse. Dans le cadre des algorithmes de classification, la précision est le rapport entre le nombre de vrais positifs, et le nombre total de positifs prédits :

$$\text{précision} = \frac{\text{vrais positifs}}{\text{nombre total de positifs}} = \frac{\text{vrais positifs}}{\text{vrais positifs} + \text{faux positifs}}$$

La précision est donc un nombre compris entre 0 et 1, souvent mis sous la forme d'un pourcentage. Plus ce pourcentage est élevé, plus l'algorithme est fiable en ce qui concerne la prédiction de l'appartenance d'un objet à une certaine classe.

Un second critère utilisé pour évaluer les algorithmes de classification est le rappel, aussi appelé sensibilité. Il est défini comme étant le nombre de vrais positifs, divisé par le nombre d'objets réellement présents parmi les données de test.

$$\text{rappel} = \frac{\text{vrais positifs}}{\text{nombre total de positifs}} = \frac{\text{vrais positifs}}{\text{vrais positifs} + \text{faux négatifs}}$$

Le rappel renseigne donc sur la capacité d'un algorithme à trouver les objets et à les classifier.

La précision et le rappel peuvent être combinés afin de donner un nouveau critère, appelé le *scoreF1*. Cette mesure est en réalité la moyenne harmonique de la précision  $p$  et du rappel  $r$ , et pénalise les mauvais compromis :

$$F1_{score} = \frac{2}{r^{-1} + p^{-1}} = 2 \cdot \left( \frac{p \cdot r}{p + r} \right)$$

Un autre indicateur beaucoup utilisé, dans le cadre des algorithmes nécessitant un entraînement supervisé, est la précision moyenne (notée AP pour *Average Precision*). Ce critère d'évaluation a été introduit à l'occasion du concours Pascal VOC 2007 [18], afin de pouvoir mieux interpréter les résultats de la compétition. De plus, ce critère donne, en quelque sorte, un indice de performance de la détection et de la classification en même temps. D'après sa première définition, lors du concours Pascal VOC 2007, la précision moyenne est définie comme étant :

$$AP = \frac{1}{11} \sum_{r \in 0,0.1 \dots 1} p_{interp}(r)$$

où  $r$  est le rappel et  $p_{interp}(r)$  est la valeur de la précision interpolée à un certain rappel  $r$ . Comme le montre la FIGURE 3.1, la précision interpolée est, pour chaque valeur de rappel  $\tilde{r}$ , la précision maximum sur l'intervalle de rappel allant de  $\tilde{r}$  à 1 :

$$p_{interp} = \max_{\tilde{r} > r} p(\tilde{r})$$

Cette interpolation permet de diminuer la sensibilité vis-à-vis des petites variations localisées de la précision.

Afin de calculer l'*AP*, il est nécessaire de déterminer le nombre de vrais et de faux positifs. Pour cela, les boîtes entourant les objets détectés sont comparées avec les boîtes des objets réellement présents, en tenant bien entendu compte de la classe de l'objet à détecter. Le critère de comparaison des boîtes est basé sur un *indice de Jaccard* entre les aires prédites et réelles. L'*indice de Jaccard* est ici le rapport entre l'intersection et l'union des aires de deux boîtes, aussi souvent noté *IoU* (*intersection over union*). Lors du concours Pascal VOC 2007, l'*IoU* minimum validant une détection a été fixé à 0.5. Par la suite, différents seuils ont été utilisés, comme par exemple 0.75 ou même 0.9. D'autres critères peuvent être construits en modifiant le nombre de points où interpoler la précision, ou en faisant une moyenne des *AP*.

### 3.2. Choix de la méthode de détection et de classification des objets

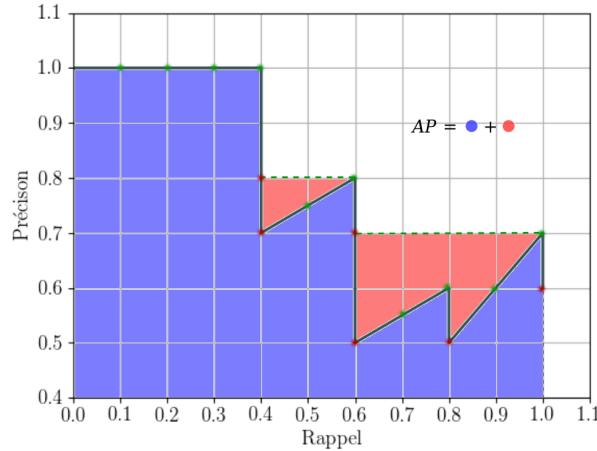


FIGURE 3.1 – Calcul de la précision moyenne par interpolation de la courbe rappel-précision

sur plusieurs seuils d'*IoU*. Dans le cadre de ce travail, l'*AP* sera toujours basé sur un seuil de 50 %, sauf mention contraire.

Le critère de précision moyenne se base donc à la fois sur la localisation des objets, et sur leur classe. La détection et la classification sont donc toutes deux considérées. Le critère d'*AP* est calculé pour chaque classe d'objet séparément. Afin d'obtenir un critère global d'évaluation d'un algorithme en particulier, le *mean – AP* est calculé, comme étant simplement la moyenne des précisions moyennes sur toutes les classes d'objets. Beaucoup d'autres critères d'évaluation existent, mais il ne seront pas utilisés dans le cadre ce travail.

Avant de présenter l'algorithme de détection et de classification choisi, un rapide tour d'horizon des méthodes modernes disponibles va être fait. Une étude plus approfondie de l'algorithme sélectionné fera l'objet de la Section 3.3.

## 3.2 Choix de la méthode de détection et de classification des objets

Afin de choisir une méthode de détection et de reconnaissance des objets circulant sur la route, un rapide tour d'horizon des algorithmes disponibles va être fait. Les explications qui suivent sont une très brève synthèse de [19] et [20], en particulier la partie concernant l'évolution des méthodes de détection des objets.

Jusqu'à très récemment, la détection et la reconnaissance d'objets se basait sur une méthode en trois étapes : la sélection de régions contenant potentiellement des objets, l'extraction de caractéristiques et finalement la classification. La sélection des régions d'intérêt était alors souvent basée sur l'utilisation de fenêtres glissantes, associée à la variation de la taille des images ou des fenêtres elles-mêmes. La seconde étape consistait en l'utilisation de filtres (tel que Haar, HOG ou encore SIFT), supposés extraire les caractéristiques des objets recherchés.

Ces filtres étaient basés sur des méthodes d'extraction de l'information théorisées par des chercheurs, et non sur un apprentissage, supervisé ou non. Finalement, la classification des objets se faisait sur base des caractéristiques extraites, à l'aide d'algorithmes tel que celui des *machines à vecteurs de support*, *AdaBoost* ou encore d'un *deformable part model*. Cette étape pouvait comporter un peu d'apprentissage supervisé, sans pour autant en dépendre entièrement.

À partir de 2010, l'évolution jusqu'alors rapide des méthodes par définition manuelle des filtres d'extraction des caractéristiques, finit par s'essouffler. En effet, ces méthodes nécessitent la définition d'algorithme d'extraction des caractéristiques par un être humain, ce qui en limite la complexité et demande des compétences spécialisées. De plus, l'utilisation de fenêtres glissantes et de tailles des fenêtres variables, demande de faire un compromis entre temps de calcul et précision. En 2014, des méthodes basées sur les réseaux de neurones convolutifs (RNC) font leur apparition, définissant un nouveau paradigme quand aux méthodes de classification. Assez rapidement, les différentes méthodes basées sur les RNCs se divisent en deux branches : les détecteurs en deux étapes, et les détecteurs en une seule étape.

#### 3.2.1 Détecteurs en deux étapes

En réalité, les RNCs existent déjà depuis longtemps, car dès les années 1980, ils sont développés dans le cadre de recherches visant à reproduire les mécanismes ayant lieu dans le cerveau humain. Leur utilisation dans le cadre de la reconnaissance d'objets, est devenu possible grâce au développement de bases de données importantes, et d'outils de calcul adaptés au traitement de données en parallèle, tel que les GPUs.

En 2014, un nouvel algorithme appelé *Region Convolutional Neural Network* (R-CNN), fait son apparition. Alors que le *mAP* atteint par les méthodes plus traditionnelles des années précédentes ne dépasse pas 33.7 % lors du concours de référence *PASCAL VOC*, R-CNN atteint un *mAP* de 58.5 %. L'algorithme est alors encore assez proche des méthodes plus anciennes, mais l'extraction des caractéristiques se fait à l'aide d'un *RNC*. Dès lors, la capacité du réseau à extraire des caractéristiques pertinentes peut être développée à l'aide d'un entraînement supervisé, permettant ainsi l'utilisation d'un ensemble de paramètres plus complexes. Cependant, R-CNN est encore relativement lent, car la sélection des régions contenant potentiellement des objets se fait toujours par une recherche séquentielle.

Afin de palier aux problèmes accompagnant R-CNN, l'algorithme *Spatial Pyramid Pooling Network* (SPPNet) permet de se passer de la nécessité de redimensionner les images et les régions, lors de la phase de détection des objets. Les couches SPPNet permettent donc de gagner beaucoup de vitesse, tout en préservant les performances de R-CNN. Cependant, l'entraînement des couches est complexe, car chaque couche doit être entraînée séparément.

Peu de temps après l'apparition de *SPPNet*, *Fast R-CNN* connaît son heure de gloire, en tirant profit des dernières avancées dans le domaine. Lors de la compétition *PASCAL VOL*, le *mAP*

### **3.2. Choix de la méthode de détection et de classification des objets**

---

fait un bon en avant, jusqu'à atteindre 70 %. Contrairement à *SPPNet*, *Fast R-CNN* peut être entraîné plusieurs couches à la fois, ce qui représente un gain de temps, et permet un degré d'abstraction supérieur. Cependant, l'algorithme se base toujours sur la recherche séquentielle de régions d'intérêt, même si l'extraction des caractéristiques se fait avec un RNC.

L'algorithme *Faster R-CNN* introduit l'idée d'utiliser un RNC lors de l'étape de proposition des régions d'intérêt. C'est donc le premier algorithme purement basé sur les RNCs. Le *mAP* augmente alors jusqu'à 73.2 %, et la vitesse d'exécution devient proche du temps-réel.

En parallèle aux algorithmes précédemment mentionnés, une multitude de propositions d'améliorations voient le jour. En particulier, des méthodes qui ne séparent plus la détection et la classification en deux étapes font leur apparition.

#### **3.2.2 DéTECTEURS EN UNE ÉTAPE**

En 2015, un nouveau type d'algorithme très rapide apparaît. Tout d'abord, une première version de l'algorithme *You Only Look Once* (*YOLO*) permet d'aller beaucoup plus vite, sans que la précision ne chute complètement. Ce gain de vitesse est permis par la fusion de l'étape de détection avec celle de classification. L'image est divisée en cases, et la présence d'objets dans chaque case est évaluée en parallèle. Donc la proposition des régions est remplacée par la division en cases de l'image, l'extraction des caractéristiques se fait à l'aide d'un RNC, et la classification se fait dans la foulée, grâce à des couches de neurones spécifiques. Dans sa première version, *YOLO* atteint un *mAP* de 63.4 %, et les améliorations apportées par les versions 2 et 3 permettent d'arriver à un *mAP* de plus de 78.6 %. Cependant, du fait de la division de l'image en cases de tailles et de positions fixes, l'algorithme peine à détecter les petits objets, en particulier lorsqu'ils sont proches les uns des autres. De plus, la localisation des objets est en général moins précise qu'avec les méthodes en deux étapes.

L'algorithme *Single Shot Multibox Detector* (*SSD*) propose une méthode de détection des objets à différentes échelles, faite par différentes couches de neurones. Cela permet une précision un peu supérieure à *YOLO*, mais le temps de calcul et la complexité sont augmentés. Dans sa première version, le *mAP* atteint 76.8 %.

Finalement, l'algorithme *RetinaNet* se base sur le postulat que, le manque de précision associé aux détecteurs à un étage, est dû au déséquilibre entre les régions occupées par les objets et celles vides. En effet, les données d'entraînement comportent des images qui, même avec une grande densité d'objets, comportent beaucoup plus de zones sans objet. *RetinaNet* propose donc une méthode qui accorde plus d'importance aux vrais positifs qu'aux vrais négatifs.

Dans le cadre ce projet, il a été décidé d'utiliser la version 3 de l'algorithme *YOLO*. En effet, *YOLO* offre de bonnes performances, sans pour autant être trop complexe à appréhender, et surtout à entraîner. De plus, le cas d'étude présente des objets relativement proches de la caméra, et étant donc de grande taille sur les images. La vitesse de l'algorithme est un atout lors de l'entraînement avec un jeu de données important, et est nécessaire dans le cas d'une

utilisation en temps réel du dispositif. Finalement, *YOLO* fait preuve d'un degré d'abstraction très important, ce qui laisse penser que l'algorithme devrait être capable de fonctionner aussi avec les données radar.

### **3.3 You Only Look Once : principe de fonctionnement de l'algorithme**

Cette section propose une explication de l'algorithme *YOLO*, directement basée sur trois articles [21],[22],[23], publiés par ses auteurs. Le but n'est pas de décrire rigoureusement tous les détails de l'algorithme, mais simplement d'en donner une compréhension suffisante pour son utilisation dans le cadre de ce travail. Certains détails techniques, notamment sur l'architecture du réseau de neurones, seront volontairement omis.

L'approche utilisée pour développer *YOLO* est similaire à celle d'un problème de régression linéaire. Sur base d'un ensemble de pixels, le modèle doit être capable de trouver les différents objets et leur localisation. Ce processus est fait en un seul passage de l'information au travers d'un réseau, comportant 106 couches de neurones. Les processus de détection et de classification se font dès lors en une seule étape. Une architecture en une seule étape a de plus le gros avantage de pouvoir être entraînée globalement, en partant des images, et en cherchant directement à optimiser le *mAP*. De plus, l'architecture *YOLO* limite le nombre de calculs redondants, comparé au détecteurs en plusieurs étapes, ce qui la rend très rapide. En traitant des images une par une (pas de *batch* d'images) à l'aide d'une carte graphique *Nvidia® Titan X*, 45 images par secondes peuvent être traitées. La version rapide de l'algorithme, *tiny-YOLO*, permet même d'atteindre 150 images par seconde. Avec le *Jetson Nano®* de *Nvidia®*, des résultats plus modestes sont obtenus : environ 3 images par seconde pour le réseau complet, et 25 images par seconde pour la version rapide.

Un autre avantage d'une architecture travaillant sur la totalité des images, est l'apparition de capacités de globalisation. En effet, les informations du contexte des objets sont en quelque sorte apprises en même temps que les caractéristiques des objets, ce qui évite à l'algorithme beaucoup de faux positifs détectés dans l'arrière plan des images. De plus, *YOLO* travaille avec des caractéristiques relativement générales, ce qui permet de ne pas être trop sensible à des variations mineures de la texture ou de la forme des objets. L'algorithme peut ainsi détecter des personnes ou des animaux sur des peintures, ou encore des objets dans des jeux vidéo ([21], p.8). Ces capacités sont encourageantes dans le cadre de ce projet, car elles laissent penser que l'algorithme pourrait interpréter de manière correcte les données radar.

Malgré tous ses atouts, *YOLO* souffre de quelques désavantages. En effet, la localisation des objets est moins précise qu'avec des détecteurs multi-étapes, en particulier pour les objets de petite taille. Une nuée d'oiseau peut par exemple donner lieu à beaucoup de faux négatifs. Les versions 2 et 3 de *YOLO* tendent à corriger ce problème, mais il est vrai que ce n'est pas un point fort de l'algorithme. Dans le cadre de ce travail, il a été considéré que les objets d'intérêt seront suffisamment grands sur les images de la caméra. Concernant les données radar, les pics sont parfois relativement petits, mais la structure du bruit paraît peu complexe. Il est donc postulé

que YOLO parviendra à détecter les pics dans le graphique distance-vitesse, même lorsqu'ils sont relativement petits.

### 3.3.1 Principe de fonctionnement de la détection et de la classification

Tout d'abord, l'image en entrée du réseau est divisée en  $13 \times 13$  cases. Chaque case est "responsable" de la détection des objets dont elle contient le centre. Les cases sont ici une alternative aux fenêtres glissantes, qui à l'avantage de permettre un calcul en parallèle.

Dans sa première version, chaque case prédisait deux boîtes autour des objets dont elle était responsable. Afin d'augmenter le rappel, les dernières versions utilisent des *anchor boxes*. Les *anchor boxes* sont des boîtes représentatives des dimensions des objets à détecter. Pour commencer, un algorithme *k-mean* est utilisé sur les données d'entraînement, pour faire ressortir 9 dimensions de boîtes différentes et représentatives des objets. Les 9 boîtes sont ensuite séparées selon trois échelles de tailles différentes. Les *anchor boxes* sont utilisés pour déterminer si un objet est présent autour d'une case. En effet, pour toutes les cases définies sur l'image, l'*IoU* de chaque boîte de référence avec chaque boîte prédite est calculé. Si l'*indice de Jaccard* d'une boîte prédite avec toutes les *anchor boxes* est inférieur à 0.5, la détection n'est pas validée. Dans le cas contraire, la détection est associée à la boîte avec laquelle elle se superpose le mieux. Dès lors, la prédiction est définie en fonction de sa boîte de référence :

$$\begin{aligned}
 b_x &= \sigma(t_x) + c_x \\
 b_y &= \sigma(t_y) + c_y \\
 b_w &= p_w e^{t_w} \\
 b_h &= p_h e^{t_h} \\
 P(\text{détexion}) \times \text{IoU}(\text{référence}, \text{détexion}) &= \sigma(t_0)
 \end{aligned}$$

où :

- $b_x, b_y$  sont les coordonnées du centre de la boîte prédite
- $b_w, b_h$  sont la largeur et la hauteur de la boîte prédite
- $t_x, t_y$  sont les déviations du centre de la boîte prédite par rapport à la boîte de référence
- $t_w, t_h$  sont les déviations des dimensions de la boîte prédite par rapport à celles de la boîte de référence
- $c_x, c_y$  sont les coordonnées du centre de la case responsable de la détection
- $p_w, p_h$  sont les dimensions de largeur et de hauteur de la boîte de référence
- $t_0$  est l'indice de confiance de la détection
- $\sigma$  est une fonction d'activation, la fonction sigmoïde définie sur la FIGURE 3.2.
- $P$  dénote une probabilité

Toutes les grandeurs introduites ci-dessus sont normalisées par rapport aux dimensions de

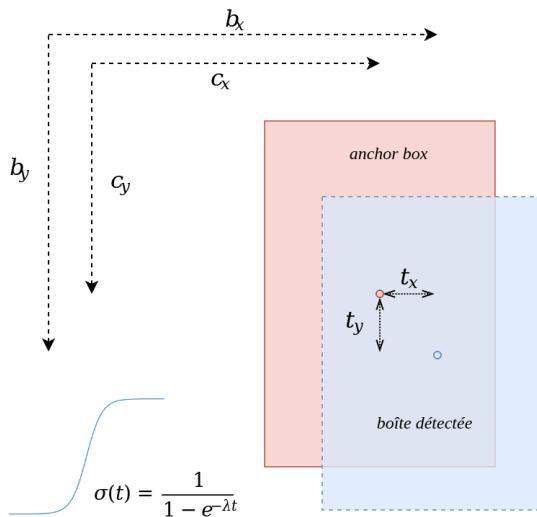


FIGURE 3.2 – Définition des coordonnées de position des boîtes détectées.

l'image d'entrée, et sont donc comprises entre 0 et 1. La normalisation des dimensions, par la hauteur et la largeur de l'image, permet de ne pas avoir à modifier les valeurs trouvées lorsque l'image est redimensionnée. Dès lors, il est aussi possible de modifier la taille des images d'entraînement sans avoir à adapter les annotations.

Lors de la phase d'entraînement, l'indice de confiance d'une boîte détectée est donné par :

$$confiance = t_0 = P(\text{objet}_i) \times IoU_{det}^{\text{réel}}$$

La probabilité  $P(\text{objet}_i)$  doit être égale à 1 si un objet est présent, et 0 sinon. Le terme  $IoU_{det}^{\text{réel}}$  dénote l'*indice de Jaccard* entre la boîte détectée et la boîte annotée pour l'entraînement. Le but de l'entraînement est bien sûr de modifier les paramètres de manière à augmenter l'indice de confiance (donc l' $IoU$ ), en se basant sur la fonction de perte définie à la Section 3.3.3.

Concernant la classification, chaque case se charge des objets qu'elle contient. Pour chaque objet détecté, une probabilité conditionnelle de chaque classe est calculée :

$$p_i = P(\text{classe}_i | \text{objet}), \quad i \text{ variant sur le nombre de classes}$$

Ces probabilités sont calculées sur base des caractéristiques extraites par le RNC.

En sortie du réseau de neurones, l'algorithme fournit un tenseur de dimension  $S \times S \times B \times (5+C)$ . En effet, le nombre d'éléments en sortie est proportionnel au nombre de cases,  $S \times S$ , et au nombre  $B$  de boîtes de référence pour chaque case. Pour chaque objet, les 5 grandeurs associées au placement et aux dimension de la boîte sont nécessaires, ainsi qu'une probabilité

### **3.3. You Only Look Once : principe de fonctionnement de l'algorithme**

---

pour chaque classe de l'ensemble  $C$ . Même si beaucoup de boîtes de référence ne sont associées à aucun objet, la dimension des données de sortie est constante.

Finalement, un indice de confiance global  $c$  peut être construit, sur base de l'indice de confiance lié à la boîte détecté, et de la probabilité conditionnelle associé à la classe  $i$  :

$$c_i = P(\text{classe}_i | \text{objet}) \times P(\text{déttection}) \times IoU_{\text{préd}}^{\text{réf}} = P(\text{classe}_i) \times IoU_{\text{préd}}^{\text{réf}}$$

Ce critère global reprend la confiance sur la détection et sur la classification.

Le principe de base de l'algorithme est donc assez simple. Certaines règles s'y ajoutent, afin de ne pas détecter les objets plusieurs fois, ou de déterminer quel objet choisir si le recouvrement ou la probabilité est identique avec plusieurs boîtes de référence ou objets. Une suppression des valeurs non maximales est aussi réalisée, car il arrive que des objets soit détectés plusieurs fois. En effet, si le centre d'un objet est proche des bords de deux cases différentes, il peut être détecté deux fois.

#### **3.3.2 Architecture du réseau de neurones**

Dans cette section, l'architecture du réseau va très rapidement être introduite. Il serait trop long, et certainement inutile, de décrire le fonctionnement mathématique de chaque couche, donc les couches seront simplement introduites sur base de leur fonction.

La version 3 du réseau *YOLO* est composée de 106 couches de neurones. Il est possible de séparer ces couches en 4 catégories différentes : les couches convolutives, les couches de détection, les couches *route*, et les couches d'augmentation (*upsampling*).

**Couche convulsive :** une convolution est faite entre les données d'entrée, et un certains nombre de filtres. La taille des données de sortie dépend du nombre de filtres utilisés, et du paramètre *stride*. Le *stride* peut être vu comme étant égal au pas d'une fenêtre glissante. Dès lors, un *stride* valant 2 réduira la taille des données par un facteur 2, car une seule valeur sera calculée toute les deux cases. Par exemple, soit une image de  $608 \times 608$  pixels et 3 canaux de couleur, à laquelle 32 filtres de taille  $3 \times 3$  et de *stride* valant 2 sont appliqués. Lors de la convolution, les sous groupes de dimension  $3 \times 3$  de l'images sont filtrés pour donner une unique valeur. Le *stride* étant de 2, cette valeur ne sera pas calculée pour chaque ligne et chaque colonne, mais seulement une fois sur deux. Dès lors, la dimension de données de sortie sera de  $304 \times 304 \times 32$ . En quelque sorte, les trois canaux de l'images on été compressés en un seul canal, mais pour 32 types de filtres différents. Les filtres permettent ainsi d'extraire certaines caractéristiques utiles de l'ensemble des données d'entrée.

**Couche de détection :** lorsque les caractéristiques souhaitées ont été extraites par les différentes couches convolutives, des couches spécifiques synthétisent les données afin de localiser et de reconnaître les objets. La dernière version de l'algorithme utilise trois couches de détection à des échelles différentes, suivant le principe des *feature pyramidal networks*. Les 9 *anchor boxes* calculées au préalable sont reparties selon ces trois mêmes échelles. Les échelles plus fines permettent de détecter des objets plus petits. Contrairement aux couches convolutives, les couches de détection sont spécifiques à *YOLO*, et tous les neurones sont connectés à tous les neurones de la couche précédent. Ainsi, un raisonnement global peut être fait.

**Couche route :** après les couches de détection, des données antérieures sont récupérées, en quelque sorte pour aller chercher de l'information plus primaire. Ces données sont *routées* par des couches spécifiques.

**Couche d'augmentation :** les trois différentes échelles de détection travaillent avec des données de plus en plus fines. Il est dès lors nécessaire d'augmenter la taille des données après les deux premières couches de détection. L'information peut être simplement dupliquée, ou interpolée.

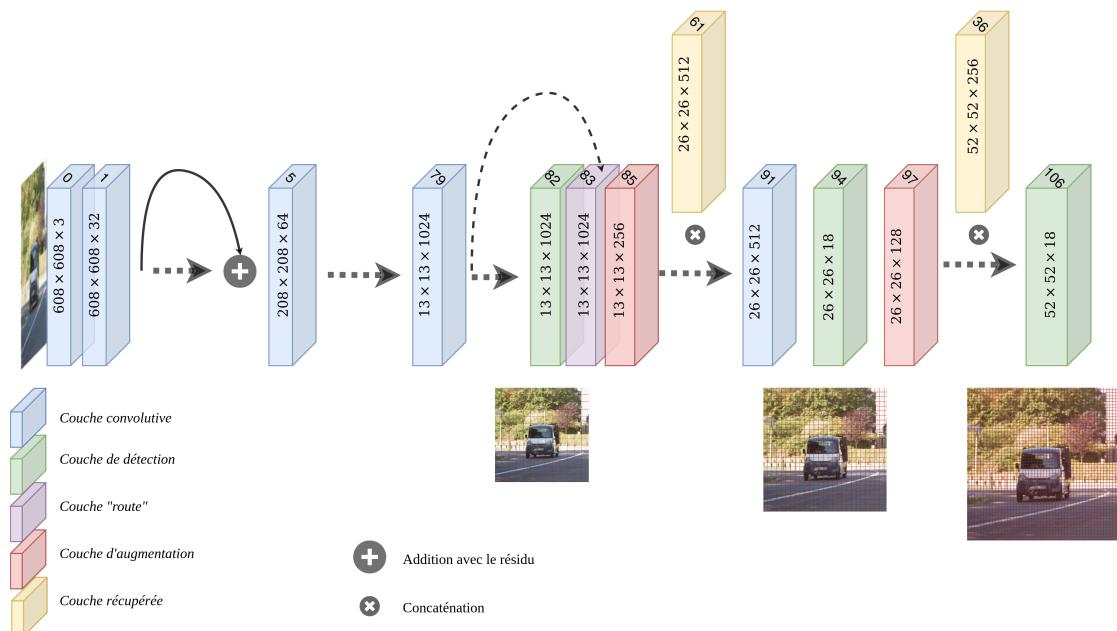


FIGURE 3.3 – Représentation simplifiée de l'architecture du réseau YOLOv3. La dimension des données d'entrée est affichée sur le côté de chaque couche. La dimension de sortie doit correspondre à la couche suivante.

La FIGURE 3.3 représente l'agencement typique des couches de neurones dans le cas de l'architetcure de YOLOv3. Avant la première couche de détection (couche 82), les caractéristiques sont extraites par des couches convolutives. Les données de sortie de certaines couches sont

### 3.3. You Only Look Once : principe de fonctionnement de l'algorithme

---

additionnées avec l'entrée de couches précédentes. Cela permet par exemple de ne pas perdre de l'information si certaines couches de neurones ont des valeurs de sortie très faibles, voir même nulles. De plus, l'entraînement est accéléré, car les couches plus profondes peuvent être entraînées par l'information qui saute les couches précédentes. Ce type d'architecture est appelé *Residual Network*, et permet aussi d'augmenter la stabilité du réseau [24].

À la suite de chaque couche d'augmentation de données, une concaténation avec des données antérieures est faite. L'information est ainsi augmentée en même temps que la dimension des données.

Il est aussi parfois fait mention de *pooling*. Cette opération est une réduction de la dimension des données, que ce soit avec une convolution ou selon un principe de réduction de données plus classique. La valeur maximale, minimale ou encore moyenne d'un groupe de pixels peut être prise. À noter que, contrairement à la convolution, les paramètres d'un *pooling* par réduction simple des données, ne peuvent pas être appris lors de l'entraînement.

#### 3.3.3 Définition de la fonction de pertes

Afin d'entraîner le réseau, il est nécessaire de définir une fonction de perte à minimiser. Dans le cadre de *YOLO*, la stratégie est simplement une minimisation de l'erreur par moindres carrés, avec cependant quelques petites subtilités. La fonction optimisée lors de l'entraînement est la suivante :

$$\begin{aligned}
 & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} \left[ (\hat{t}_{x,i} - t_{x,i})^2 + (\hat{t}_{y,i} - t_{y,i})^2 \right] \\
 & + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} \left[ (\hat{t}_{w,i} - t_{w,i})^2 + (\hat{t}_{h,i} - t_{h,i})^2 \right] \\
 & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} (\hat{t}_{0,i} - t_{0,i})^2 \\
 & + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{noobj} (\hat{t}_{0,i} - t_{0,i})^2 \\
 & + \sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{i \in classes} (\hat{p}_i(c) - p_i(c))^2
 \end{aligned}$$

où  $\hat{t}_*$  dénote la grandeur estimée par le réseau,  $\mathbb{1}_i^{obj}$  vaut 1 si  $obj$  apparaît dans la case  $i$ ,  $\mathbb{1}_{ij}^{obj}$  dénote que la boîte de référence  $j$  est responsable de l'objet dans la case  $i$ . Deux paramètres d'équilibrage ont été introduits :  $\lambda_{coord}$  et  $\lambda_{noobj}$ . Le but de ces paramètres est de donner plus de poids aux vrais positifs qu'aux vrais négatifs. En effet, sans équilibrage, et comme

beaucoup de zones de l'image sont vides, le réseau pourrait ne jamais détecter d'objets tout en ayant une valeur de pertes raisonnable. Dès lors,  $\lambda_{coord}$  vaut 5, et  $\lambda_{noobj}$  vaut 0.5.

L'observation du second terme de la fonction de pertes, et de la définition de  $t_w$  et  $t_h$ , fait ressortir un second principe d'équilibrage. En effet, des erreurs de dimensions équivalentes sur des petits objets sont plus graves que sur des grands objets. Par exemple, une erreur de 10 pixels sur un objet qui en fait 200 de large, ne devrait pas être pénalisée de la même manière qu'une erreur de 10 pixels sur un objet qui en fait 20. Dans la première version de l'algorithme, la fonction de perte minimisait la différence des racines carrés de la largeur et de la hauteur des boîtes. Ce système a été abandonné dans les dernières versions, et est remplacé par une définition adéquate de  $b_w$  et  $b_h$ . En inversant la définition de  $b_w$ , il vient :

$$t_w = \ln\left(\frac{b_w}{p_w}\right) = \ln(b_w) - \ln(p_w)$$

De cette manière, les erreurs sont prises en compte de manière moins importante pour les boîtes de grandes dimensions.

#### **3.3.4 Une version optimisée pour la vitesse : *tiny-YOLO***

La version de l'algorithme optimisée pour la vitesse, nommée *tiny-YOLO*, compte seulement 24 couches de neurones. Le principe est exactement le même que celui de l'algorithme complet, la seule différence étant la profondeur du réseau. Le *mAP* obtenu reste pourtant supérieur à 52.7 %, et la vitesse de calcul est augmentée de presque 10 fois, sur le *Jetson Nano*®.

Dans le cadre de ce projet, il a été décidé d'utiliser la version 3 de *tiny-YOLO*. Une phase de tests préliminaires a permis d'observer que, même sans entraînement spécifique (utilisation d'un réseau entraîné avec un jeu de données génériques), la version complète de l'algorithme atteint une précision et un rappel élevés, sur les données du cas d'étude. Dès lors, il semble que la version complète soit inutilement complexe pour l'utilisation qui en serait faite. De plus, l'entraînement et la détection avec le réseau réduit sont beaucoup plus rapides.

Dans la suite du document, il sera parfois fait mention de *YOLO* ou de *yolov3*, en parlant de la version 3 de l'algorithme *tiny-YOLO*.

# 4 Entraînement du réseau de détection et de classification des objets

La méthode de détection et de classification des objets ayant été choisie, il convient ensuite d'entraîner les couches convolutives du réseau de neurones, afin d'obtenir les meilleurs résultats possibles. La stratégie d'entraînement adoptée pour les données vidéo et les données radar diffère selon quelques aspects. L'entraînement ayant été réalisé avec les données des deux capteurs est donc expliqué séparément, dans la suite de ce chapitre.

## 4.1 Stratégie d'entraînement avec les données vidéo

Pour l'algorithme associé à la vidéo, il a été décidé de faire un entraînement en deux temps. Dans l'idée de garder une certaine capacité d'adaptation, une première étape consiste en l'entraînement du réseau avec des données générales, et sans redéfinir les *anchor boxes*. Par la suite, dans le but d'avoir des meilleures performances avec les données du cas d'étude, le réseau est réglé avec précision. Uniquement les données capturées sur le lieu d'étude sont alors utilisées, et les *anchor boxes* redéfinies.

### 4.1.1 Élaboration des jeux de données d'entraînement

Pour commencer, il est nécessaire de définir les différentes classes d'objets que l'algorithme doit être capable de détecter et de classifier. En évaluant rapidement la diversité des objets présents dans le jeu de données capturé sur le lieu d'étude, douze classes d'objets ont été définies. Ces classes sont les suivantes : *sedan*, *personne*, *camion*, *SUV (sport utility vehicle)*, *camionnette*, *vélo*, *moto*, *bus*, *trottinette*, *véhicule ucl léger*, *remorque* et *chien*. Un exemple de chaque classe est présenté à la FIGURE 4.1. Dans l'idée d'utiliser la taille des objets pour évaluer leur distance, les véhicules ont été rassemblés en classes selon leur similarité, mais aussi selon leur hauteur apparente. Seulement 12 types d'objets ont été définis, car il est nécessaire de trouver un bon compromis pour le nombre de classes. En effet, trop peu de classes limiteraient la quantité d'information donnée lors d'une détection. À l'inverse, trop de classes demanderaient énormément d'images d'entraînement, et une définition très précise des

caractéristiques des objets au sein d'une même catégorie.

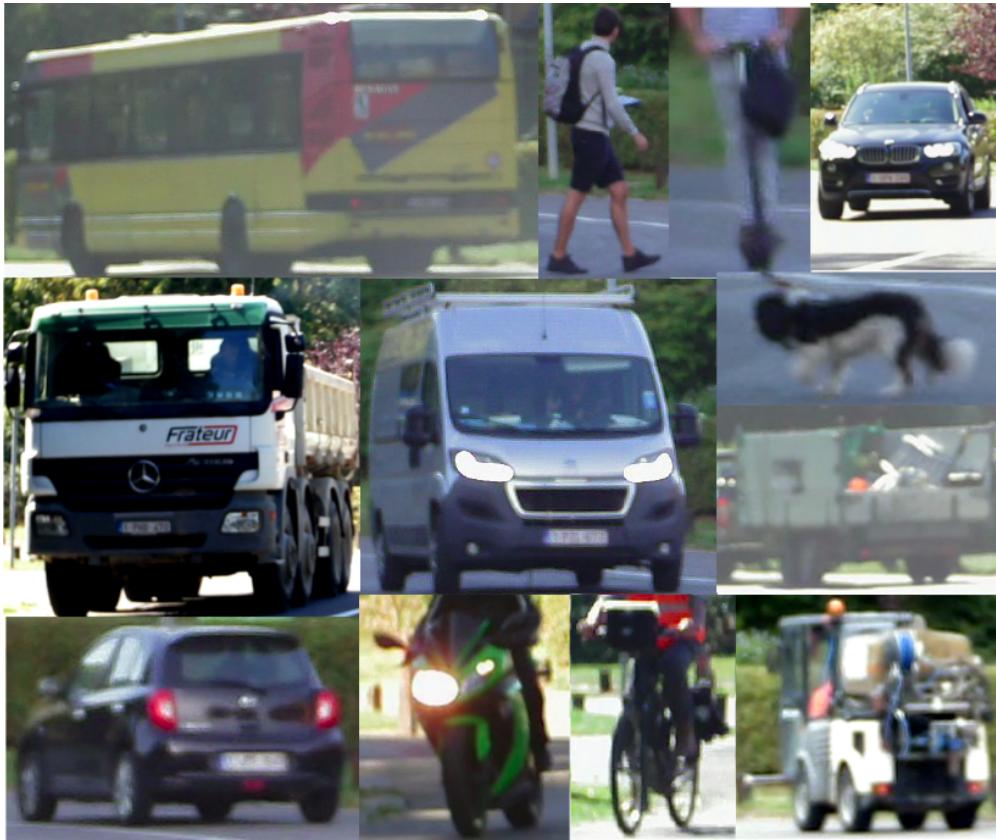


FIGURE 4.1 – Un objet représentatifs de chaque classe. De gauche à droite et de haut en bas : bus, personne, trottinette, SUV, camion, camionnette, chien, remorque, sedan, moto, vélo et véhicule UCL

Même si, théoriquement, il n'y a pas de nombre minimum d'images d'entraînement par classe, il est conseillé de respecter la règle suivante : le jeu de données d'entraînement doit comprendre au moins un objet de chaque classe se présentant dans chaque configuration possible, c'est-à-dire sous toutes ses formes, ses luminosités, ses rotations, ses tailles relative et sous tous ses angles de vue. Il est aussi nécessaire que l'arrière plan change si ce sera le cas pour les données d'intérêt. En suivant ces règles, il faut idéalement environ 2000 images annotées par classe. Ces conseils sont donnés par Alexey AB, très actif et reconnu par la communauté, et peuvent être retrouvés sur son github [25]. Le problème qui se pose très vite est celui du temps que l'annotation des données prend. En effet, il faut tout d'abord trouver et sélectionner les images intéressantes. Il faut ensuite annoter chaque objet un par un. En comptant en plus de tout cela les pertes de temps associées à la manipulation d'un jeu de données importants, il est raisonnable de compter quasiment une minute de travail par image prête pour l'entraînement. Cela représente donc un investissement de temps d'environ 33 heures par classe, si 2000 images sont annotées.

#### **4.1. Stratégie d'entraînement avec les données vidéo**

---

Dans le cadre d'un travail comme celui-ci, il n'est donc pas envisageable d'annoter des milliers d'images pour chaque classe d'objet. Il est dès lors nécessaire, soit d'utiliser des bases de données open source, soit de réaliser une augmentation de données. Beaucoup de bases de données annotées existent, comme par exemple *COCO dataset* [26], *Pascal voc dataset* [27], *kitti dataset* [28]. Le problème qui se pose assez rapidement avec ce genre de *dataset* très général, est que les classes d'objet y sont, soit très abstraites, soit au contraire très spécifiques. Par exemple, seule une classe "*voiture*" y sera définie, comprenant les SUV, les sedan et les camionnettes. Au contraire, chaque type de voiture peut y être différencié, jusqu'au modèle exacte du véhicule. Une étape de conversion des classes est donc parfois inévitable.

Dans une certaine mesure, les données peuvent aussi être augmentées, c'est-à-dire que certaines images du jeu de données sont utilisées pour en générer d'autres. Par exemple, les contraste, la luminosité ou même les couleurs d'une image peuvent être changés afin d'en générer une nouvelle. Des transformations géométriques sont aussi possibles, et dans certains cas un changement d'arrière plan de l'image peut être réalisé. La précision peut ainsi être augmentée sans passer trop de temps à annoter des images<sup>1</sup>. Dans le cas de l'algorithme *YOLO*, le *dataset* peut être augmenté directement lors de la phase d'entraînement, via quelques paramètres à régler. Tous les paramètres d'entraînement sont expliqués à la Section 4.1.2. À ce stade, il faut simplement retenir qu'aucune augmentation de données n'est nécessaire avant la phase d'entraînement.

Dans le cadre de ce travail de fin d'études, il a été décidé d'utiliser le jeu de données *COCO*, en y redéfinissant certaines classes. D'autre part, un maximum d'images provenant des prises de données sur le lieu d'étude ont été annotées. Le *dataset COCO* compte pas moins de 80 classes d'objets différentes, dont les *personnes*, les *voitures*, les *camions*, les *vélos*, les *motos*, les *bus* et les *chiens*. Les classes *voiture* et *camion* comprennent les objets de type *camion*, *sedan*, *SUV* et *remorque*. Afin de modifier les labels de manière adéquate, un script python a été développé dans le cadre de ce travail. Chaque objet est présenté à l'opérateur, et la classe de l'objet est modifiée en fonction de la touche clavier appuyée. Cela permet de ne passer que quelques secondes sur chaque véhicule, mais cela demande un certain niveau de concentration. Malheureusement, le dataset *COCO* ne comporte pas de trottinettes et relativement peu de *remorques*. De plus, ce type d'objet n'a été observé que rarement sur le lieu d'étude. Afin d'avoir suffisamment d'images représentant ces objets, il a été décidé d'utiliser le moteur de recherche de *GOOGLE* et d'annoter ensuite quelques images téléchargées.

Pour finir, 60 séquences vidéos ont été choisies parmi les données capturées. Ces séquences vidéos seront utilisées pour l'entraînement et les tests des résultats, mais pas pour l'analyse finale des performances de détection sur le lieu d'étude. Le soin a été pris de sélectionner des séquences enregistrées à toute heure de la journée, afin d'avoir une luminosité variable. Cependant, aucune image capturée de nuit n'a été utilisée. Les nombres d'objets de chaque classe, ainsi que la provenance des images, sont repris dans le TABLEAU 4.1.

---

1. Pour plus d'informations sur l'efficacité de l'augmentation de données consulter : <http://cs231n.stanford.edu/reports/2017/pdfs/300.pdf>

Classe	Label	COCO	Google	Dispositif	Total
0	Voiture	15 154	195	1445	16 794
1	Personne	89 202	444	565	90 211
2	Camion	1 844	15	427	2 286
3	SUV	964	36	333	1 333
4	Camionnette	407	36	488	931
5	Vélo	2 474	54	115	2 643
6	Moto	3 088	42	101	3 231
7	Bus	2 029	9	116	2 154
8	Trottinette	0	216	57	273
9	Véhicule UCL	0	0	175	175
10	Remorque	23	165	68	256
11	Chien	1 954	3	15	1 972

TABLEAU 4.1 – Nombre d'objet de chaque classe dans le jeu de données d'entraînement pour la caméra

Parmi les données d'entraînement, 10 % des images ont été sélectionnées pour tester les performances de l'algorithme. Ces images ne seront jamais utilisées pour entraîner le réseau, mais il est à noter que le réseau pourra être entraîné avec des images provenant de séquences vidéo identiques.

Pour chaque image du jeu de données, un fichier texte de même nom reprend les informations de localisation et de classe des objets présents sur l'image. Le format imposé par *Yolo* est le suivant :

0 0.246093 0.656944 0.495312 0.686110

où le premier numéro se réfère à la classe d'appartenance de l'objet, et les numéros suivants aux informations liées à la boîte entourant cet objet. Ces dimensions sont les coordonnées  $(x, y)$  du centre, la largeur  $w$  et la hauteur  $h$  de la boîte, normalisées en fonction des dimensions de l'image. La normalisation permet redimensionner l'image sans avoir à modifier le fichier texte associé. L'ordre des éléments est le suivant :

$$classe \quad \frac{x}{w_{image}} \quad \frac{y}{h_{image}} \quad \frac{w}{w_{image}} \quad \frac{h}{h_{image}}$$

où  $w_{image}$  et  $h_{image}$  représentent respectivement la largeur et la hauteur de l'image. Dans l'unique fichier texte associé à une image, une ligne doit être écrite par objet. Un dernier détail imposé par l'algorithme est que les images doivent être enregistrées au format *jpg*.

Pour conclure avec l'élaboration du *dataset* d'entraînement, il peut être utile de rappeler

quelques bonnes pratiques données par *Alexey AB* [25]. Premièrement, aucun objet appartenant à une classe doit se trouver sans annotation associée dans le jeu de données d'entraînement. De plus, la manière dont les objets vont être entourés par les boîtes lors de l'annotation déterminera la façon dont elles seront détectées par l'algorithme. Autrement dit, si un espace est laissé entre les objets et les boîtes lors de l'annotation, un espace apparaîtra lors de la détection. Dans le cas présent, aucun espace n'est laissé, afin de détecter la hauteur des objets de la manière la plus précise possible. Finalement, il est parfois utile de mettre dans le *dataset* des images sans aucun objet (avec un fichier texte vide), afin de réduire le nombre de faux positifs. L'arrière plan du lieu d'étude étant statique, il serait dommage de ne pas en profiter.

Les données d'entraînement étant prêtes, les paramètres de configuration de l'algorithme *YOLOv3-tiny* doivent être définis.

### 4.1.2 Choix des paramètres d'entraînement

Le choix des paramètres d'entraînement de l'algorithme *Yolo* se fait au moyen d'un fichier de configuration. Ce document reprend différents types de paramètres, ainsi que l'architecture du réseau.

Tout d'abord, cinq valeurs concernant le format des images et la gestion de la mémoire de la carte graphique sont à définir. Le réseau va être entraîné par *batch* d'images, divisés en sous-groupes. Afin de ne pas saturer la mémoire de la carte graphique, il ne faut pas charger trop d'images en une fois. Au contraire, un entraînement plus rapide et plus efficace demande de travailler avec des *batch* de grande taille. Dans le cadre de ce travail, l'entraînement se fait à l'aide de *GOOGLE COLAB*<sup>2</sup>, une plateforme gratuite de *GOOGLE* destinée à la recherche. L'accès à une carte graphique *Nvidia Tesla K80*<sup>®</sup> avec environ 12 GB de mémoire disponibles est proposé, moyennant certaines conditions d'utilisation. Les paramètres suivants ont dès lors été choisis :

- |                    |   |
|--------------------|---|
| — batch = 64       | # nombre d'images par batch                       |
| — subdivisions = 8 | # nombre de sous groupes d'images par batch       |
| — width = 608      | # largeur des images, doit être un multiple de 32 |
| — height = 608     | # hauteur des images, doit être un multiple de 32 |
| — channels = 3     | # nombre de canaux de couleurs de images          |

Les quatre paramètres suivants sont des valeurs déterminant l'augmentation des données. Les différents chiffres indiquent des variations aléatoires autour des valeur de base des images, dans un référentiel de couleur *TSL* (teinte, saturation, luminosité). Il a été décidé de ne pas appliquer de rotation aux images, car les véhicules et les piétons ne devraient pas être inclinés, à condition que le dispositif soit bien fixé. Les autres valeurs reprises ci-dessous sont les valeurs par défaut :

---

2. La démarche suivie a été adaptée de ce tutoriel : <http://blog.ibanyez.info/blogs/coding/20190410-run-a-google-colab-notebook-to-train-yolov3-using-darknet-in/>

## Chapitre 4. Entraînement du réseau de détection et de classification des objets

---

```

— angle = 0           # angle maximum de rotation des images
— saturation = 1.5   # variation maximum de valeur de saturation
— exposure = 1.5     # variation maximum de variation d'exposition
— hue=.1             # variation maximum des couleurs

```

Certains paramètres un peu plus techniques, concernant l'algorithme du gradient, peuvent aussi être réglés. Il pourrait paraître redondant de définir un paramètre de *batch* et un de *sous groupes* d'images, si ces paramètres n'influençaient que l'utilisation de la mémoire de la carte graphique. En réalité, l'algorithme *Yolo* est entraîné par une méthode dite de "descente de gradient par mini-*batch*" [29]. Le gradient est alors calculé sur des lots de 64 images, et les poids sont actualisés. Le paramètre de taille des lots se réfère donc à la méthode du gradient, et le paramètre du nombre de sous groupes permet de ne pas saturer la mémoire. L'algorithme de convergence ne sera pas exposé ici en détail, seule une rapide explication des paramètres suffit pour comprendre leur influence. En résumé, il s'agit de minimiser la fonction de perte définie à la Section 3.3.3. Pour ce faire, les paramètres du modèle sont modifiés à chaque itération, selon les équations suivantes :

$$\begin{cases} v_{i+1} = & \text{momentum} \cdot v_i - \text{decay} \cdot \epsilon \cdot w_i - \epsilon \cdot \nabla L|_{w_i, D_i} \\ w_{i+1} = & w_i + v_{i+1} \end{cases}$$

où  $i$  est l'indice d'itération,  $v$  est la variable de moment,  $\epsilon$  est le taux d'apprentissage,  $L$  la fonction de pertes et  $w$  la règle d'actualisation des paramètres ([30], Section 5). L'indice  $D_i$  indique que le gradient est calculé sur le lot d'images  $i$ . Le paramètre *momentum* ajoute une sorte d'inertie, à l'instar de celle d'un objet dévalant une pente. Le *decay* permet de limiter l'influence de l'inertie, en fonction du vecteur de modification des paramètres. À noter que le gradient est soustrait car la direction de la pente descendante est obtenue selon l'opposé du gradient.

```

— momentum = 0.9          # inertie du moment
— decay = 0.0005          # limitation de l'inertie du moment
— learning_rate = 0.001    # taux d'apprentissage
— burn_in = 1000          # nombre d'itérations avant le taux d'apprentissage
— max_batches = 75000     # nombre maximum d'itérations
— policy = steps          # méthode d'actualisation du taux d'apprentissage
— steps = 400000, 450000   # itérations où le taux d'apprentissage est actualisé
— scales = .1,.1          # facteur de multiplication du taux d'apprentissage

```

Le paramètre de *burn\_in* définit le nombre d'itération avant lesquelles le taux d'apprentissage atteindra sa valeur maximale. En effet, le modèle est susceptible de diverger si l'entraînement commence à un taux d'apprentissage élevé, selon les créateurs de l'algorithme *Yolo* ([21], paragraphe 2.2). Finalement, un paramètre *max\_batches* limite le nombre maximum d'itération, qui doit être au moins égal à 2000 fois le nombre de classes à entraîner, selon Alexey AB [25]. Les paramètres suivants permettent de gérer l'actualisation du taux d'apprentissage, en le divisant par 10 à la 400000 et à la 450000 itération par exemple. Cela permet de modifier les

#### 4.1. Stratégie d'entraînement avec les données vidéo

paramètres de manière plus précise en fin d'entraînement, c'est-à-dire proche d'un minimum local. Dans le cadre de ce travail, il a été décidé de garder les valeurs par défauts de tous ces paramètres, car leur choix se base principalement sur des critères empiriques. Seul le nombre maximum d'itérations a été changé.

Pour finir, il convient de modifier certaines valeurs au sein même de l'architecture du réseau. En effet, le nombre de neurones de certaines couches est lié au nombre de classes. Il faut par exemple définir le nombre de filtres avant chaque couche *Yolo* en fonction du nombre de classes. Pour chaque classe, il y a 5 paramètres et trois *anchor boxes* définies à une certaine échelle.

```
— filters = 51          # (nombre de classes + 5)*3  
— classes = 12         # nombre de classes différentes  
— random = 1           # re-dimensionner aléatoirement les images
```

Le paramètre *random* repris ci-dessus indique que l'algorithme va re-dimensionner aléatoirement les images, de manière à rendre la détection invariante vis à vis de la résolution des images et des objets.

Dans certains cas, il est aussi nécessaire de modifier les rapports de dimension des *anchor boxes*. Dans le cadre de ce travail de fin d'études, il a été décidé de garder les dimensions par défaut lors d'une première étape d'entraînement, et de les recalculer lors d'une seconde étape.

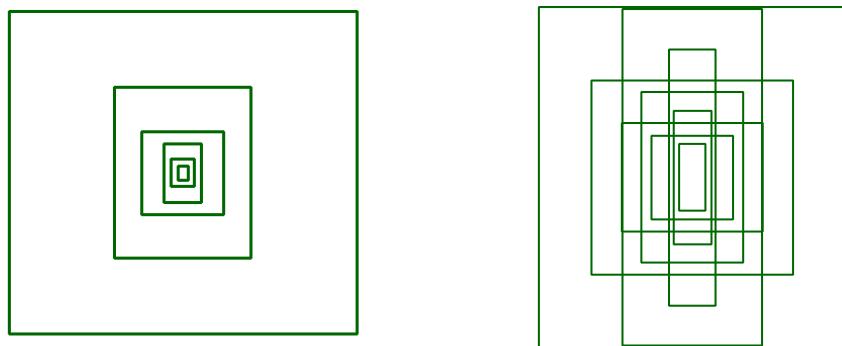


FIGURE 4.2 – Représentation des *anchor boxes* pour l'entraînement du réseau *YOLO*. À gauche, les valeurs par défauts. À droite, les valeurs obtenues à l'aide des données du cas d'étude.

Lors de la première étape d'entraînement, avec les données très générales du *dataset COCO*, les hauteurs et les largeurs de *anchor boxes* sont celles par défaut, définies de la manière suivante :

*anchors* = 10,14, 23,27, 37,58, 81,82, 135,169, 344,319

Dans un deuxième temps, un algorithme d'apprentissage non-supervisé (*k-mean*) a permis de calculer de nouvelles boîtes plus adaptées au données du cas d'étude. L'algorithme récursif rassemble les objets en 9 différents groupes, sur base d'un critère de distance par rapport à un centroïde. Dans le cas présent, le critère de distance est le rapport entre l'intersection et l'union de deux boîtes, ou *IoU*.

*anchors = 39, 99, 56, 198, 121, 124, 69, 380, 209, 161, 151, 253, 299, 288, 207, 499, 456, 505*

La FIGURE 4.2 représente les *anchor boxes* par défaut et celles recalculées. Cela permet de se rendre compte de la diversité des objets dans le jeu de données. Il est possible d'imaginer que les boîtes allongées vers le haut correspondent plutôt aux piétons, et qu'à l'inverse les boîtes plus larges sont associées à des véhicules.

## 4.2 Analyse des résultats de détection pour la vidéo

Au fil de l'entraînement, la qualité de l'apprentissage peut être suivie grâce à l'évolution du *mAP* et des pertes, comme définies aux Sections 3.1 et 3.3.3 respectivement. La précision, le rappel et le score F1 apportent d'autres informations utiles, et permettent d'adapter la stratégie d'entraînement.

### 4.2.1 Entraînement avec les images du jeu de données COCO

Lors de l'entraînement avec les données générales du *dataset COCO*, une évolution relativement lente du *mAP* a été observée. En effet, après 50 000 itérations, une valeur de *mAP* d'environ 44 % est atteinte, comme le montre la FIGURE 4.3. Il semble que cette valeur pourrait encore augmenter légèrement, mais il a été décidé de faire un arrêt précoce de l'entraînement. En effet, cela permet d'éviter le phénomène de sur-apprentissage [31]. Même si les données de test sont différentes du jeu de données d'entraînement, un sur-apprentissage pourrait arriver, au moins pour certaines classes. En arrêtant l'entraînement de manière précoce, il est assuré de garder de la flexibilité.

Les pertes évoluent de manière inverse au *mAP*, pour finir par osciller autour d'une valeur d'environ 3,4. La FIGURE 4.4 montre la décroissance en exponentielle inverse des pertes. D'après Alexey AB, la valeur finale des pertes moyennes devrait se situer entre 0.05, pour un petit jeu de données facile, et 3.0, pour un grand jeu de données difficile à entraîner [25]. Une valeur de 3.4 est donc un peu élevée, mais est cohérente avec le choix d'un arrêt précoce de l'entraînement.

Afin de comprendre pourquoi la valeur du *mAP* reste basse, l'étude de la précision et du rappel peut aider. L'évolution des ces grandeurs lors de l'entraînement est représentée sur la FIGURE 4.5.

La précision oscille autour de 60 %, ce qui n'est pas très bon, mais laisse penser que l'entraî-

## 4.2. Analyse des résultats de détection pour la vidéo

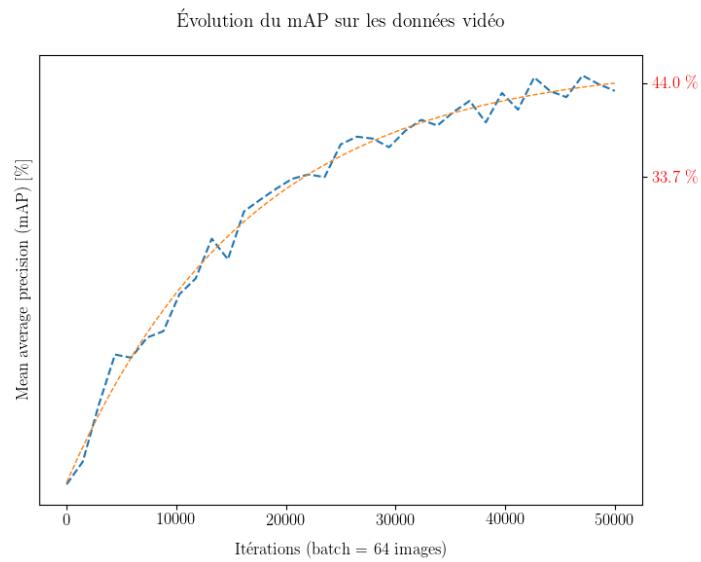


FIGURE 4.3 – Évolution du *mAP* lors de l’entraînement avec le jeu de données COCO.

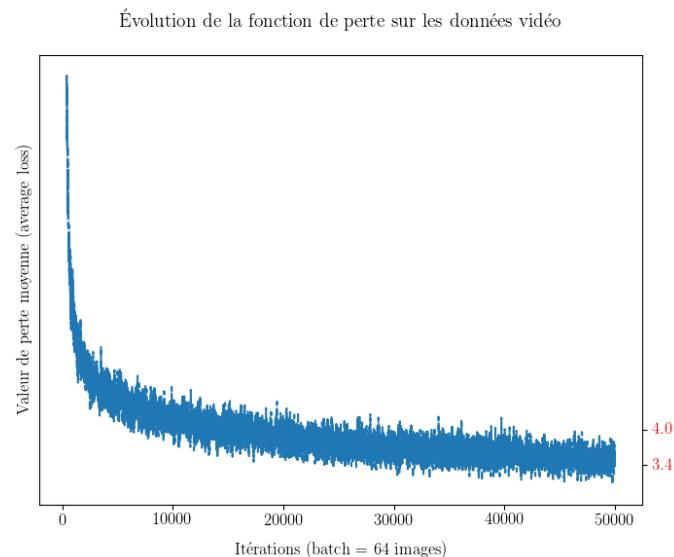


FIGURE 4.4 – Évolution des pertes moyennes lors de l’entraînement avec le jeu de données COCO.

nement fonctionne. En revanche, le rappel est bas, de l’ordre de 35 %, comme le montre la FIGURE 4.5. Il semble donc que les objets ne soit pas bien détectés sur les images. Cela peut s’expliquer en partie parce que le jeu de données comporte beaucoup d’objets de petite taille, partiellement occultés. D’autre part, il se peut que les *anchor boxes* ne soient pas adaptés aux données.

La FIGURE 4.6 montre que le *score-F1* est impacté par la valeur basse du rappel, et n’est donc

## Chapitre 4. Entraînement du réseau de détection et de classification des objets

---

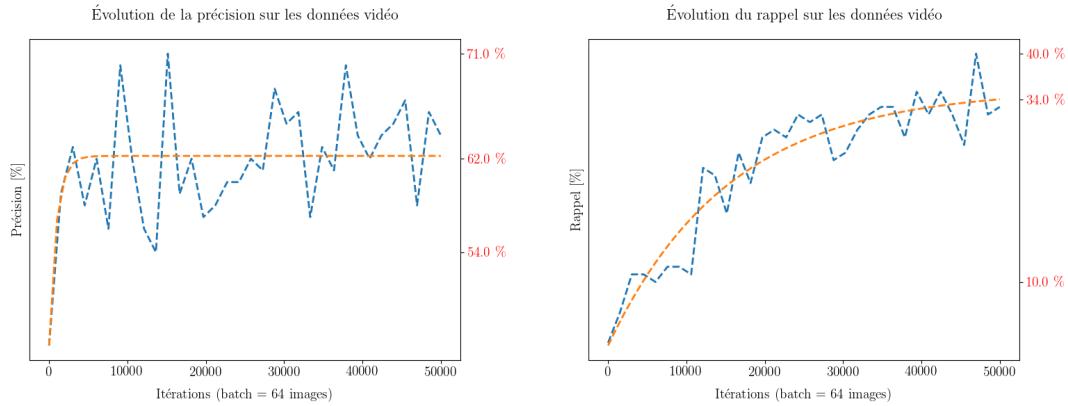


FIGURE 4.5 – Évolution de la précision et du rappel lors de l'entraînement avec le jeu de données *COCO*.

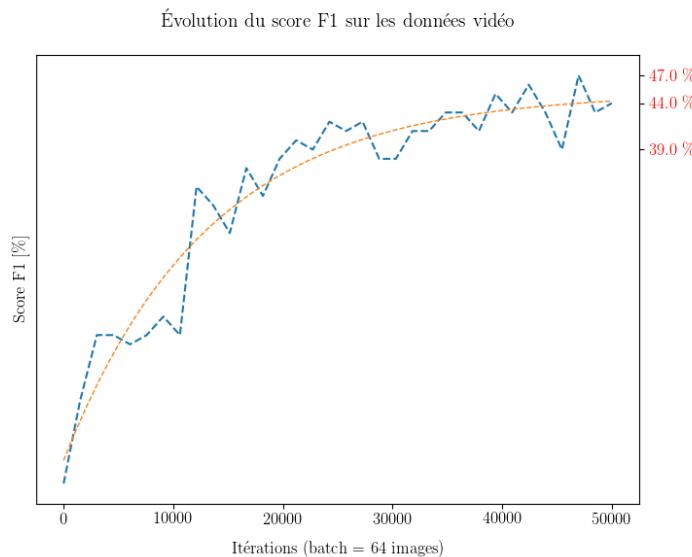


FIGURE 4.6 – Évolution du score F1 lors de l'entraînement avec le jeu de données *COCO*.

que de 45 % environ. Dans le but d'améliorer le rappel, les *anchor boxes* sont redéfinies et le réseau *Yolo* est entraîné avec les données du cas d'étude uniquement.

### 4.2.2 Entraînement avec les images du jeu de données du cas d'étude

Dès les premières itérations, la valeur du *mAP* monte en flèche, pour finalement osciller autour de 82.6 %, comme le montre la FIGURE 4.7. La valeur de *mAP* semble donc se stabiliser assez vite, même si les pertes moyennes continuent à descendre, pour atteindre une valeur de 0.3 environ. La FIGURE 4.8 représente l'évolution des pertes, et semble indiquer qu'elles pourraient encore baisser un peu. L'entraînement est pourtant stoppé, car la valeur *mAP* n'aug-

## 4.2. Analyse des résultats de détection pour la vidéo

mente plus. Pour cette deuxième étape d'entraînement, il a été décidé de ne pas faire d'arrêt précoce. En effet, les données de test ne sont pas les mêmes que les données d'entraînement donc le risque de sur-entraînement est faible. De plus, toutes les données provenant du lieu d'étude sont relativement similaires, donc le réseau ne devra pas faire preuve de beaucoup de flexibilité.

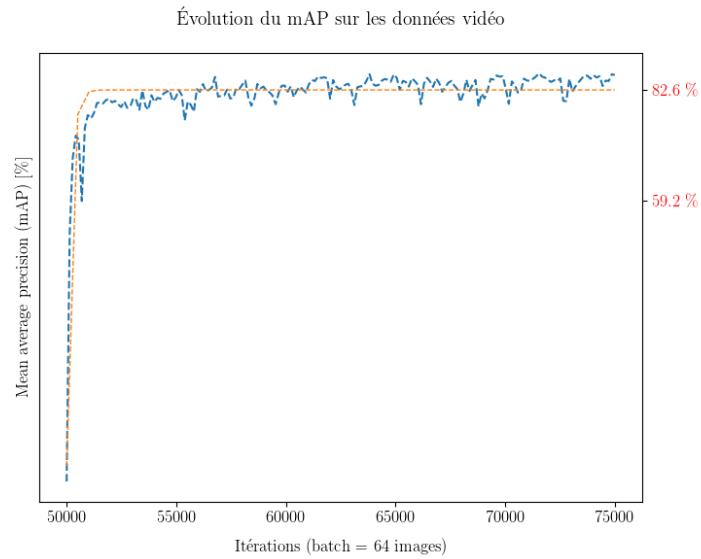


FIGURE 4.7 – Évolution du *mAP* lors de l'entraînement avec le jeu de données du cas d'étude.

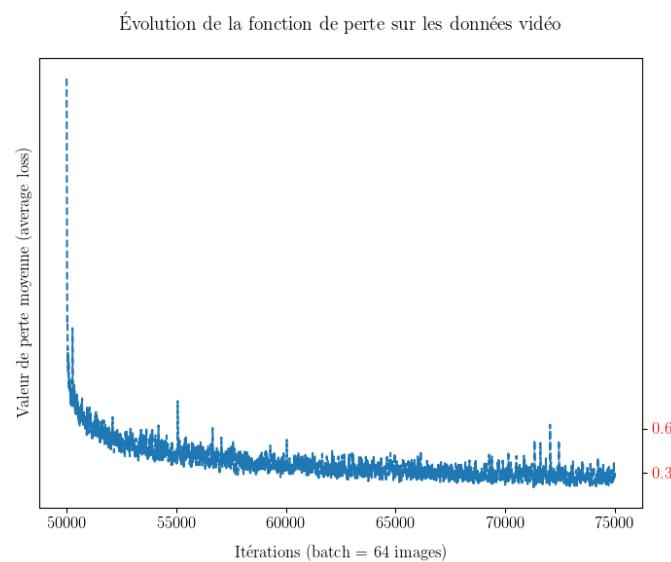


FIGURE 4.8 – Évolution des pertes moyennes lors de l'entraînement avec le jeu de données du cas d'étude

La FIGURE 4.9 montre une nette amélioration de la précision et du rappel. En effet la précision passe d'environ 60 % à plus de 80 %, et le rappel double quasiment. Les objets sont donc

## Chapitre 4. Entraînement du réseau de détection et de classification des objets

---

détectés sur les images dans beaucoup plus de cas. Il semble donc que redéfinir les *anchor boxes* ait été bénéfique pour la détection des objets. Il faut cependant rester prudent vis-à-vis de ce résultat, car le jeu de données d'entraînement a été modifié.

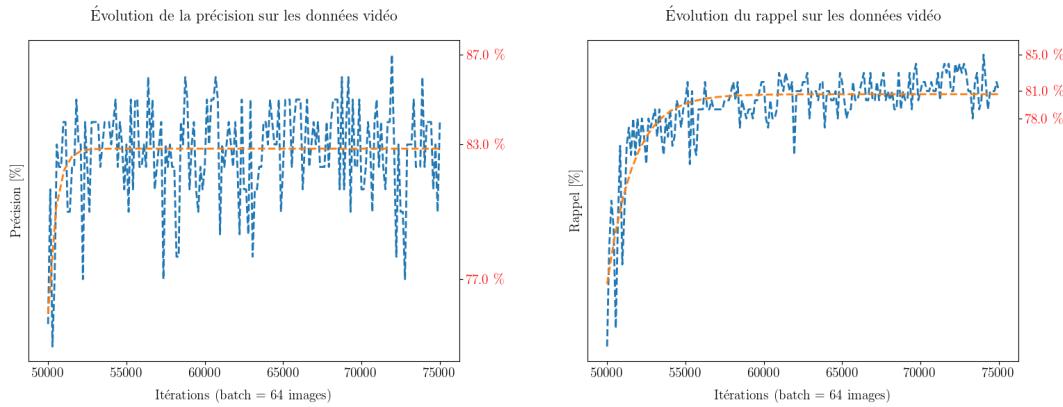


FIGURE 4.9 – Évolution de la précision et du rappel lors de l'entraînement avec le jeu de données du cas d'étude

L'évolution de la précision et du rappel impactent directement la valeur du *score-F1*, comme il est possible de l'observer à la FIGURE 4.10. Il semble aussi que les oscillations de la valeur du *score-F1* sont plus rapides que dans le premier cas, tout en étant moins amples. Cela peut s'expliquer par le fait que le *dataset* est beaucoup plus petit et plus homogène que lors de la première étape d'entraînement.

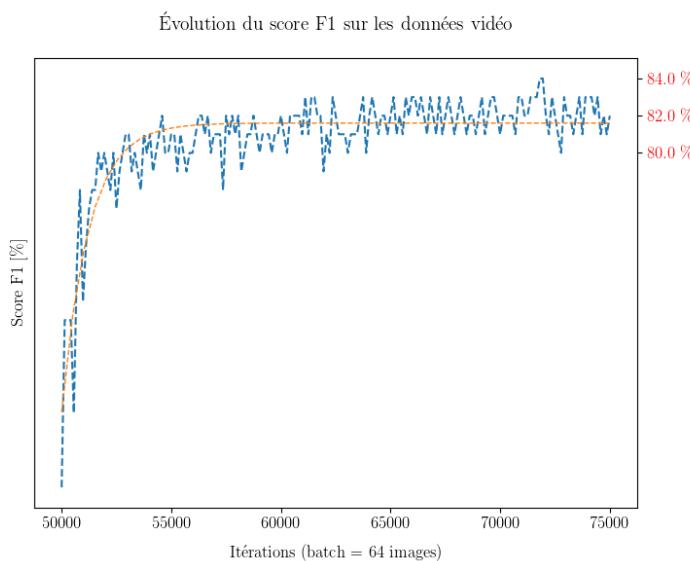


FIGURE 4.10 – Évolution du score F1 lors de l'entraînement avec le jeu de données du cas d'étude

À l'issue des deux étapes d'entraînement, le *score-F1* est d'environ 82 %, pour des valeurs de

### **4.3. Optimisation des paramètres de seuil pour la vidéo**

---

précision et de rappel presque équivalentes.

Afin d'améliorer encore un peu la détection et la classification des objets, il a été décidé d'utiliser un système de fenêtre glissantes. En effet, les images enregistrées par la caméra sont beaucoup plus larges que hautes, donc un re-dimensionnement en images de  $608 \times 608$  pixels de résolution impacte beaucoup la qualité. La résolution native des images étant de  $1280 \times 720$ , le rapport de réduction des images serait d'environ 2.1. En revanche, en choisissant de décomposer les images en trois fenêtres de  $720 \times 720$  pixels, le rapport de réduction n'est plus que de 1.2 environ. Ainsi, la qualité des images peut être préservée.

Certaines difficultés apparaissent cependant avec l'utilisation de fenêtres glissantes. Premièrement, le temps de calcul est multiplié par quatre, car il est nécessaire de faire une détection par fenêtre, plus une détection sur l'image complète. En effet, les gros objets, comme par exemple les camions, occupent quasiment toute l'image, et ne peuvent donc pas être détectés sur une fenêtre unique. En plus d'une augmentation considérable du temps de calcul, certains objets sont détectés plusieurs fois, à des emplacements légèrement différents. Il est donc nécessaire de filtrer les faux positifs associés à l'utilisation des fenêtres.

## **4.3 Optimisation des paramètres de seuil pour la vidéo**

Afin d'améliorer les performances du détecteur, une stratégie d'optimisation des paramètres associés à l'utilisation des fenêtres glissantes a été développée. Tout d'abord, six paramètres de seuil ont été définis.

### **4.3.1 Définition des paramètres de seuil**

**Le seuil de recouvrement pour une détection unique :** un objet peut être détecté sur plusieurs fenêtres en même temps, avec des boîtes légèrement différentes. Lors de la fusion des données, plusieurs boîtes entourent l'objet, comme sur la FIGURE 4.11 (a). Le seuil concerne donc l'*indice de Jaccard* de l'aire des deux boîtes, à condition qu'elles se réfèrent à un objet de même classe. Si le seuil est trop haut, plusieurs boîtes vont rester et des objets fantômes risquent d'apparaître. Au contraire, si le seuil est trop bas, les objets de même classe trop proches les uns des autres seront filtrés. Par exemple, des piétons marchants en groupes risquent de ne pas bien être détectés.

**Le seuil de recouvrement pour une détection coupée :** un objet partiellement présent sur une des fenêtres glissantes peut donner lieu à une détection erronée. Sur la FIGURE 4.11 (b), une des trois fenêtres glissantes a détecté le bord gauche de la voiture. En connaissant le pas des fenêtres, il est possible de détecter les boîtes partielles. Un nouveau seuil, l'*indice de Jaccard* est utilisé comme seuil de détection. En revanche, il a été décidé de ne pas tenir compte de la classe, car les détections partielles ont tendance à mener à des erreurs de classification. Par

exemple, le côté d'une voiture sera parfois détecté en tant que camionnette, mais devra tout de même être filtré. Des objets de classe différente étant trop proches pourraient donc être filtrés, mais il faudrait pour cela que l'un soit placé exactement au bord d'une des fenêtres glissantes. Une autre possibilité aurait été de ne pas prendre en compte tous les objets au bords des fenêtres glissantes.

**Le seuil de recouvrement pour un objet unique :** un objet peut être détecté sur deux fenêtres différentes mais identifié comme appartenant à des classes différentes. La FIGURE 4.11 (c) montre les cas d'un petit camion, classé en tant que camionnette et en tant que camion. À nouveau, cette situation est déterminée sur base de l'*indice de Jaccard* des aires des boîtes entourant l'objet. Le choix de l'objet se fait ensuite à l'aide de l'indice de confiance de la détection. Dans le cas représenté sur la FIGURE 4.11 (c), un camion devra être détecté car la probabilité que l'objet soit une camionnette n'est que 7 %. Le seuil ne doit pas être trop bas afin de pouvoir détecter les objets proches ou l'occultant.

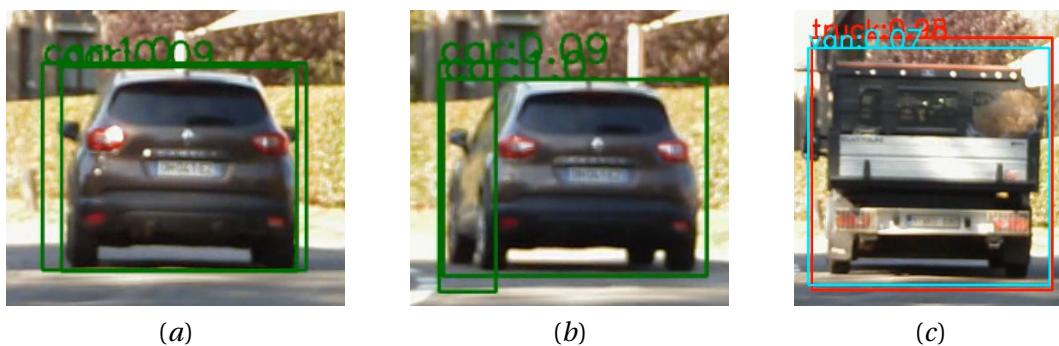


FIGURE 4.11 – Situations de recouvrement des boîtes à corriger

**La pondération des probabilités :** il a été décidé de ne pas prendre complètement en compte l'indice de confiance des détection provenant des fenêtres. Par exemple, un objet détecté sur une seule fenêtre sera pénalisé. Cela permet de prendre en compte les détections faites sur une seule fenêtre sans y accorder trop d'importance.

**Le facteur d'augmentation de l'indice de confiance :** lorsqu'un objet est détecté plusieurs fois, la probabilité que la détection soit vraie augmente. La nouvelle probabilité  $p$  vaut :

$$p = \frac{p_1 + p_2}{\alpha}$$

où  $p_1$  et  $p_2$  sont les indices de confiance de la détection 1 et 2, respectivement, et  $\alpha$  est le paramètre à optimiser. Si  $\alpha$  vaut 2, la moyenne des deux probabilités est simplement prise.

### 4.3. Optimisation des paramètres de seuil pour la vidéo

**Le seuil d'indice de confiance :** finalement, les faux positifs restant sont filtrés sur base des indices de confiance associés aux détections. Ce paramètre est important, car s'il est trop bas la précision chute, mais s'il est trop haut, le rappel devient très faible.

L'optimisation des 6 paramètres définis précédemment n'est pas évident. En effet, si les paramètres peuvent prendre 10 valeurs différentes chacun, 1 000 000 d'itérations sont nécessaires afin de calculer toutes les combinaisons possibles. Il n'est donc pas possible de simplement trouver les paramètres optimaux en calculant toutes les possibilités. Dès lors, l'influence des paramètres doit être étudié afin de trouver le meilleur compromis possible.

#### 4.3.2 Influence des paramètres de seuil

Tout d'abord, il est à noter qu'une méthode d'optimisation sur quelques paramètres n'est pas optimale. En effet, une stratégie plus élaborée ferait appel à un apprentissage supervisé et à davantage de paramètres. Il serait pas exemple possible de définir différents paramètres en fonction des classes. Une démarche plus poussée, avec une définition plus complète du problème, est décrite dans [32]. Dans le cadre de ce travail, il a été décidé de s'en tenir à un algorithme relativement simple, afin de ne pas focaliser l'étude sur ce problème en particulier.

Dans un premier temps, une étude de l'influence des paramètres a été faite. Afin de limiter le temps de calcul, les valeurs ont été étudiées en deux groupes de trois. D'une part, les trois paramètres basés sur l'*indice de Jaccard* ont été changés, en gardant les trois paramètres liés aux indices de confiance constants. D'autre part, les trois premiers paramètres ont été fixés, et les seuils de probabilités ont alors été changés. Le TABLEAU 4.2 reprend les données des différents calculs.

Dans un second temps, tous les paramètres ont été changés en même temps, mais dans des intervalles de valeurs beaucoup plus petit.

Paramètre	S. 1	S. 2	S. 3	S. 4	S. 5
IoU détection unique	[0, 1.0]	0.5	[0, 1.0]	[0.90, 1.00]	[0.94, 0.99]
IoU détection coupée	[0, 1.0]	0.85	[0, 1.0]	[0.40, 0.50]	[0.42, 0.47]
IoU objet unique	[0, 1.0]	0.5	[0, 1.0]	[0.75, 0.85]	[0.79, 0.84]
Ponderation fenêtres	1.0	[0, 1.0]	[0, 1.0]	[0.05, 0.15]	[0.07, 0.12]
Augmentation de la probabilité	2.0	[0, 2.0]	[0, 2.0]	[0.70, 0.80]	[0.74, 0.79]
Seuil d'indice de confiance	0.0	[0, 1.0]	[0, 1.0]	[0.25, 0.35]	[0.27, 0.32]
Nombre d'itérations	9621	9621	46 656	46 656	46 656

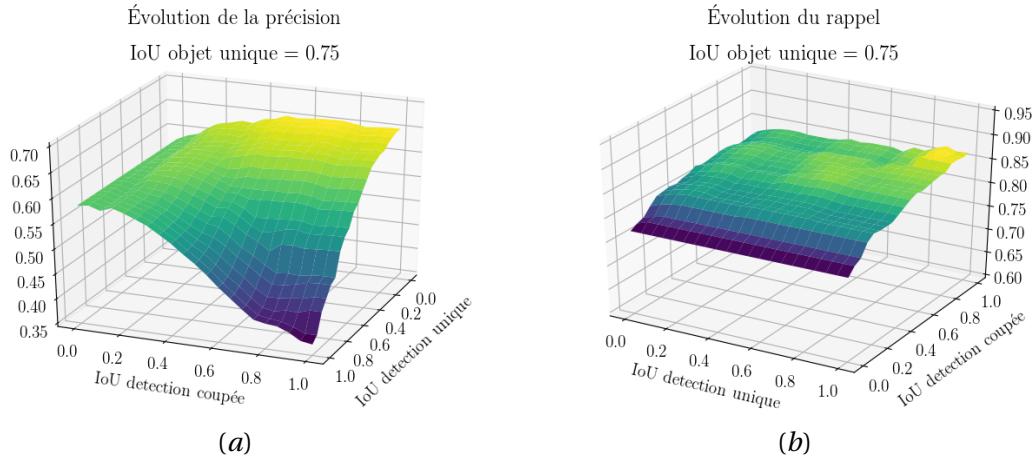
TABLEAU 4.2 – Paramètres de simulation lors de l'optimisation des seuils

La FIGURE 4.12 (a) représente l'influence des trois premiers paramètres sur la précision. Il peut être observé que, si les seuil deviennent trop importants, la précision chute rapidement. En effet, un seuil trop haut fera apparaître des objets fantômes, ce qui fait chuter la précision. Le seuil pour le recouvrement d'objets coupés par une fenêtre doit préféablement être petit,

## Chapitre 4. Entraînement du réseau de détection et de classification des objets

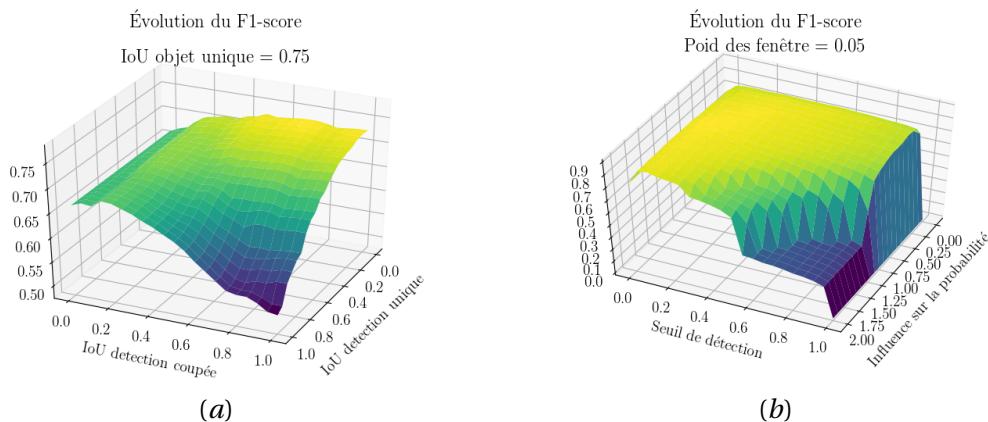
---

alors que celui pour les objets détectés plusieurs fois, peut être un peu plus grand. Afin d'augmenter la précision, il est possible de faire varier les trois paramètres de seuil d'indice de confiance. Dès lors, il convient plutôt de faire attention au rappel, qui ne pourra pas être augmenté sur base de l'indice de confiance.



La Figure 4.12 (b) représente l'évolution de la valeur du rappel en fonction des trois premiers paramètres. À l'inverse de la précision, les seuils de détection ne doivent pas être trop bas, et le rappel ne varie pas beaucoup en fonction des seuils d'*indices de Jaccard*. Il est pourtant important de bien prendre en compte l'influence de ces paramètres, car ils sont le seul moyen d'optimiser le rappel.

Finalement, l'évolution du score-F1, reprise sur la FIGURE 4.13 (a), reflète simplement les variations du rappel et de la précision. Le rappel variant assez peu, la forme de la surface est très similaire à celle de la précision.



Sur les graphiques précédemment décrits, la valeur du seuil de détection d'un objet unique a

### **4.3. Optimisation des paramètres de seuil pour la vidéo**

---

été fixée à 0.75. En modifiant cette valeur, l'évolution des surfaces reste similaire, avec quelques distorsions. L'ANNEXE A.1 rassemble les graphiques représentants l'influence de ce paramètre. De manière identique, la pondération des probabilités des fenêtres a été fixée à 0.05 lors de l'étude de l'influence des trois derniers paramètres. L'influence de la pondération des probabilités peut également être observée en ANNEXE A.1.

La FIGURE 4.13 (b) montre l'influence du seuil d'indice de confiance et du facteur d'augmentation de l'indice de confiance, sur le score-F1. De manière générale, l'augmentation du seuil de détection augmente la précision mais diminue le rappel. Lorsque ce seuil est à 1, le score-F1 tombe subitement à zéro. En effet, l'indice de confiance des objets est toujours inférieur ou égal à 100 [%]. Si le paramètre d'augmentation d'indice de confiance augment trop et que le seuil de détection est relativement haut, le rappel chute. Cela est logique, en se souvenant que l'augmentation de l'indice de confiance en cas de détections multiples est inversement proportionnelle au paramètre qui lui est associé.

L'observation de l'évolution des surfaces précédemment décrites permet de comprendre l'influence des paramètres, mais rend difficile la localisation de maxima pour les six paramètres en même temps. Dès lors, il a été décidé de faire varier tous les paramètres en même temps, mais sur des intervalles de valeurs plus petits. Les paramètres des trois simulations ainsi réalisées sont repris dans les trois dernières colonnes du TABLEAU 4.2.

L'optimisation a été faite, sur base du score-F1, afin de s'assurer d'avoir un bon compromis entre la précision et le rappel. Sur la FIGURE 4.14, le score-F1 semble atteindre sa valeur maximale lorsque le seuil de détection se trouve entre 20 [%] et 60 [%]. En faisant une nouvelle simulation autour du pic principale, la valeur optimale du seuil se précise. La FIGURE 4.15 montre l'évolution du score-F1 sur un intervalle réduit du seuil de détection. En resserrant successivement l'intervalle, il a été trouvé que la valeur optimale de ce paramètre est d'environ 29 [%].

Une stratégie par réduction de l'intervalle de variation des paramètres a été faite pour chaque paramètre. Ainsi, uniquement une petite partie de toutes les valeurs possibles des paramètres est testée. Il est possible qu'un pic se trouve dans un intervalle de valeurs inexploré, mais cela semble peu probable compte tenu de la régularité des surfaces calculées précédemment. Autrement dit, les paramètres semblent faire évoluer le score F1 de manière assez monotone, et l'apparition d'un pic élevé au milieu des valeurs faibles semble peu probable. L'optimisation de chacun des six paramètres peut être trouvée dans l'ANNEXE A.2.

#### **4.3.3 Choix des paramètres de seuil**

Sur base de l'étude de l'influence des paramètres, et du calcul des maxima de manière itérative, les valeurs des paramètres maximisant le score-F1 ont pu être trouvées. Le TABLEAU 4.3 reprend les résultats important, atteints grâce à la stratégie d'optimisation.

Les résultats tendent à montrer que le seuil de validation d'une détection unique peut être

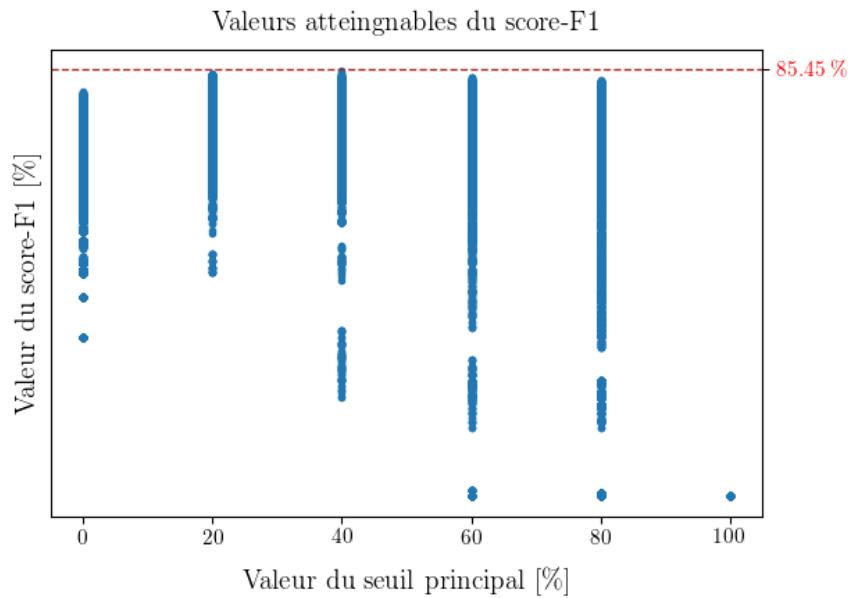


FIGURE 4.14 – Valeurs possibles du score-F1 en fonction du seuil de détection

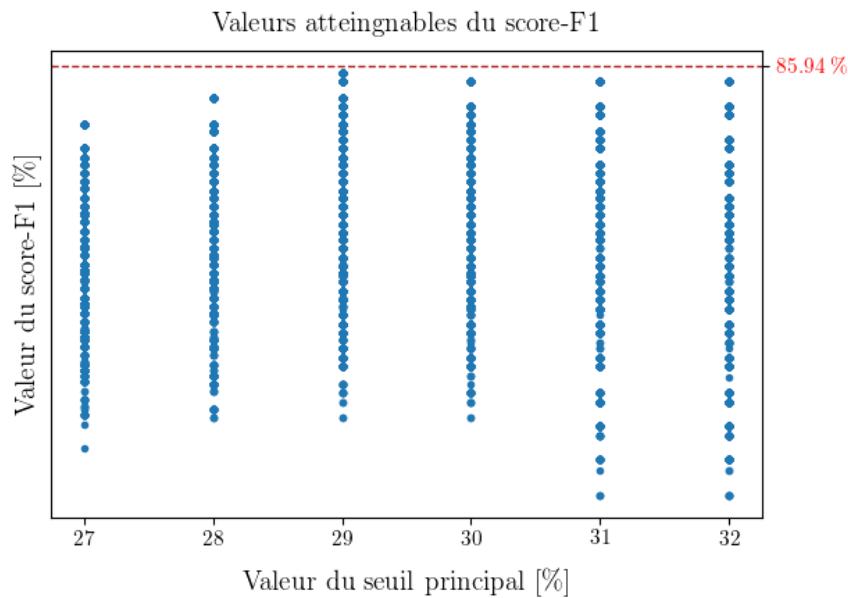


FIGURE 4.15 – Valeurs possibles du score-F1 sur un intervalle réduit du seuil de détection

élevé. Cela indique que l'algorithme détecte les objets de manière presque identique sur les différentes fenêtres. En revanche, les objets coupés semblent être détectés avec une localisation moins précise. Deux objets sont considérés les mêmes si leur *indice de Jaccard* est supérieur

#### 4.4. Stratégie d'entraînement avec les données radar

IoU détection unique	0.95	Précision avec fenêtres glissantes [%]	88.14
IoU détection coupée	0.44	Rappel avec fenêtres glissantes [%]	83.85
IoU objet unique	0.80	Score-F1 avec fenêtres glissantes [%]	85.94
Pondération de la probabilité	0.1	Précision sans fenêtres glissantes [%]	84.37
Augmentation de la probabilité	0.75	Rappel sans fenêtres glissantes [%]	83.08
Seuil de l'indice de confiance	0.29	Score-F1 sans fenêtres glissantes [%]	83.72

TABLEAU 4.3 – Résultats de l'optimisation des paramètres de détection pour la caméra

à 80 %, ce qui semble être un bon compromis entre rappel et précision. Il est intéressant de constater que l'indice de confiance des fenêtres est pris très peu en compte, mais que le facteur d'augmentation de la probabilité augmente beaucoup confiance, lors des détections multiples. Avec les paramètres trouvés, l'augmentation de l'indice de confiance est en fait quasiment uniquement dû au facteur de division de la moyenne, et non à la probabilité de l'objet sur une fenêtre glissante. Pourtant, pour que la moyenne ait lieu, il faut au moins deux détections différents du même objet. Finalement, le seuil de l'indice de confiance, appliqué en dernier lieu, est relativement bas. Cela peut s'expliquer par le fait que, même sans seuil, l'algorithme obtient déjà une précision élevée. Les faux positifs restants ont donc un indice de confiance souvent très bas.

Pour finir, il est pertinent de s'interroger sur ce qu'apporte l'utilisation de fenêtres glissantes. Pour cela, il est possible de s'appuyer sur les valeurs de précision, rappel et score-F1, avec et sans l'utilisation de fenêtres glissantes. Les résultats des deux méthodes sont repris dans la partie droite du TABLEAU 4.3. Étonnamment, l'apport des fenêtres glissantes se fait plutôt au niveau de la précision, et non du rappel. En effet, presque 4 % de précision sont gagnés alors que le rappel n'augmente que de 0.8 % environ. L'idée d'utiliser une meilleure résolution d'image afin de détecter les objets plus petits ne semble pas fonctionner dans le cas présent. Il est aussi possible que l'augmentation du rappel impacte de manière trop importante la précision. Le score-F1 augmente quand à lui d'environ 2 %. Compte tenu de ces résultats, il semble réaliste d'utiliser un algorithme avec fenêtres glissantes si la puissance de calcul de l'ordinateur le permet, mais de se contenter d'un algorithme classique dans le cas d'un système embarqué fonctionnant en temps réel.

## 4.4 Stratégie d'entraînement avec les données radar

Contrairement au cas des données vidéos, aucun *dataset* en libre accès sur internet n'a été trouvé. Dès lors, seulement les données capturées sur le lieu d'étude ont été utilisées. Les *anchor boxes* ont été redéfinies dès le début de l'entraînement.

Du fait de l'utilisation du jeu de données du cas d'étude uniquement, moins d'images annotées ont été disponibles. Dès lors, il a été décidé de ne définir que 3 classes d'objets différents. Les camions et les bus sont catégorisés en tant que *gross objets*. Les *objets moyens* comportent les

voitures, les camionnettes, les véhicules ucl et les remorques. La troisième catégorie, celle des *petits objets*, comporte les personnes, les vélos, les motos, les trottinettes et les chiens.

L'entraînement avec les données radar se fait donc en une seule étape, sur base des paramètres donnés donc la section suivante.

### 4.4.1 Choix des paramètres d'entraînement

La démarche de sélection des paramètres d'entraînement pour le radar est identique à celle réalisée pour la caméra. Par conséquent, chaque paramètre ne va pas être expliqué à nouveau dans cette section.

En l'absence d'images en couleurs, l'augmentation de données est désactivée. Le nombre maximum d'itérations est tout d'abord fixé à 10 000. Le choix de ne pas dimensionner les images est justifié parce que toutes les images radar sont de même dimension. D'autre part, l'hypothèse est faite que, la taille relative des pics aident à la classification des objets. En redimensionnent les images, les grands pics pourraient se trouver avoir la même taille que les pics plus petits. Finalement, les *anchor boxes* sont redéfinies sur base des données d'entraînement.

```
— channels = 1          # nombre de canaux de couleurs de images
— angle = 0             # angle maximum de rotation des images
— saturation = 0        # variation maximum de valeur de saturation
— exposure = 0          # variation maximum de variation d'exposition
— hue= 0                # variation maximum des couleurs
— max_batches = 10000   # nombre maximum d'itérations
— filters = 24           # (nombre de classes + 5)*3
— classes = 3            # nombre de classes différentes
— random = 0             # re-dimensionner aléatoirement les images
— anchors = 12, 14, 17, 20, 25, 25, 40, 24, 33, 35, 50, 35, 59, 51, 93, 33, 101, 54
```

## 4.5 Analyse des résultats de détection pour le radar

Le point d'inflexion du *mAP* et des pertes moyennes est atteint à environ 1500 itérations, comme le montre la FIGURE 4.16. En fin d'entraînement, la valeur du *mAP* oscille autour de 71 % environ, et celle des pertes moyennes autour de 0.6. L'entraînement avec les données radar semble donc fonctionner, même si la convergence semble moins importante qu'avec les images vidéos. À noter que les précisions moyennes (*AP*) de la détection des petits, moyens et grands objets sont respectivement de 65.14 %, 74.05 % et 79.02 % en fin d'entraînement. Les objets de grande taille sont donc détectés plus facilement. En réalité, il est probable que ce soit surtout les petits objets se déplaçant à faible vitesse, presque masqué par les bruit

## 4.5. Analyse des résultats de détection pour le radar

important autour de la vitesse nulle, qui soient difficilement détectables. Lors de l'annotation des graphiques Doppler, il a en effet parfois été difficile de repérer ces objets, même pour un opérateur humain.

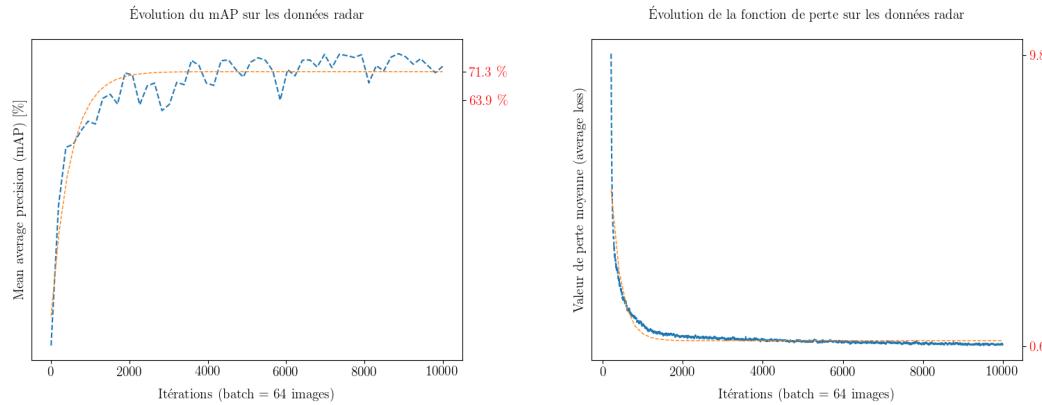


FIGURE 4.16 – Évolution du *mAP* et des pertes moyennes lors de l'entraînement avec les données radar

L'évolution de la précision et du rappel est représentée sur la FIGURE 4.17. La précision augmente rapidement, pour finir autour de 71 %, avec cependant des oscillations d'amplitude relativement importantes. Le rappel connaît une évolution un peu moins abrupte dans un premier temps, mais finit par osciller autour de 69 %. Compte tenu de l'amplitude des oscillations de la précision et du rappel, il est possible que les données d'entraînement soit un peu trop hétérogènes pour le taux d'apprentissage choisi. Il serait dès lors possible d'adapter le taux d'apprentissage, ou de changer la taille de *batch*, comme préconisé dans [33]. Pourtant, les pertes moyennes évoluent de manière régulière, ce qui indique que l'algorithme converge. Autrement dit, le minimum local visé ne semble pas être dépassé.

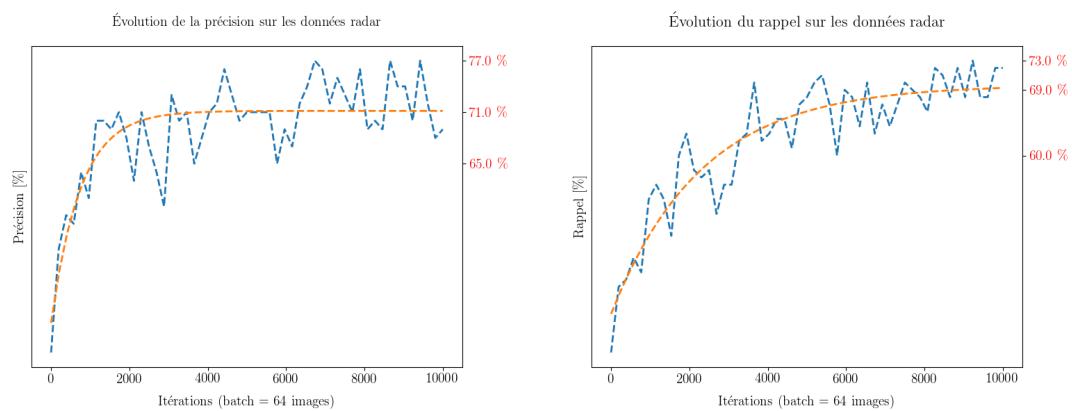


FIGURE 4.17 – Évolution de la précision et du rappel lors de l'entraînement avec les données radar

Pour finir, l'observation de l'évolution de score-F1, représenté sur la FIGURE 4.18, reflète bien

celle de la précision et du rappel. Dès lors, sa valeur en fin d'entraînement est d'environ 70 %. Les oscillations après 6000 itérations semblent un peu moins violentes que dans le cas de la précision et du rappel. Cela est cohérent avec la convergence de l'algorithme, visible grâce aux pertes moyennes. Autrement dit, il semble que les paramètres plus globaux, tel que les pertes moyennes et le score-F1, oscillent moins que les indicateurs partielles, tel que la précision et le rappel. Dès lors, il semble que l'entraînement s'est bien déroulé.

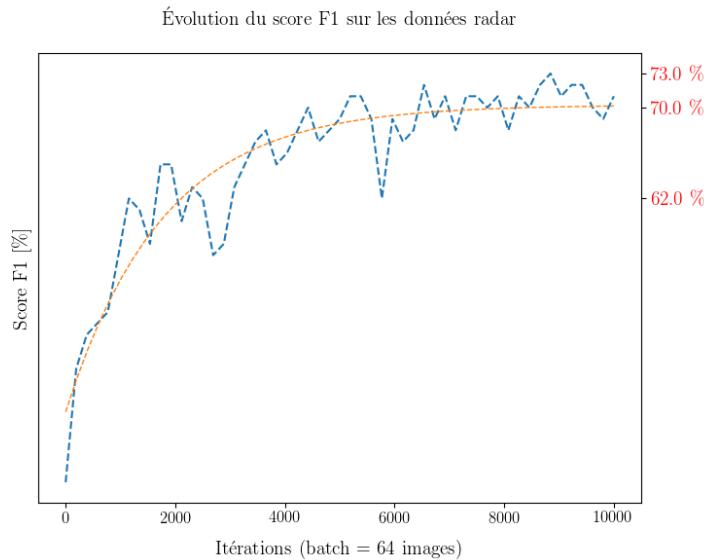


FIGURE 4.18 – Évolution du score F1 lors de l'entraînement avec les données radar

En osant une théorie un peu plus poussée, il est possible que l'augmentation du rappel sans détérioration majeure de la précision soit difficile dans le cas de données radar, car la structure du bruit est parfois semblable à des petits pic représentants des objets bien réels. le problème était peut-être moins présent dans le cas des données vidéos, car il est peu probable que des objets du contexte de fond ressemblent fortement à des véhicules ou des personnes.

Afin de gagner un peu en précision, un paramètre est à optimiser. Il s'agit du seuil d'indice de confiance minimale validant une détection. L'enjeu de l'optimisation est bien entendu de gagner en précision, sans trop perdre en rappel.

## 4.6 Optimisation du paramètre de seuil pour les détections radar

Afin de choisir le seuil d'indice de confiance, il est utile d'observer l'évolution de la précision et du rappel en fonction du seuil. La FIGURE 4.19 représente la précision en fonction du rappel, lorsque le seuil de détection évolue. Il semble que la précision chute un peu à partir de 60 % de seuil, et fortement au-delà de 40 %.

Il a été décidé de garder une valeur de rappel la plus haute possible, sans pour autant perdre trop de précision. En effet, les faux positifs pourront encore être filtrés lors de la fusion des

## 4.6. Optimisation du paramètre de seuil pour les détections radar

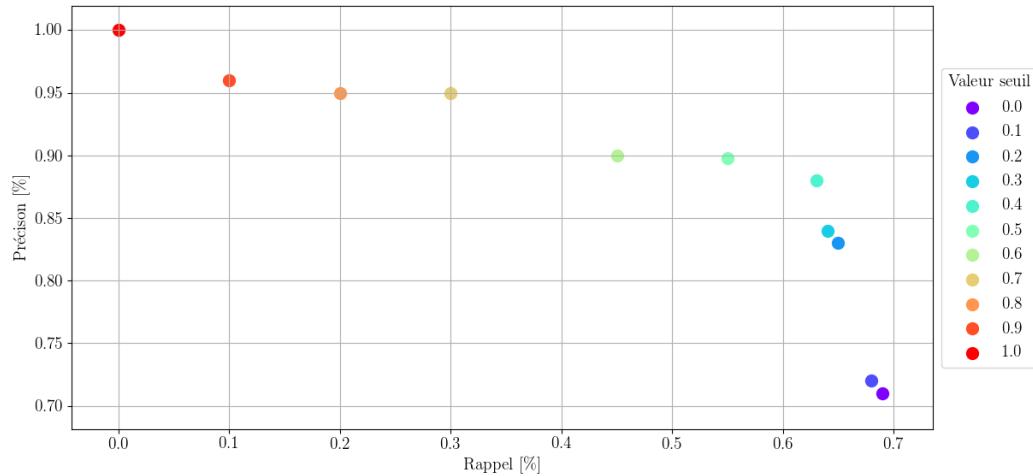


FIGURE 4.19 – Évolution de la précision en fonction du rappel

Seuil	Précision	Rappel	Score F1
40 [%]	88.24 [%]	63.31 [%]	73.72 [%]

TABLEAU 4.4 – Paramètres de détection des objets avec le radar

données. En revanche, si le rappel est trop bas, les objets risquent de ne pas être détectés, ce qui impliquerait une importante imprécision quand à la distance et à la vitesse. La grosse différence avec les vidéos provenant de la caméra est le nombre d'images par seconde. En effet, la caméra prend suffisamment d'images par secondes pour pouvoir se permettre de favoriser la précision par rapport au rappel. Au contraire, le radar prend juste assez d'images par secondes, et les objets doivent donc être bien détectés. Le TABLEAU 4.4 reprend les informations liées à la détection des objets avec le radar.

Tous les paramètres liés à la détection des objets ayant été choisis, les données provenant des deux capteurs doivent à présent être fusionnées. Pour cela, il est tout d'abord nécessaire de filtrer le bruit à haute fréquence superposé aux détections.



# 5 Stratégie de fusion des données et de suivi des cibles

Afin de fusionner les données provenant de la caméra et du radar, il a été décidé d'utiliser une variante du filtre de Kalman. Le choix s'est porté sur ce type de filtre pour plusieurs raisons. Premièrement, même si un filtre à particules peut être plus efficace dans le cas du suivi de cibles multiples [34][35], l'implémentation d'un filtre de Kalman est plus facile, donc potentiellement plus robuste. De plus, le filtre de Kalman nécessite bien moins de puissance de calcul qu'un filtre à particules. Finalement, la fusion des données du radar et de la caméra peut se faire en parallèle au filtrage du bruit, sur base d'un algorithme unique.

Dans le cas présent, plusieurs défis sont à surmonter, le principal étant l'association des pistes avec les données des capteurs. En effet, les capteurs ne mesurent pas les mêmes dimensions, et il est donc difficile d'associer les mesures du radar avec celles de la caméra. De plus, différentes cibles peuvent être présentes à un même instant. Il s'agit donc de mettre en place une stratégie de suivi de cibles multiples, avec la fusion de données provenant de capteurs hétérogènes.

Dans un premier temps, il a été décidé d'adapter la démarche réalisée dans [36]. La démarche mathématique (Sections 5.1 et 5.2), ainsi que certaines équations, sont donc directement reprises de [36], tout en prenant soin d'expliquer les résultats trouvés. Le réglage des paramètres est quand à lui adapté au cas étudié dans ce travail de fin d'études, et les résultats obtenus en découlent. L'implémentation de l'algorithme en *Python* est également le fruit d'un travail personnel. Le Chapitre 6 propose des améliorations de l'algorithme, et expose les gains de précision qui en résultent.

## 5.1 Définition de l'état des objets

Afin de décrire l'état des objets à un instant  $k$ , les coordonnées de position et de vitesse dans un référentiel sphérique sont utilisées. Le vecteur d'état, pour tout objet se déplaçant dans le champ de capture du dispositif, est défini comme étant :

$$\vec{x} = \begin{bmatrix} \alpha & \epsilon & r & \dot{\alpha} & \dot{\epsilon} & \dot{r} \end{bmatrix}^T$$

## Chapitre 5. Stratégie de fusion des données et de suivi des cibles

---

où  $\alpha$  [°] est l'angle azimutal,  $\epsilon$  [°] l'angle d'élévation et  $r$  [m] la distance radiale. La vitesse associée à une grandeur quelconque  $a$  sera notée  $\dot{a}$  dans le reste de ce document. Les coordonnées cartésiennes ( $x, y, z$ ) peuvent être calculées sur base des relations suivantes :

$$\begin{aligned}x &= r \cdot \sin(90 - \epsilon) \cdot \sin(\alpha) \\y &= r \cdot \sin(90 - \epsilon) \cdot \cos(\alpha) \\z &= r \cdot \cos(90 - \epsilon) \\\dot{x} &= \dot{r} \cdot \sin(90 - \epsilon) \cdot \sin(\alpha) + r[\dot{\alpha} \cdot \sin(90 - \epsilon) \cdot \cos(\alpha) - \dot{\epsilon} \cdot \cos(90 - \epsilon) \cdot \sin(\alpha)] \\\dot{y} &= \dot{r} \cdot \sin(90 - \epsilon) \cdot \cos(\alpha) - r[\dot{\alpha} \cdot \sin(90 - \epsilon) \cdot \sin(\alpha) - \dot{\epsilon} \cdot \cos(90 - \epsilon) \cdot \cos(\alpha)] \\\dot{z} &= \dot{r} \cdot \cos(90 - \epsilon) - r \cdot \dot{\epsilon} \cdot \sin(90 - \epsilon)\end{aligned}$$

Afin de simplifier les calculs, il a été décidé d'utiliser les coordonnées sphériques lors du calcul des positions et du suivi des objets, et de n'utiliser les coordonnées cartésiennes que pour l'affichage des résultats.

Des pistes sont utilisées afin de suivre le déplacement des objets dans le temps. Chaque piste est construite sur base des informations présentées dans le TABLEAU 5.1.

Paramètres des pistes	
Identifiant unique de la piste	$I_d$
Vecteur d'état	$\vec{x}$
Matrice de covariance	$\mathbf{P}$
Classe d'objet	$C$
Âge de la piste	$A$
Itérations sans détections	$I_{nd}$

TABLEAU 5.1 – Définition des paramètres des pistes

Dans un premier temps, il a été décidé de n'utiliser qu'un seul type de piste, indépendamment du fait que les données de capture peuvent provenir de deux types de capteurs différents.

### 5.2 Description théoriques des étapes du filtre de Kalman

Afin de ne pas surcharger ce document, les explications suivantes ne décrivent pas le fonctionnement des filtres de Kalman standard, fonctionnant avec un seul capteur, et un seul objet à suivre. Au contraire, un filtre d'association probabiliste des données du radar et de la caméra est directement introduit, dans le cas d'un scénario avec plusieurs cibles. Cependant, les différentes grandeurs utilisées seront interprétées, afin d'en comprendre l'utilité, et les implications.

### 5.2.1 Étape de prédiction

À chaque itération de l'algorithme de Kalman, l'état du système au temps  $t + \Delta t$  est prédict. L'accélération des objets n'étant jamais mesurée, elle est considérée comme du bruit. Dès lors, l'état prédict du système à l'état  $k + 1$  est donné par :

$$\tilde{x}_{k+1} = \mathbf{F} \cdot \hat{x}_k + \vec{q} \quad \text{où} \quad \mathbf{F} = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Le vecteur  $\vec{q}$  prédit l'erreur statique associée au modèle. Dans la suite du document,  $\vec{x}$  représente un vecteur connu,  $\tilde{x}$  la prédiction du vecteur  $\vec{x}$ , et  $\hat{x}$  sa valeur estimée.

Chaque piste est aussi associée à une matrice de covariance  $\mathbf{P}$ , représentant l'incertitude globale sur la position de la piste. Comme l'incertitude sur la position augmente au cours du temps, la matrice  $\mathbf{P}$  est mise à jour à chaque pas de temps, selon la règle suivante :

$$\tilde{\mathbf{P}}_{k+1} = \mathbf{F} \hat{\mathbf{P}}_k \mathbf{F}^T + \mathbf{Q}$$

En effet, la matrice de covariance du système doit donner la covariance de  $\hat{x}$  autour de  $\vec{x}$  :

$$\tilde{\mathbf{P}}_{k+1} = E[(\hat{x}_{k+1} - \vec{x}_{k+1})(\hat{x}_{k+1} - \vec{x}_{k+1})^T] = \mathbf{F} \cdot E[(\hat{x}_k - \vec{x}_k)(\hat{x}_k - \vec{x}_k)^T] \cdot \mathbf{F}^T = \mathbf{F} \hat{\mathbf{P}}_k \mathbf{F}^T$$

La matrice  $\mathbf{Q}$  représente l'augmentation de l'incertitude, et est à nouveau modélisée comme étant une matrice de covariance d'un bruit blanc Gaussien. En réalité, cette matrice est aussi la matrice de covariance du vecteur  $\vec{q}$ . L'erreur est donc considérée comme étant un bruit blanc Gaussien, de moyenne pouvant être non nulle.

Dans le cadre de ce travail de fin d'études, plusieurs cibles vont être suivies en même temps. Dès lors, un indice  $i$  est associé à chaque piste :  $\hat{x}_{i,k}, \hat{x}_{i,k+1}, \tilde{\mathbf{P}}_{i,k+1}, \dots$

Dans le cas où il n'y a aucune piste à l'instant  $k$ , l'étape de prédiction est passée, et l'étape de filtrage des pistes est directement réalisée. En effet, cette étape est aussi responsable de la création de nouvelles pistes.

### 5.2.2 Étape d'association

L'étape d'association consiste à associer les détections prisent à l'instant  $k$ , aux pistes déjà existantes. Pour cela, un critère de distance entre deux vecteurs d'état va être défini. Dès lors, une détection sera associée à une piste si elle en est assez proche. Il convient ici de se souvenir que les détections proviennent de deux capteurs différents. De plus, les mesures sont hétérogènes, c'est à dire que le radar et la caméra ne capturent pas les mêmes grandeurs. Par conséquent, il est nécessaire de ne prendre en compte que les dimensions mesurées, lors du calcul de la distance entre les pistes et les détections.

À chaque pas de temps, il y a  $j$  détections et  $i$  pistes. Tout d'abord, il convient de définir l'erreur entre chaque piste  $\tilde{x}_{i,k+1}$ , et chaque mesure  $\vec{z}_{j,k+1}$  :

$$\tilde{\nu}_{k+1}^{i,j} = \vec{z}_{j,k+1} - \mathbf{H}_s \tilde{x}_{i,k+1}$$

La matrice  $\mathbf{H}_s$  permet de sélectionner uniquement les grandeurs mesurées par le capteur  $S$ . Par définition,  $\mathbf{H}_s = \mathbf{H}_c$  pour la caméra, et  $\mathbf{H}_s = \mathbf{H}_r$  pour le radar, avec :

$$\mathbf{H}_c = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{et} \quad \mathbf{H}_r = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Le radar mesure donc uniquement la vitesse et la distance radiale, et la caméra les angles azimutal et vertical. Afin de pouvoir associer les détections du radar avec celles de la caméra, il a été décidé de faire une estimation de la distance des objets, avec la caméra. Dès lors, il est fait comme si la caméra mesurait une distance, mais avec une incertitude importante. L'estimation est faite sur base de la hauteur des objets et de la distance focale.

Afin d'estimer l'erreur totale due à l'incertitude sur la position de la piste, additionnée à celle de la mesure par le capteur  $s$ , la matrice  $\mathbf{S}_i$  est construite :

$$\mathbf{S}_i = \mathbf{H}_s \tilde{\mathbf{P}}_{i,k+1} \mathbf{H}_s^T + \mathbf{R}_s$$

où  $\mathbf{R}_s$  est la matrice de covariance du capteur  $s$ , c'est à dire l'incertitude sur les mesures provenant du ce capteur. La matrice de covariance  $\tilde{\mathbf{P}}_{i,k+1}$  est donc projetée dans les dimensions mesurées par le capteur  $s$ .

Il est à présent possible de calculer la distance probabiliste  $\hat{d}^{i,j}$ , entre la piste  $i$  et la mesure  $j$  :

$$(\hat{d}^{i,j})^2 = (\tilde{v}_{k+1}^{i,j})^T \mathbf{S}_i^{-1} \tilde{v}_{k+1}^{i,j}$$

Il est intéressant d'observer que plus l'incertitude est importante, plus la matrice  $\mathbf{S}_i$  fera diminuer la distance probabiliste. C'est une manière d'assouplir le critère sur la distance, en fonction de l'incertitude sur la position et sur la mesure.

Sur base de la distance probabiliste (ou plutôt de son carré), il est possible de choisir d'associer ou non les détections aux pistes déjà existantes. Une détection  $j$  ne pourra être associée à une piste  $i$ , que si  $(\hat{d}^{i,j})^2 < \beta$ . Dans le cas où plusieurs pistes sont suffisamment proches de la détection, la distance minimum est retenue. Dès lors, les pistes peuvent avoir plusieurs détections associées, mais les détections ne peuvent être associées qu'à une seule piste.

À la fin de l'étape d'association, les détections ont été associées ou non aux différentes pistes. Lors de cette étape, il devrait toujours y avoir au moins une piste, car sinon l'étape de prédiction aurait renvoyé l'algorithme directement à l'étape de filtrage des pistes. En revanche, il est possible qu'il n'y ait aucune détection. Dans ce cas, l'état prédit de chaque piste devient l'état estimé, et l'étape de filtrage des pistes est directement réalisée.

### 5.2.3 Étape d'actualisation

Lors de l'étape d'actualisation, il convient tout d'abord d'adapter l'effet de la prédiction sur l'estimation de l'état suivant, et d'en faire de même pour chaque détection associée aux pistes. Comme les vecteurs  $\tilde{v}^{i,j}$  sont tous supposés suivre des lois normales de moyenne nulle et d'écart type  $|S_i| = \det(S_i)$ , il est possible de calculer la probabilité pondérée de chaque détection  $j$ , associée à la piste  $i$ . En considérant que  $m$  détections sont associées à la piste  $i$ , il vient :

$$p^{i,j} = \frac{e^{-0.5(\hat{d}^{i,j})^2}}{\lambda \frac{1-P_D}{P_D} \sqrt{|\mathbf{S}_i|} + \sum_{l=0}^m e^{-0.5(\hat{d}^{i,l})^2}} ; \quad p^{i,j} \leq 1 \quad \text{et} \quad \sum_i p^{i,j} = 1$$

Cette formule se démontre facilement, sur base de la densité de probabilité d'une loi normale multidimensionnelle. Afin de détecter les faux positifs, un terme a cependant été ajouté :

$$\text{alarme de faux positif} = \lambda \frac{1-P_D}{P_D} \sqrt{|\mathbf{S}_i|}$$

Dès lors, la probabilité qu'aucune détection ne soit correcte peut se calculer de la manière suivante :

$$p^{i,0} = \frac{\lambda^{\frac{1-P_D}{P_D}} \sqrt{|\mathbf{S}_i|}}{\lambda^{\frac{1-P_D}{P_D}} \sqrt{|\mathbf{S}_i|} + \sum_{l=0}^m e^{-0.5(\hat{d}^{i,l})^2}}$$

Les paramètres  $P_D$  et  $\lambda$  permettent de régler le seuil de l'alarme de faux positifs, et seront déterminés à la Section 5.3. En pratique, il faut simplement voir  $p^{i,j}$  comme l'importance donnée à la détection  $j$ , lors de l'actualisation de la piste  $i$ .

L'influence des détections ayant été déterminée, il est aussi nécessaire de définir la pondération entre la prédiction  $\tilde{x}_{i,k+1}$  et les mesures  $\vec{z}_{j,k+1}$ . Pour cela, la matrice  $\mathbf{K}_i$  de gain de Kalman est construite de la manière suivante :

$$\mathbf{K}_i = \tilde{\mathbf{P}}_{i,k+1} \mathbf{H}_s^T \mathbf{S}_i^{-1}$$

En se souvenant que la matrice  $\mathbf{S}_i^{-1}$  donne l'incertitude sur l'erreur entre une piste  $i$  et les mesures d'un certain capteur, il est possible d'interpréter la matrice  $\mathbf{K}_i$ , comme étant le rapport entre l'incertitude du modèle, et celle des mesures. Dès lors, l'estimation de la position peut être calculée de la manière suivante :

$$\hat{x}_{i,k+1} = \tilde{x}_{i,k+1} + \mathbf{K}_i \sum_{j=1}^m p^{i,j} \tilde{v}_{k+1}^{i,j}$$

Si le gain de Kalman est nul, les détections ne sont pas du tout prises en compte. Au contraire, si le gain vaut la matrice identité, les détections sont prises en compte de manière plus importante que la prédiction. En effet, en considérant  $m = 1$  et  $p^{i,1} = 1$ , il vient :

$$\hat{x}_{i,k+1} = \tilde{x}_{i,k+1} + \mathbf{K}_i \tilde{v}_{k+1}^{i,1} = \tilde{x}_{i,k+1} + \mathbf{K}_i (\vec{z}_{j,k+1} - \mathbf{H}_s \tilde{x}_{i,k+1}) = (\mathbf{I}_3 - \mathbf{K}_i) \tilde{x}_{i,k+1} + \mathbf{K}_i \vec{z}_{j,k+1}$$

Dans ce cas, si le gain devient égal à la matrice identité, la prédiction n'est même plus prise en compte.

Il faut aussi actualiser la matrice de covariance des pistes.

$$\begin{aligned} \hat{\mathbf{P}}_{i,k+1} &= p^{i,0} \cdot \tilde{\mathbf{P}}_{i,k+1} \\ &+ (1 - p^{i,0}) (\tilde{\mathbf{P}}_{i,k+1} - \mathbf{K}_i \mathbf{S}_i \mathbf{K}_i^T) \\ &+ \mathbf{K}_i \left( \sum_{j=1}^m p^{i,j} \tilde{v}_{k+1}^{i,j} (\tilde{v}_{k+1}^{i,j})^T - (1 - p^{i,0}) \left( \sum_{j=1}^m p^{i,j} \tilde{v}_{k+1}^{i,j} \right) \left( \sum_{j=1}^m p^{i,j} \tilde{v}_{k+1}^{i,j} \right)^T \right) \mathbf{K}_i^T \end{aligned}$$

### 5.3. Définition des matrices de covariance et des paramètres de seuil

---

Le premier terme de l'équation est simplement l'augmentation de l'incertitude due à la prédiction, pondérée par la probabilité qu'il n'y ait aucune mesure correcte. Le deuxième terme montre que l'incertitude est diminuée proportionnellement au gain  $K_i$ , lorsque des mesures sont associées. Finalement, le troisième terme montre que l'incertitude augmente, si la variance entre les différentes mesures est grande, de nouveau de manière proportionnelle au gain.

Durant cette étape, les pistes auxquelles aucune détection n'a été associée sont simplement adaptées selon la prédiction faite. Le nombre d'itérations sans détection (paramètre  $I_{nd}$  du TABLEAU 5.1) est alors incrémenté de 1. Dans le cas contraire, ce paramètre est remis à 0.

#### 5.2.4 Étape de filtrage des pistes

Lors de la dernière étape du filtre de Kalman, les pistes déjà existantes sont filtrées, et les détections qui n'ont pas été associées donnent lieu à des nouvelles pistes. Un paramètre unique, noté  $I_{max}$ , permet de déterminer quelles pistes supprimer. Si le nombre d'itérations sans détections est plus grand que  $I_{max}$ , alors la piste est supprimée. Dans un premier temps, toutes les détections non associées créent des nouvelles pistes.

Lorsqu'une piste est créée, il faut initialiser le vecteur d'état, ainsi que la matrice de covariance. Dans un premier temps, le vecteur d'état est initialisé avec les valeurs mesurées par les capteurs, en mettant une valeur nulle pour les dimensions inconnues. L'incertitude est initialisée avec la matrice de covariance du capteur, en ajoutant cependant de l'incertitude pour les grandeurs non mesurées.

## 5.3 Définition des matrices de covariance et des paramètres de seuil

Tout d'abord, il est nécessaire de définir le pas de temps,  $\Delta t$ , introduit dans la Section 5.2.1. Il a été décidé d'utiliser un pas de temps constant, de manière à ne pas avoir de matrices de covariance variables. Dès lors, le pas de temps doit être plus petit que la période de capture de la caméra. Cette dernière prenant 30 images par seconde, il a été décidé de faire 100 itérations du filtre de Kalman par seconde. Il est vrai qu'il n'est certainement pas nécessaire d'avoir un taux d'actualisation si rapide, mais l'algorithme de Kalman s'exécute en temps réel malgré le pas de temps choisi. De cette manière, l'erreur temporelle introduite par l'échantillonnage de période  $\Delta t$  reste petite.

Afin d'implémenter l'algorithme, il est aussi nécessaire de définir les valeurs des matrices  $\mathbf{Q}$ ,  $\mathbf{R}_r$ ,  $\mathbf{R}_c$  et du vecteur  $\vec{q}$ . Dans un premier temps, il est considéré que le vecteur  $\vec{q}$  est nul, en faisant l'hypothèse qu'il n'y a pas d'erreur statique sur la position. De plus, le signe des accélérations des objets n'est pas connu. Il serait possible d'avoir une idée de cette accélération, par exemple en considérant que les véhicules ralentissent à l'approche d'un virage. Cependant, il a été décidé de considérer que l'algorithme n'a pas accès à la topologie de la route. Afin de trouver

les matrices  $\mathbf{Q}$ ,  $\mathbf{R}_r$  et  $\mathbf{R}_c$ , il faut estimer l'erreur sur les mesures des deux capteurs, ainsi que l'incertitude liée au modèle.

Pour les deux capteurs, l'erreur est supposée être un bruit blanc Gaussien de moyenne nulle. De plus, il est approximé que la covariance entre les différentes dimensions mesurées est nulle. Les matrices  $\mathbf{R}_r$  et  $\mathbf{R}_c$  seront donc des matrices diagonales.

### 5.3.1 Erreur intrinsèque des capteurs

Concernant le radar, la fiche technique [8] donne une résolution en vitesse de  $1 \text{ km}^{-1}$  et une résolution en distance de 1 m. Ces données sont l'écart-type des erreurs, et il faut donc les éléver au carré pour obtenir la variance. Dans un premier temps, il est considéré que la caméra a une résolution parfaite, et que l'erreur est simplement prise en compte dans l'erreur de positionnement des boîtes détectées.

### 5.3.2 Autres sources d'erreur

Le calcul des erreurs liées au positionnement des boîtes est compliqué, car il fait intervenir des fonctions non-linéaires (tel que *arctan*), et se base sur des paramètres incertains (la hauteur des objets par exemple). Dès lors, il a été décidé de ne pas trouver les covariances des erreurs avec un calcul théorique, mais d'utiliser des données annotées.

Tout d'abord, il convient de définir plus clairement la notion de position des objets. Il va être considéré ici que les variables d'état des objets se réfèrent à la face de l'objet la plus proche du dispositif. Il s'agira donc en général de la face avant ou de la face arrière des véhicules. Pour les objets ayant un petit volume, comme les piétons, le choix de la face ne fait pas grande différence.

Afin de déterminer l'erreur sur la localisation du centroïde par l'algorithme *YOLO* uniquement, il est considéré que les capteurs sont parfaits. Afin de ne pas prendre trop de temps, les données sont annotées rapidement, et peu d'images sont sélectionnées. Le but est simplement d'obtenir un ordre de grandeur de l'erreur, car il serait de toute manière trop long de calculer le déviation standard de la position sur un grand nombre d'images. De plus, l'annotation manuelle n'est pas parfaite, et introduit une erreur supplémentaire. Les ordres de grandeurs des variances de l'erreur, pour chaque dimension mesurée par un des deux capteurs, sont reprises dans le TABLEAU 5.2.

L'erreur intrinsèque en vitesse du radar est de  $1 \text{ km h}^{-1}$ , donc l'écart type est de  $0.27 \text{ ms}^{-1}$ . En effet, la courbe de l'erreur des mesures du radar donnerait une courbe en cloche, dont 95 % des valeurs seraient comprises à une distance inférieure ou égale à  $1 \text{ km h}^{-1}$  autour de la grandeur réel. L'intervalle à 95 % donnant deux fois l'écart-type, ce dernier vaut bien  $1 \text{ km h}^{-1}$ , en considérant que le constructeur donne la résolution du capteur sur base d'un intervalle similaire. La variance est donnée par l'écart type élevé au carré, et vaut donc  $0.0729 \text{ m}^2 \text{s}^{-2}$ .

### 5.3. Définition des matrices de covariance et des paramètres de seuil

Grandeur	Erreur intrinsèque	Erreur des boîtes	Variance totale $\sigma^2$
Angle azimuthal caméra	0	2.25	2.25
Angle d'élévation caméra	0	0.3	0.3
Distance caméra	0	25	25
Distance radar	1	9.3	10.3
Vitesse radar	0.0729	0.765	0.8378

TABLEAU 5.2 – Variances des erreurs sur la valeur des variables d'état mesurées du système.

#### 5.3.3 Erreur de prédiction du modèle

Étant donné que le pas de temps entre les itérations est petit, l'erreur associée aux prédictions doit être petite. En effet, les objets parcouruent très peu de distance, et change très peu de vitesse en 1 centième de seconde. Par exemple, une voiture accélérant à 1 g n'aura changé que de  $0.089 \text{ m s}^{-1}$  entre deux pas de temps. Dès lors, il a été décidé de considérer les valeurs de variances suivantes :

$$\mathbf{Q} = \begin{pmatrix} 0.02 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.002 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.002 \end{pmatrix}$$

#### 5.3.4 Valeurs des paramètres de seuils

En plus des matrices de covariance, il est nécessaire de déterminer les valeurs des 4 paramètres suivants :  $\lambda$ ,  $P_D$ ,  $I_{max}$  et  $\beta$ . Le paramètre  $\lambda$  est à réglé en fonction du nombre de faux positifs parmi les détections. Pour commencer, sa valeur est mise à 1.0, et il sera adapté en fonction des résultats. Étant donné que l'algorithme *YOLO* donne directement un indice de confiance associé à chaque détection,  $P_D$  est choisi comme étant égal à ce même indice de confiance. À la différence de [36], ce paramètre est donc dépendant des détections. Un objet devenant invisible pendant une seconde sera considéré comme perdu, et la piste associée sera supprimée. Dès lors,  $I_{max}$  est choisi comme valant 100. La distance minimale d'association des détections,  $\beta$ , est plus difficile à choisir. Il est donc décidé de trouver une valeur adéquate expérimentalement, en commençant par une valeur de 0.5.

#### 5.3.5 Estimation de la distance des objets

Afin d'estimer la distance des objets détectés par la caméra, il est tout d'abord nécessaire d'en estimer leur hauteur. Pour cela, il a été décidé de réaliser une rapide recherche des hauteurs

## Chapitre 5. Stratégie de fusion des données et de suivi des cibles

---

Alarme de faux positif	$\lambda$	1.0
Indice de confiance des détection	$P_D$	$p_{YOLO}$
Itérations sans détection	$I_{max}$	100
Distance d'association minimum	$\beta$	0.5

TABLEAU 5.3 – Valeurs initiales des paramètres du filtre de Kalman

typiques des différents objets, pour chaque classe reprise dans le TABLEAU 5.4. Lors de l'étape de détection, si deux objets de classe différente sont détectés au même endroit, la moyenne pondérée de la hauteur des deux objets est prise en compte. La hauteur estimée de chaque objet est pondérée par l'indice de confiance de la détection :

$$\text{hauteur estimée} = \frac{p_1 \cdot (\text{hauteur objet 1}) + p_2 \cdot (\text{hauteur objet 2})}{p_1 + p_2}$$

Il est vrai que la hauteur des objets n'est pas forcément liée à leur aspect, mais dans le cas présent la seule information disponible est celle donnée par l'algorithme *YOLO*. Si une mauvaise estimation de la distance des objets est observée pour une certaine classe d'objets, la hauteur moyenne de la classe peut être adaptée en conséquence.

Classe	Voiture	Personne	Camion	SUV	Van	Vélo	Moto	Bus	Trottinette	UCL	Remorque	Chien
h [m]	1.423	1.65	3.4	1.581	1.76	0.76	1.10	3.31	1.17	1.5	0.8	0.5

TABLEAU 5.4 – Hauteur moyenne des objets pour chaque classe

## 5.4 Résultats préliminaires

Avant tout, il est nécessaire de définir un critère de performance du suivi des pistes. L'idéal serait de connaître la trajectoire des véhicules, et de minimiser l'erreur en fonction de l'évolution des paramètres. Cependant, il serait alors nécessaire d'annoter un grand nombre de données. En plus de prendre beaucoup de temps, il pourrait être difficile d'annoter de manière précise les trajectoires des objets ne suivant pas la route, comme par exemple les piétons. Les paramètres seront donc optimisés de manière à ce que les véhicules suivent la route. La trajectoire des piétons sera vérifiée sur base des vidéos, en prenant comme point de repère les routes secondaires du carrefour.

De plus, il a été décidé de se focaliser sur une zone comprise entre 25 m et 55 m uniquement, comme représentée sur la FIGURE 5.1. En effet, la détection des objets trop proches du dispositif, ou trop loin au bord du champ de détection des capteurs, introduit des distorsions. Les objets sont en effet détectés partiellement par la caméra, ce qui déplace leur centroïde. Dès lors, tenter de corriger ces phénomènes avec les paramètres du filtre de Kalman pourrait être contre productif. De plus, seule la zone où des interactions entre les véhicules et les piétons ont lieu est étudiée dans le cadre de ce projet.

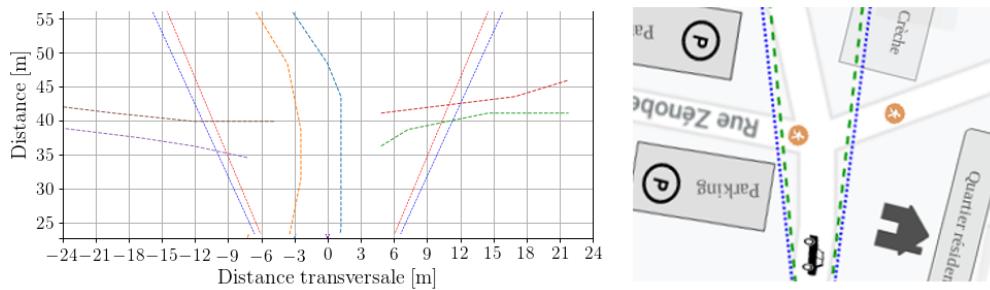


FIGURE 5.1 – Zone d’intérêt du carrefour étudié. Sur le graphique, les angles de vue du radar et de la caméra, ainsi que la disposition des routes sont représentés. Les lignes associées aux routes ne sont pas le bord de la chaussée, mais le milieu de chaque voie. Ainsi, les véhicules devraient suivre ces lignes.

Les objets détectés sont représentés par des cercles, dont le rayon correspond au déterminant de la matrice de covariance de la piste correspondante. De plus, la classe associée à la piste donne la couleur du marqueur.

Des objets ayant des rayons importants feront dès lors penser que le modèle est imprécis, et que les valeurs de seuil ou de variance doivent être changées. Au contraire, si des objets de petite taille apparaissent en dehors de la route, il sera nécessaire de revoir l’influence des détections sur le modèle.

### 5.4.1 Validation des mesures des deux capteurs

Dans un premier temps, chaque capteur est testé séparément, afin de vérifier la pertinence des mesures. Cela permet de ne pas être impacté par un délai éventuel entre les capteurs, ou un problème de fusion des données.

La caméra permet une visualisation intéressante des données, car les angles sont mesurés, et la distance est estimée. Dès lors, il est facile de s’assurer de la qualité des mesures, en vérifiant que les véhicules suivent bien la route. Malgré une hypothèse très forte faite sur la hauteur des objets (cf. Section 5.3.5), la distance et la vitesse des objets peuvent être correctes, avec cependant une incertitude importante sur la validité des pistes. En effet, certains objets dont les dimensions diffèrent trop de la moyenne, sont détectés plus proches ou plus loin par rapport à leur distance réelle. De plus, les objets détectés correctement, mais dont la classification est erronée, peuvent donner lieu à d’importantes erreurs lors de l’estimation de la distance.

Le radar mesure quand à lui seulement la distance et la vitesse radiale des objets. Dès lors, les détections se situent sur l’axe  $x = 0$  dans le repère du dispositif. Les véhicules circulant sur la route donnent donc lieu à des points se déplaçant sur l’axe des ordonnées. La vitesse étant mesurée, la prédiction du modèle fonctionne bien, tant que l’accélération n’est pas trop importante. Le peu d’images par secondes du radar ne semble pas être un problème, car les mesures sont cohérentes avec les prédictions du modèle.

### 5.4.2 Suivi d'une cible unique

Dans un second temps, les résultats du suivi de cible avec l'algorithme de Kalman sont vérifiés, avec le cas simplifié où il n'y a qu'une seule cible à suivre. Dès lors, toutes les détections sont associées à l'unique piste existante. Cela revient en fait à imposer que la distance maximale d'association soit infinie ( $\beta = \infty$ ).

Grâce à la bonne précision de la détection avec l'algorithme *YOLO*, il y a peu de faux positifs, autant pour la caméra que pour le radar. De plus, la précision intrinsèque des capteurs est relativement bonne. Par conséquent, les objets devraient être suivis sans trop de difficulté, à condition que les paramètres du filtre de Kalman soit correctes.

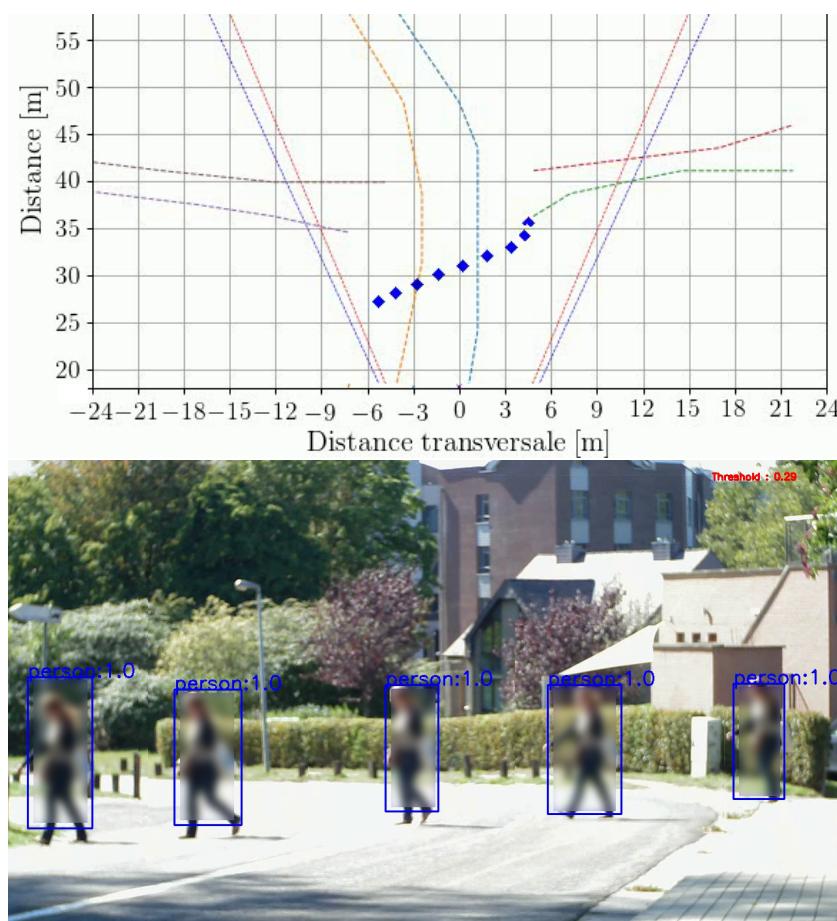


FIGURE 5.2 – Suivie de la trajectoire d'un piéton venant de la route secondaire droite. Le piéton ne se dirige pas vers la route secondaire gauche, mais vers un sentier plus proche du dispositif. Malgré une certaine imprécision sur la dimension des boîtes entourant le piéton, la trajectoire est correcte.

La FIGURE 5.2 montre le suivi de cible d'un piéton traversant la route. La trajectoire semble bien calculée. De plus, l'incertitude est petite, comme le montre la taille du marqueur. La

caméra prenant relativement beaucoup d'images par seconde, l'incertitude reste limitée, même si aucune détection n'est faite pour plusieurs pas de temps consécutifs.

Les véhicules sont bien détectés et suivis, mais l'utilisation des données radar semble être nécessaire. Lorsque la fusion est bonne, la distance détectée par le radar corrige bien l'estimation faite sur base des informations de la caméra. La FIGURE 5.3 montre le suivi d'une voiture, avec et sans l'utilisation des données radar. Lorsque les détections radar sont désactivées, la distance de la voiture est très imprécise. De plus, la vitesse radiale est mal estimée, ce qui donne lieu à des pistes distordues.

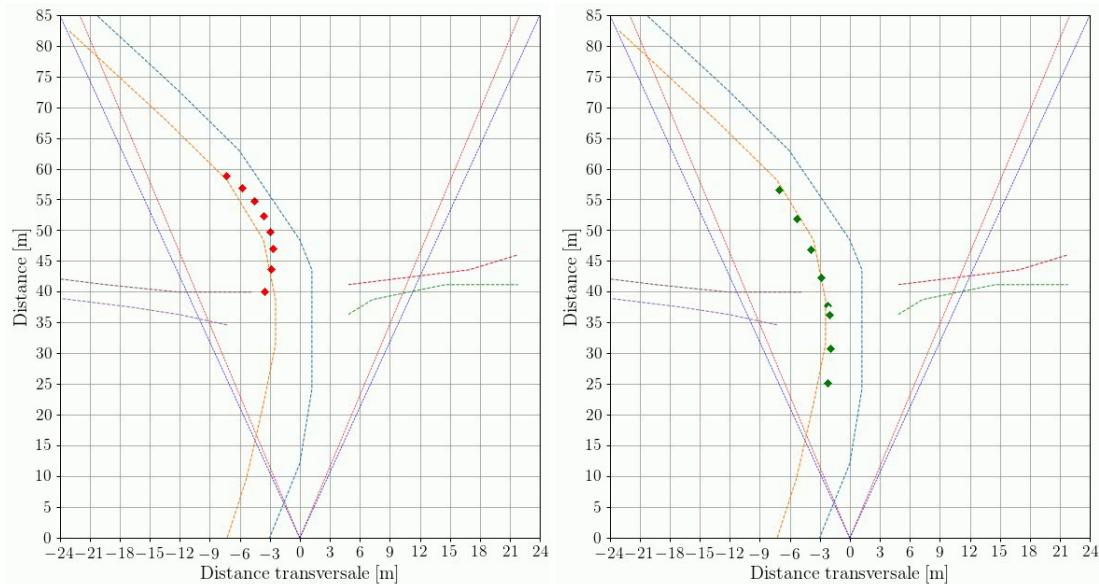


FIGURE 5.3 – Suivie d'une voiture avec et sans l'utilisation des données radar. À gauche, les données radar ne sont pas utilisées, et la distance est estimée sur base des données de la caméra uniquement. À droite, les données de la caméra et du radar sont fusionnées.

Malheureusement, le suivi d'une seule cible à la fois n'est bien souvent pas suffisant. De plus, le but de cette étude est d'étudier l'interaction entre plusieurs usagers de la route. Il devient dès lors nécessaire de mettre en place une stratégie d'association des données permettant de différencier les différentes pistes.

### 5.4.3 Association des pistes lors du suivi de cibles multiples

Afin de suivre plusieurs objets simultanément, il est tout d'abord nécessaire de modifier le paramètre de distance probabiliste maximale d'association. Une valeur trop grande de ce paramètre mène à de mauvaises associations, surtout dans le cas où plusieurs objets se trouvent à la même distance. Au contraire, si la distance limite est trop petite, les détections de la caméra ne sont pas associées à celles du radar, et la fusion des données n'a pas lieu.

Il convient de rappeler que l'association d'une détection de la caméra, avec une détection du

## **Chapitre 5. Stratégie de fusion des données et de suivi des cibles**

---

radar, se fait sur base de la distance uniquement. Du fait de l'erreur importante sur la distance estimée par la caméra, la fusion n'est pas triviale. En effet, les pistes peuvent se diviser, ou au contraire des objets singuliers peuvent fusionner.

Sur base des résultats obtenus, il a été observé que le suivi de cibles multiples sur base de l'information de distance uniquement, ne permet pas d'étudier correctement l'interaction entre les usagers de la route. Il est donc nécessaire de mettre en place une stratégie d'association plus complète.

### **5.4.4 Influence des paramètres sur les résultats**

Les résultats du suivi d'une cible unique montre que les données du radar et de la caméra peuvent être fusionnés, afin d'obtenir des meilleurs résultats qu'avec un seul capteur. Il semble donc que les paramètres du modèle, ainsi que les matrices de covariance des capteurs, soit correctement choisies. Le vecteur  $\vec{q}$  a cependant été mis à une valeur non nulle, en ajoutant un décalage de  $1^\circ$  pour toutes les mesures. En effet, la topologie de la route a été mesurée sur base d'images satellites, et il semble que l'angle du dispositif par rapport à la route diverge de  $1^\circ$  par rapport à la représentation qui en est faite dans les graphiques.

La valeur du paramètre d'alarme de faux positifs,  $\lambda$ , ne semble pas avoir beaucoup d'influence pour le moment. Il est possible que la bonne précision de l'algorithme *YOLO*, ainsi que l'utilisation de l'indice de confiance pour pondérer la probabilité de fausse détection, rendent moins critique le paramètre  $\lambda$ .

Le paramètre du nombre d'itération maximum sans détection doit être raisonnable. Si le paramètre est trop petit, les pistes sont discontinues. Si au contraire il est trop grand, des pistes "fantômes" apparaissent, pouvant même s'associer avec des détections d'autres objets.

L'étape critique s'est révélée être l'association des détections des deux capteurs, même si ce n'est pas vraiment une surprise, car les mesures sont totalement hétérogènes. Pour surmonter ce défis, la simple utilisation d'une distance probabiliste ne semble pas suffire. Il est dès lors nécessaire d'utiliser l'information de manière plus poussée.

# 6 Améliorations de la précision du filtre probabiliste d'association

Afin d'améliorer le processus de fusion et de filtrage des faux positifs, plusieurs améliorations ont été proposées. Tout d'abord, chaque modification de l'algorithme est présentée séparément. L'utilité des améliorations est finalement discutée.

## 6.1 Rectification du déséquilibre des composantes en quadrature du radar

Comme mentionné dans la Section 2.1, l'électronique imparfaite du radar mène à l'apparition de détections fantômes, symétriques aux vrais objets dans le graphique Doppler. Afin de filtrer ces faux positifs, il a été décidé de compter comme fausses détections les objets se trouvant dans une zone interdite, comme le montre la FIGURE 6.1. Dès lors, si le centroïde d'une détection est dans cette zone, la détection est comptée comme faussement positive, pour autant que son indice de confiance soit inférieure à la détection se trouvant en symétrie.

La zone critique s'étend sur un intervalle d'environ  $2 \text{ m}$  et  $3 \text{ km h}^{-1}$ . Il a cependant été observé qu'il est rare que deux détections symétriques apparaissent. L'algorithme *YOLO* semble déjà filtrer en partie le phénomène de déséquilibre des composantes du radar. En effet, les fausses détections symétriques apparaissent très souvent lors d'un faux négatif de l'objet réel.

Une zone est donc retirée du champ de détection du radar, et il serait possible qu'un objet réel y passe. C'est pourquoi il est nécessaire que cette zone soit la plus petite possible. L'intervalle choisi semble suffisant pour filtrer les fausses détections, et il est peu probable qu'un objet se cache dans la zone interdite pour une longue période de temps. Il faudrait en effet que deux objets se déplacent selon des directions opposées, avec une vitesse radiale qui diffère d'au plus  $3 \text{ km h}^{-1}$ . La distance des objets, par rapport à la moitié de la distance maximale détectable, devrait de plus rester quasiment identique.

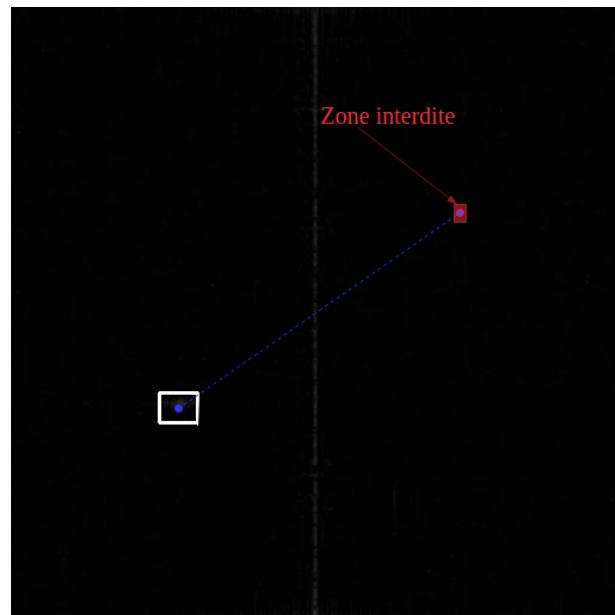


FIGURE 6.1 – Représentation de la zone interdite. Tous les centroïdes dans la zone rouge (à droite) sont comptés comme faux positifs.

## 6.2 Prise en compte de la classe lors de l'étape d'association

Afin d'éviter que des objets trop différents ne s'associent de manière abusive, il a été décidé de donner un malus de distance probabiliste d'association, si la classe de la piste et de la détection ne sont pas les mêmes. En effet, un piéton ne sera par exemple presque jamais détecté comme étant un camion, et cela n'a donc pas vraiment de sens d'associer des détections de camions, à des pistes de piétons. Dès lors, quatre niveaux de malus ont été choisi : pas de malus, un malus faible, un malus moyen et un malus haut.

Le malus d'association d'une détection  $j$  à une piste  $i$ , doit être inversement proportionnel à l'espérance d'obtenir une détection de la classe de  $j$ , sachant que l'objet est en réalité de la même classe que  $i$ .

Dans le cas où une voiture et un piéton se trouvent sur la route, imposer un gros malus inter-classes permet en fait presque de se ramener au cas du suivi d'une cible unique. Cette fois-ci, toutes les détections de piétons par la caméra, et de petits objets par le radar, sont associées à la personne. Les autres détections sont associées à la voiture. Il devient dès lors possible d'étudier l'interaction des objets.

Lorsque plusieurs objets de la même classe sont présents, la distance probabiliste minimale d'association ne doit pas être trop grande. Il est dès lors inutile de choisir des malus trop importants.

### 6.3 Détection des situations d'obstruction

Lorsque plusieurs objets se suivent, comme par exemple des voitures sur une bande de la route principale, il arrive régulièrement qu'un objet en cache d'autres, comme le montre la FIGURE 6.3. Dans ce cas, et si les objets sont détectés avant qu'ils ne soient cachés, il est possible d'identifier une situation d'obstruction. En effet, la zone cachée par un objet peut être calculée, à condition de connaître la distance et la largeur de l'objet (voir FIGURE 6.2).

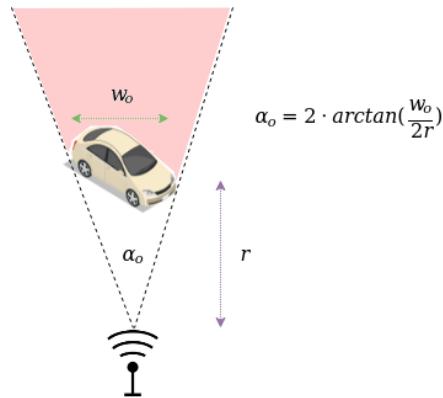


FIGURE 6.2 – Calcul de la zone cachée par un objet

Il est dès lors nécessaire de donner une approximation de la largeur des objets, et l'angle d'obstruction  $\alpha_o$  peut être calculé sur base de la distance. Les objets se trouvant dans la zone cachée sont dès lors considérés comme obstrués, et les pistes bénéficient d'un bonus de survie en cas de non détection. Autrement dit, le paramètre  $I_{max}$  est augmenté pour les pistes cachées. Une piste pourra ainsi survivre quelques temps si un objet passe devant. Il a été choisi de laisser 2 secondes de temps supplémentaire aux objets obstrués.

Afin d'éviter des fausses détections de situation d'obstruction, les pistes impliquées devront tout de même avoir un certain âge, et un nombre minimum de détections associées. En effet, des cas d'obstruction apparaissant lors des cas de fausses détections peuvent mener à l'instabilité du système.

### 6.4 Association d'une unique détection par piste

Lors de l'étape d'association du filtre de Kalman, une détection ne peut être associée qu'à une seule piste, celle dont elle est la plus proche. En revanche, une piste peut se voir assigner plusieurs détections. Étant donné qu'une étape de filtrage des détections multiples a déjà été implémentée, via l'ajustement des paramètres de l'algorithme *YOLO*, chaque détection devrait pourtant appartenir à un objet différent.

Afin d'améliorer le rappel des pistes, il a donc été décidé d'imposer que chaque détection d'une même image soit associée à une piste différente. L'objectif principal est la dissociation des piétons appartenant à un même groupe.

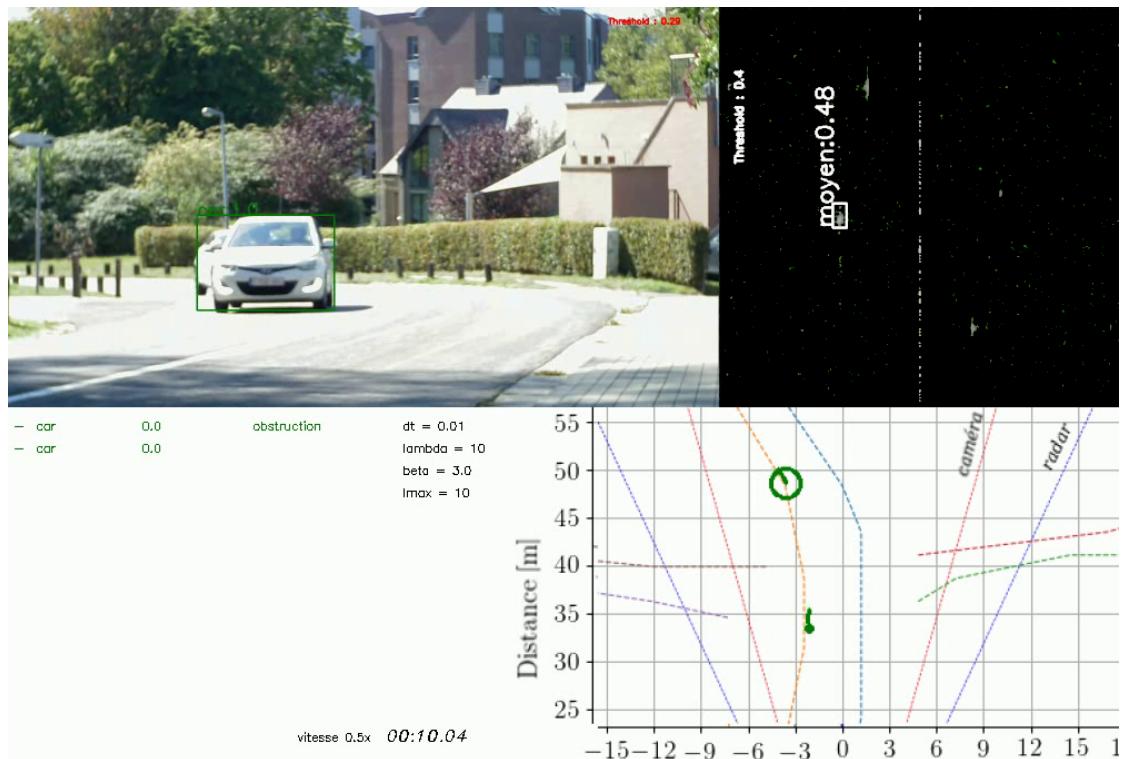


FIGURE 6.3 – Cas d'une obstruction détectée lors du passage de deux voitures. L'interface présente l'image vidéo en haute à gauche, ainsi que les boîtes détectées par l'algorithme YOLO. En haut à droite, les détections radar sont visibles. La partie de l'image en bas à droite montre la zone d'intérêt, débutant à 25 m du dispositif. Finalement, les informations concernant l'algorithme de Kalman sont reprises en bas à gauche, et les cas d'obstruction y sont indiqués.

## 6.5 Imposition d'une distance minimum pour la création d'une piste

Lorsqu'une détection est un peu trop éloignée de la piste à laquelle elle devrait appartenir, l'association ne se fait pas, et une nouvelle piste est créée. Il vient donc l'idée d'imposer une distance minimum entre les pistes et une détection, pour approuver la création d'une nouvelle piste. Il convient cependant d'être attentif au fait que la distance minimale à imposer dépend du type d'objet. En effet, alors que les centroïdes de plusieurs camions ne pourront pas se trouver à moins de quelques mètres les uns de autres, il est tout à fait probable que ce soit le cas pour des piétons.

Une distance minimale pour la création d'une nouvelle piste a donc été déterminée, pour toutes les classes d'objets. Pour chaque nouvelle détection, il y a donc maintenant trois possibilités. Soit la détection est assez proche d'une piste, et une association a lieu. Soit la détection est assez loin de toutes les pistes, et une nouvelle piste est créée. Si aucune de ces conditions n'est remplie, la détection est simplement ignorée. L'idée est ici d'implémenter un filtrage des données aberrantes.

## 6.6 Filtrage des pistes à l'affichage

Il est possible de filtrer les pistes à l'affichage, c'est-à-dire de ne pas afficher toutes les pistes existantes. Il s'agit alors d'accorder une certaine confiance aux pistes, en fonction du nombre de détections associées par exemple. Il a donc été décidé de n'afficher que les pistes ayant plus de 5 détections associées, ou au moins une détection de chaque capteur. Ainsi, les fausses détections restantes sont quasiment toutes filtrées. Cela introduit cependant un délai avant l'affichage d'une nouvelle piste, de l'ordre de 0.1 seconde, si chaque image de la caméra détecte l'objet.

Il a été envisagé de n'afficher que les pistes avec au moins une détection de chaque capteur associée, mais certains piétons trop loin du dispositif ne sont alors jamais affichés. En imposant un nombre minimum de détections, les piétons non détectés par le radar sont alors affichés, mais avec une erreur en distance radiale importante. En effet, la position de piétons est détectée avec une incertitude de quelques mètres, si aucune détection radar ne s'est produite.

Il est aussi possible de ne pas afficher les pistes ayant une incertitude trop grande. L'incertitude est évaluée sur base du déterminant de la matrice de covariance, qui est lié à l'entropie différentielle du vecteur position de la piste. Autrement dit, le déterminant de la matrice  $\mathbf{P}$  est proportionnel à l'erreur minimale probable sur la position du vecteur  $\hat{x}$ . Si l'erreur potentielle est trop grande, la piste peut ne pas être affichée.

Étant donné que l'incertitude d'une piste augmente à chaque itération si aucune détection ne lui est associée, il serait en fait possible de ne pas imposer de limite d'itération maximum sans détection, et de mettre uniquement une limite d'incertitude. Il a cependant été décidé de garder deux paramètres distincts, car le nombre maximum d'itérations sans détection est beaucoup plus simple à optimiser qu'une limite d'incertitude.

## 6.7 Discussion des résultats

Au travers de l'observation des résultats avec et sans l'activation des mécanismes proposés dans ce chapitre, il a été observé une amélioration du rappel et de la précision. En particulier, l'association par classe d'objets (décrise dans la Section 6.2), s'est avérée être un principe simple mais efficace. De même, le filtrage de l'affichage permet de supprimer la majorité des faux positifs parmi les pistes. Ces deux modifications de l'algorithme représentent donc une amélioration globale du filtre de Kalman.

L'imposition d'une distance minimum pour la création d'une piste, du filtrage des détections symétriques du radar, ainsi que l'association d'une seule détection par piste, semblent bénéfiques pour la précision. Cependant, l'apport de précision semble se concentrer sur quelques cas particuliers, sans impacter la totalité des résultats. L'apport des ces améliorations est donc marginal, sans pour autant être nul.

## **Chapitre 6. Améliorations de la précision du filtre probabiliste d'association**

---

Finalement, le système de détection des obstructions donne des résultats intéressants, mais est aussi source d'erreur. En effet, malgré une amélioration significative du suivi des cibles lorsqu'une obstruction a réellement lieu, des cas de fausses détections ont été observés. Par exemple, un vélo est détecté au même endroit qu'un piéton, et il est considéré que le piéton cache le vélo. Dès lors, une piste peut se créer et survivre pendant quelques secondes. Il serait donc éventuellement nécessaire de revoir les règles menant à la détection d'une situation d'obstruction.

À ce stade, la fiabilité des pistes, ainsi que l'erreur en position des objets, ne permettent pas de calculer précisément la distance entre les objets. La présence d'un véhicule et d'un piéton peut être détectée, mais la distance minimale entre les deux pistes donne seulement une approximation de la distance réelle. De plus, l'hypothèse de hauteur des objets amène une imprécision importante dans le calcul des distances, lorsqu'aucune détection radar n'est associée.

# 7 Analyse statistique du lieu d'étude

Après l'optimisation de la détection des objets avec le radar et la caméra, l'implémentation d'une stratégie de fusion des données et de suivi des cibles, une rapide analyse du trafic au niveau du lieu d'étude peut être faite. La capacité de l'algorithme à recenser le nombre et le type d'objets circulant est tout d'abord testée. Une méthode de détection des situations impliquant des piétons et des véhicules est ensuite confrontée à la réalité.

## 7.1 Comptage des véhicules et des piétons

Il est tout d'abord proposé dévaluer le nombre et le type d'objets passant devant le dispositif. L'étude se porte sur 7 séquences d'enregistrement, durant lesquels différents types de véhicules passent. Les résultats obtenus avec l'algorithme de suivi des cibles peuvent être comparés avec le nombre réel d'usagers de la route, sur base des informations reprises dans le TABLEAU 7.1. Une détection est validée pour toute piste existant plus de 2 secondes. L'hypothèse est donc faite qu'aucun objet ne peut passer devant le dispositif en moins de 2 secondes, ce qui permet de filtrer un nombre important de faux positifs.

Classe	<i>Voiture</i>	<i>SUV</i>	<i>Personne</i>	<i>Camion</i>	<i>Camionnette</i>	<i>Vélo</i>	<i>Véhicule UCL</i>
Detections	39	5	14	3	5	1	1
Réalité	41	2	16	3	6	0	1

TABLEAU 7.1 – Nombre d'objets détectés, et nombre d'objets passant réellement devant le dispositif.

Le nombre de véhicules détectés est proche de la réalité, malgré un écart de quelques individus pour chaque classe. Il est vraisemblable que des erreurs de classification entre les *SUV* et les *sedan* puissent avoir lieu. En effet, l'algorithme *YOLO* a souvent du mal à faire la différence entre ces deux classes, et un trop grand nombre de *SUV* est souvent détecté. Les piétons sont détectés dans la plupart des cas, malgré 2 faux négatifs.

## 7.2 Détection des situations impliquant des véhicules et des piétons

La distance calculée n'étant précise qu'à quelques mètres, la dangerosité des situations impliquant des véhicules et des piétons est difficile à évaluer. Il sera donc considéré que la présence de piétons et de véhicules circulant au même moment aux alentours du carrefour est une situation potentiellement dangereuse. La distance entre les pistes est calculée sur base de deux dimensions uniquement, car prendre en compte la distance verticale des centroïdes n'a pas vraiment de sens. La distance  $d$  entre deux pistes est dès lors donnée par :

$$d_k = \sqrt{(X_{1,k} - X_{2,k})^2 + (Y_{1,k} - Y_{2,k})^2}$$

où  $(X_{a,k}, Y_{a,k})$  sont les coordonnées de l'objet  $a$  dans le plan horizontal du dispositif de capture, et  $k$  se réfère à l'instant auquel la distance est calculée.

L'évaluation des distances est faite sur les 7 séquences d'enregistrement utilisés pour l'étude statistique de fréquentation du lieu d'étude. Une interaction entre un véhicule (voiture, camion, SUV, camionnette, bus, véhicule UCL ou remorque), et un usager vulnérable traversant la route (personne, vélo), est considérée potentiellement dangereuse si la distance minimum entre les deux pistes est inférieure à 3 m. Un indice de confiance est construit sur base du nombre minimum de détections associées aux pistes. Si une des deux pistes impliquées dans l'interaction a moins de 3 détections associées, un cas de faux positif est considéré.

Les résultats de l'analyse sont repris dans le TABLEAU A.1, en ANNEXE A.3. La présence de piétons proches des véhicules est bien détectée, car seul 1 cas de faux négatif a été observé. En revanche, la distance minimale entre les pistes comporte une erreur importante, ce qui rend la détection d'un risque de collision incertaine. Plus de la moitié des faux positifs pourraient être filtrés en imposant un niveau de certitude supérieur à 3, mais alors le nombre de faux négatifs augmenterait. Dans ce cas, la précision serait en effet de 44 % environ, et le rappel de 64 %.

Les distances calculées entre les véhicules et les piétons varient entre 3 m et moins de 1 cm. Il est dès lors évident que certaines valeurs doivent être considérées comme comportant une erreur importante, car il n'est simplement pas possible que le centroïde d'un véhicule soit si proche d'un piéton.

## 7.3 Discussion des résultats

L'algorithme implanté a permis de faire une étude approximative du nombre de véhicules et de piétons fréquentant le lieu d'étude. Malgré quelques erreurs de classification, ainsi que des objets non détectés, l'ordre de grandeur des résultats correspond à la réalité.

La détection des situations d'interaction entre les usagers de la route ne permet pas une étude de risque du lieu d'étude. Il serait de plus nécessaire d'analyser la vitesse et la direction des

### **7.3. Discussion des résultats**

---

objets, afin de savoir si un risque de collision existe réellement.

Compte tenu du nombre d'usagers différents circulant non loin des piétons, un danger potentiel est cependant considéré. Le risque d'accident ne peut cependant pas être évalué de manière objective.



## 8 Améliorations proposées

Malgré de bon résultats de détection et de classification des objets, une marge de progression existe, permettant de diminuer les erreurs se propageant à l'étape de fusion et de suivi des cibles. L'algorithme de suivi des cibles donne des résultats encourageants, mais ne permet pas d'étudier précisément les interactions entre les différents usagers de la route. Ce chapitre tente dès lors d'apporter les éléments techniques pouvant mener à cet objectif. Dans un premier temps, les améliorations sont présentées comme si le projet était à refaire, tout en suivant la stratégie déjà implémentée. Pour finir, une approche alternative de la problématique est imaginée, et une courte discussion sur les enjeux de l'objectif scientifique suivi est faite.

### 8.1 Synchronisation physique des deux capteurs

Un délai temporel persiste entre le radar et la caméra. En effet, il a été observé que la latence des deux capteurs variait légèrement entre les différentes mesures. Dès lors, le délai est aussi variable. Même si celui-ci est faible au final, il faut garder à l'esprit qu'à  $50 \text{ km h}^{-1}$ , une erreur temporelle de 0.1 s implique une erreur en distance d'environ 1.4 m. Cette erreur est plus importante que la déviation standard de l'erreur en distance radiale mesurée par le radar, qui est de 1 m. Tout délai subsistant entre le radar et la caméra est donc une source d'erreurs importantes.

Afin d'obtenir une synchronisation quasiment parfaite des deux capteurs, il est nécessaire d'utiliser le même signal d'horloge pour gérer l'instant de capture des données. Pour la caméra, les composants sont accessibles facilement, et il suffit en théorie de trouver le signal fourni par l'oscillateur, afin de le remplacer par un signal d'horloge de référence. En ce qui concerne la radar, l'opération risque d'être un peu plus complexe, car les composants ne sont pas visibles, et le fonctionnement du radar est géré par un module de technologie FPGA (*Field-Programmable Gate Array*). Une stratégie serait alors de modifier le programme installé sur le FPGA, afin de changer la provenance du signal d'horloge gérant la capture des données. Il serait aussi possible de rendre accessible le signal d'horloge du radar via un fil, et de synchroniser la caméra à partir de celui-ci.

## **Chapitre 8. Améliorations proposées**

---

Les opérations à mener sur le radar sont cependant risquées, car elles pourraient mener au dysfonctionnement du capteur, et peut être même à sa mise hors service permanente. Dès lors, il serait peut-être préférable de contacter le fabricant (RFbeam), afin d'obtenir plus d'informations techniques sur l'imposition de la fréquence de capture du radar.

### **8.2 Amélioration de la détection et de la classification**

Afin d'améliorer la précision et le rappel lors de la détection des objets avec l'algorithme *YOLO*, il serait tout d'abord possible d'annoter plus de données. Cependant, un investissement de temps important serait alors nécessaire. Il pourrait aussi être fait usage de la version complète de l'algorithme *YOLO*, car uniquement la version rapide *tiny-YOLO* est utilisée dans le cadre de ce projet. Le temps de calcul serait alors augmenté d'un facteur pouvant atteindre 10.

Une stratégie moins extrême serait de réaliser une partie seulement des détections avec la version complète de l'algorithme. Par exemple, une image sur dix de la caméra peut être analysée de manière plus poussée. En accordant davantage de confiance aux détections faites par le réseau plus profond, il serait ainsi possible de filtrer plus facilement les faux positifs faits par l'algorithme réduit, et même de détecter certains objets jusqu'alors invisibles. Plusieurs stratégies de fusion de l'information sont alors possibles. Les données peuvent être considérées comme provenant du même capteur, et les indices de confiance des détections sont adaptés en fonction de la version de l'algorithme *YOLO* utilisée. Une seconde option est d'aller jusqu'à considérer que deux caméras différentes prennent des données, et d'adapter les matrices de covariances dans l'algorithme d'association probabiliste des données.

Une méthode basée sur l'utilisation partielle de la version complète de *YOLO* permettrait potentiellement de meilleurs résultats que ceux obtenus par l'utilisation de fenêtres glissantes, mais nécessiterait l'entraînement d'un réseau supplémentaire.

### **8.3 Fusion des données et suivi des cibles multiples**

Compte tenu des résultats obtenus avec l'algorithme de fusion des données et de suivi des cibles, certaines améliorations plus profondes seraient nécessaires.

#### **8.3.1 Utilisation des angles mesurés par le radar**

Tout d'abord, l'utilisation des angles des cibles mesurés par le radar aiderait presque certainement lors de l'étape d'association des données. Même si une ambiguïté sur les angles existe (car un déphasage supérieur à  $180^\circ$  ne peut pas être perçu correctement), il est fort probable que l'information comporte moins d'erreur que l'estimation de distance faite sur base des images de la caméra. Concernant la fusion des données, la priorité serait donc de compléter le vecteur des mesures du radar, par les valeurs des angles azimutaux et verticaux.

### 8.3.2 Optimisation par apprentissage des paramètres du filtre de Kalman

Une limitation importante de l'algorithme de suivi des cibles implémenté, est le nombre élevé de paramètres à régler. Pour commencer, il est nécessaire de trouver les valeurs des matrices de covariances, pour chacun des capteurs ainsi que pour le modèle. Ces paramètres influent sur l'incertitude des pistes, et la distance minimum d'association doit être adaptée en conséquence. Ce type d'interdépendance rend difficile, voir même impossible, l'optimisation des paramètres du filtre de Kalman par une méthode manuelle d'essai-erreur. Ce constat amène à adopter une stratégie d'optimisation des paramètres par apprentissage supervisé. Il serait dès lors nécessaire d'annoter des données de référence, ou de prendre des mesures GPS précises comme trajectoires de référence. La fonction de pertes pourrait simplement être construite sur base d'une méthode des moindres carrés. En s'inspirant de la fonction de pertes de l'algorithme *YOLO*, la valeur de perte  $L$  peut être définie comme étant :

$$\begin{aligned} L = & \lambda_p \sum_{i=1}^n \left[ (\hat{\alpha}_i - \alpha_i)^2 + (\hat{\epsilon}_i - \epsilon_i)^2 + (\hat{r}_i - r_i)^2 \right] \\ & + \lambda_v \sum_{i=1}^n \left[ (\hat{v}_{\alpha,i} - v_{\alpha,i})^2 + (\hat{v}_{\epsilon,i} - v_{\epsilon,i})^2 + (\hat{v}_{r,i} - v_{r,i})^2 \right] \\ & + \sum_{j \in \text{classes}} (p_j(c) - p_j(c))^2 \end{aligned}$$

où  $n$  points de la piste ont été annotés, et  $\lambda_p$  et  $\lambda_v$  permettent d'adapter l'influence de l'erreur en position et en vitesse, respectivement. Le  $p_j(c)$  terme est défini comme étant :

$$p_j(c) = P(\text{classe}_j | \text{piste})$$

et évalue la probabilité que la piste corresponde à une objet de classe  $j$ .

La fonction de pertes pourrait encore être développée, pour pénaliser les faux positifs parmi les pistes par exemple. Il serait dès lors nécessaire de définir un critère permettant de choisir quelles détections sont fausses. L'algorithme *YOLO* utilise comme critère l'*IoU* des aires des objets annotés et des détections. Un critère basé sur le coût de la transformation d'une piste afin d'en obtenir une autre serait ici plus adapté. Des critères de comparaison des trajectoires peuvent être trouvés dans [37].

En calculant le gradient des pertes en fonction de l'évolution des paramètres, un minimum local d'erreur peut être atteint après suffisamment d'itérations, et à condition que l'algorithme d'optimisation ne diverge pas.

### 8.3.3 Faire de la classe des objets une variable d'état des pistes

Malgré une bonne précision de l'algorithme *tiny-YOLO*, un certain nombre de faux positifs subsistent, ainsi que des erreurs de classification. Dans son état actuel, le filtre de Kalman supprime une partie des faux positifs, en accordant peu de confiance aux pistes ayant peu de détections associées. La classe de référence de la piste est calculée sur base de la classe de toutes les détections qui lui ont été associées. De plus, un système de malus a été implémenté, afin de ne pas associer des objets trop différents. Ce système nécessite beaucoup de paramètres, et complique l'optimisation du filtre de Kalman. Une alternative serait d'inclure la classe de l'objet suivi directement dans le vecteur d'état. Une nouvelle dimension pourrait être imaginée, dans laquelle chaque classe d'objets est placée. Les objets similaires sont placés proches les uns des autres, sur base de l'espérance des erreurs de classification. La distance entre une piste et une détection dépend dès lors directement des classes d'objets, et seul le paramètre de distance minimum d'association doit être optimisé.

## 8.4 Proposition d'une stratégie alternative

La FIGURE 8.1 met en lumière l'évolution de l'information au travers de l'algorithme développé dans le cadre de ce travail de fin d'études. Après l'étape de détection et de classification des objets, une grosse partie de l'information a déjà été perdue. Or, cette information pourrait être utile lors de la fusion des données et du suivi des cibles. De la même manière que les détecteurs en plusieurs étapes décrits dans la Chapitre 3.2.1, chaque étage doit être optimisé séparément, sur base de critères d'efficacité intermédiaires.

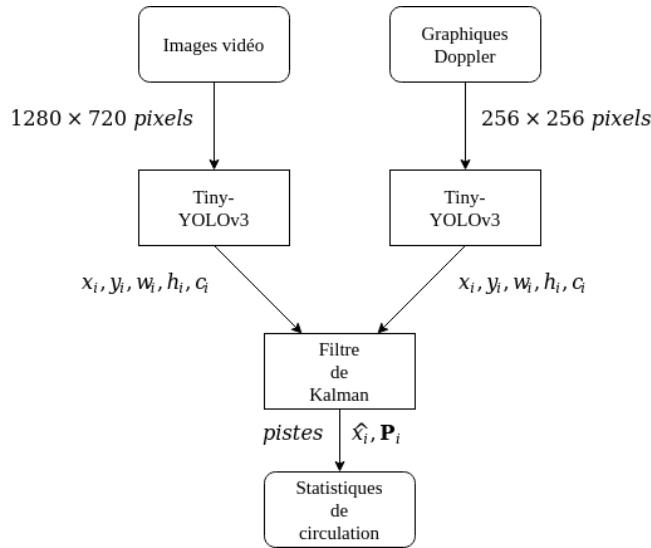


FIGURE 8.1 – Évolution de l'information entre chaque étape de l'algorithme implémenté

Dans l'esprit de l'algorithme *YOLO*, une stratégie n'utilisant qu'un seul réseau de neurones pourrait être adoptée. Une piste d'architecture d'un tel réseau de neurones est représenté sur

## **8.5. Discussion de l'impact potentiel de la technologie visée**

la FIGURE 8.2. De cette manière, plus de paramètres pourraient être optimisés, directement sur base d'une fonction de pertes s'appliquant au suivi des objets. Un exemple d'une solution allant dans ce sens a été implémenté dans [38].

Il serait tout d'abord nécessaire de concevoir des couches de neurones capables de fusionner des données, hétérogènes ou non. L'algorithme de Kalman pourrait en être une base de développement. Le réseau devrait de plus être capable de mémoriser des informations, afin de faire le lien entre les pistes et les mesures provenant des capteurs. Le temps  $\tau$  entre les captures devrait de plus être renseigné au réseau, et les pistes existantes injectées en amont du réseau.

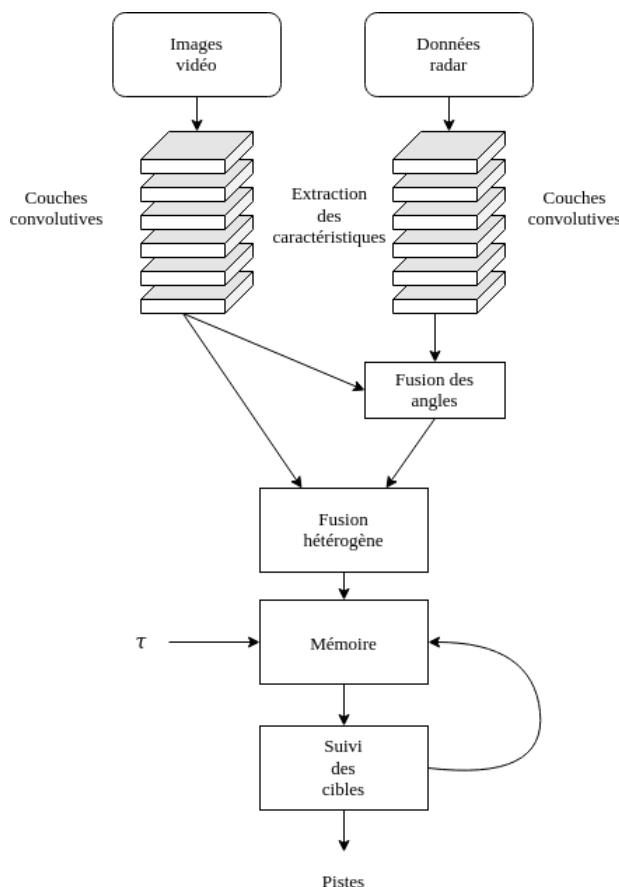


FIGURE 8.2 – Exemple de l'architecture d'un réseau de neurones, partant de l'information des capteurs au suivi des pistes

## **8.5 Discussion de l'impact potentiel de la technologie visée**

La possibilité d'étudier la dangerosité d'un lieu de manière objective, pourrait avoir un impact positif sur la sécurité des usagers de la route. En effet, un carrefour pourrait être qualifié de dangereux, avant même qu'un accident n'arrive. La mise en place d'éléments de prévention

## **Chapitre 8. Améliorations proposées**

---

des accidents serait alors possible. L'efficacité des dispositifs de prévention pourrait de plus être vérifiée avec l'utilisation combinée du dispositif d'analyse du trafic, ce qui permettrait un choix optimal. Si la technologie le permet, un algorithme fonctionnant en temps réel pourrait même permettre d'éviter un accident lorsqu'une situation à risque se présente. Il est même possible d'imaginer un dispositif communiquant avec des véhicules intelligents, afin d'apporter une source d'information supplémentaire aux capteurs installés sur les véhicules modernes.

Dans le cadre de ce travail de fin d'études, l'utilisation de la technologie visée pour verbaliser les comportements à risque des usagers de la route n'a pas été un objectif. Cependant, il est par exemple techniquement possible d'utiliser un dispositif similaire afin de pénaliser les véhicules ne s'arrêtant pas aux passages pour piétons. Des dérives extrêmes peuvent même être imaginées, telle qu'une utilisation dans le cadre d'une société dirigée par l'analyse des données [39]. Il convient donc de garder le contrôle de la technologie, et d'encourager un réel débat sur les conséquences en terme de libertés individuelles, de l'analyse automatique des comportements des usagers de la route.

Enfin, il est aujourd'hui important de ne pas ignorer l'impact écologiques des technologies développées. Afin d'éviter une mise sur le marché inconsidérée de la technologie, une analyse du cycle de vie du dispositif doit être réalisée. Il faut alors garder à l'esprit que la complexité d'assemblage des matériaux associée à la micro électronique rend tout procédé de recyclage très peu rentable. L'utilisation des systèmes embarqués devrait donc être réservée à une utilisation apportant un réel gain, en évitant de tomber dans le fantasme d'une société dans laquelle l'intelligence artificielle est omniprésente. Par exemple, un appareil mobile servant à l'étude des routes serait peut-être plus rentable que des dispositifs permanents.

# Conclusion

Ce mémoire avait pour ambition de concevoir un capteur multimodal, permettant d'étudier les interactions entre les véhicules et les piétons. Dans ce but, un dispositif composé d'un radar et d'une caméra a été fabriqué, afin d'enregistrer le passage des usagers de la route.

Une stratégie de détection et de classification des objets a été mise en place. Les usagers de la route ont été répartis en 12 classes différentes, et les images prises par la caméra ont été analysées. Une précision de 88.1 % et un rappel de 83.9 % ont été atteint. L'utilisation du même algorithme pour l'identification des cibles radar a par ailleurs permis de démontrer la pertinence de la méthode *tiny-YOLO*, en dehors de son utilisation avec des images vidéos. Une précision de 88.2 % et un rappel 63.3 % ont été atteints, pour trois types d'objets différents enregistrés par le radar.

L'algorithme implémenté n'a pas permis d'étudier, de manière fiable et précise, l'interaction entre les véhicules et les piétons, malgré des résultats de fusion des données et de suivi des cibles encourageant. La fusion des données sur base d'une estimation de la distance des objets par la caméra, a été une limitation majeur de la précision des pistes. L'utilisation des angles capturés par le radar, ainsi que d'une méthode d'optimisation automatique seraient dès lors nécessaires.

L'évolution récente des méthodes d'apprentissage non supervisé, montre que l'utilisation de réseaux de neurones uniques pour résoudre les problèmes de détection et de classification des objets surpassé les méthodes en plusieurs étapes. La vitesse de détection ainsi que la précision de la classification sont améliorées. Une méthode de suivi de cibles multiples basée sur l'entraînement d'un seul réseau de neurones, permettrait peut-être dès lors d'unifier la détection, la classification et le suivi des cibles, et de simplifier le problème d'optimisation des paramètres.



# A Annexes

## A.1 Influence des paramètres de seuil sur le score-F1

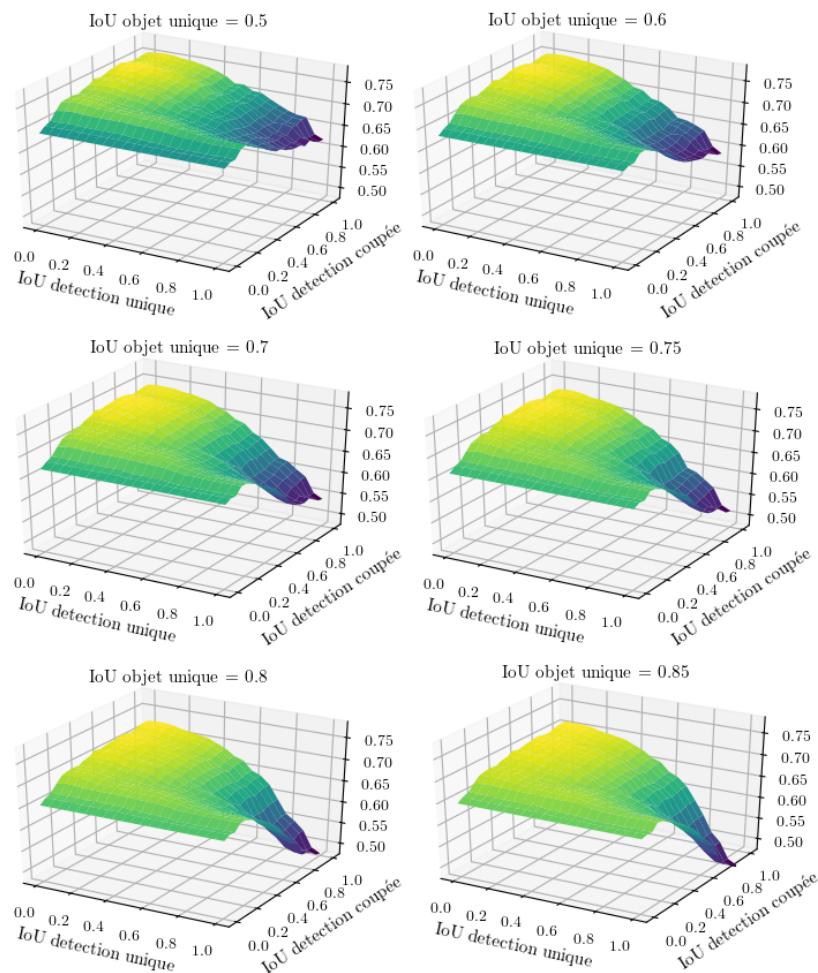


FIGURE A.1 – Évolution de score-F1 en fonction du paramètre d'IoU objet unique

## Annexe A. Annexes

---

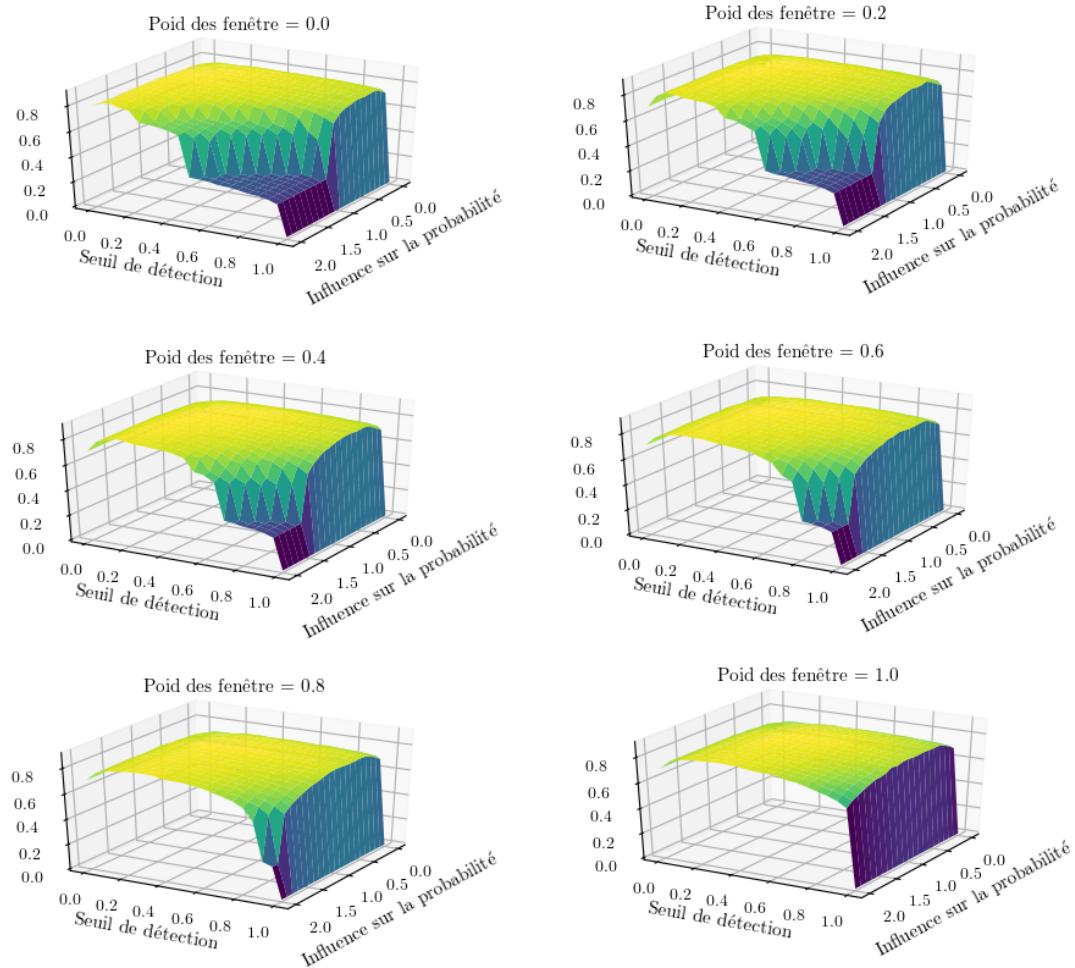


FIGURE A.2 – Évolution de score-F1 en fonction de l'influence accordée aux fenêtres

## A.2. Optimisation des paramètres de seuil par intervalles

### A.2 Optimisation des paramètres de seuil par intervalles

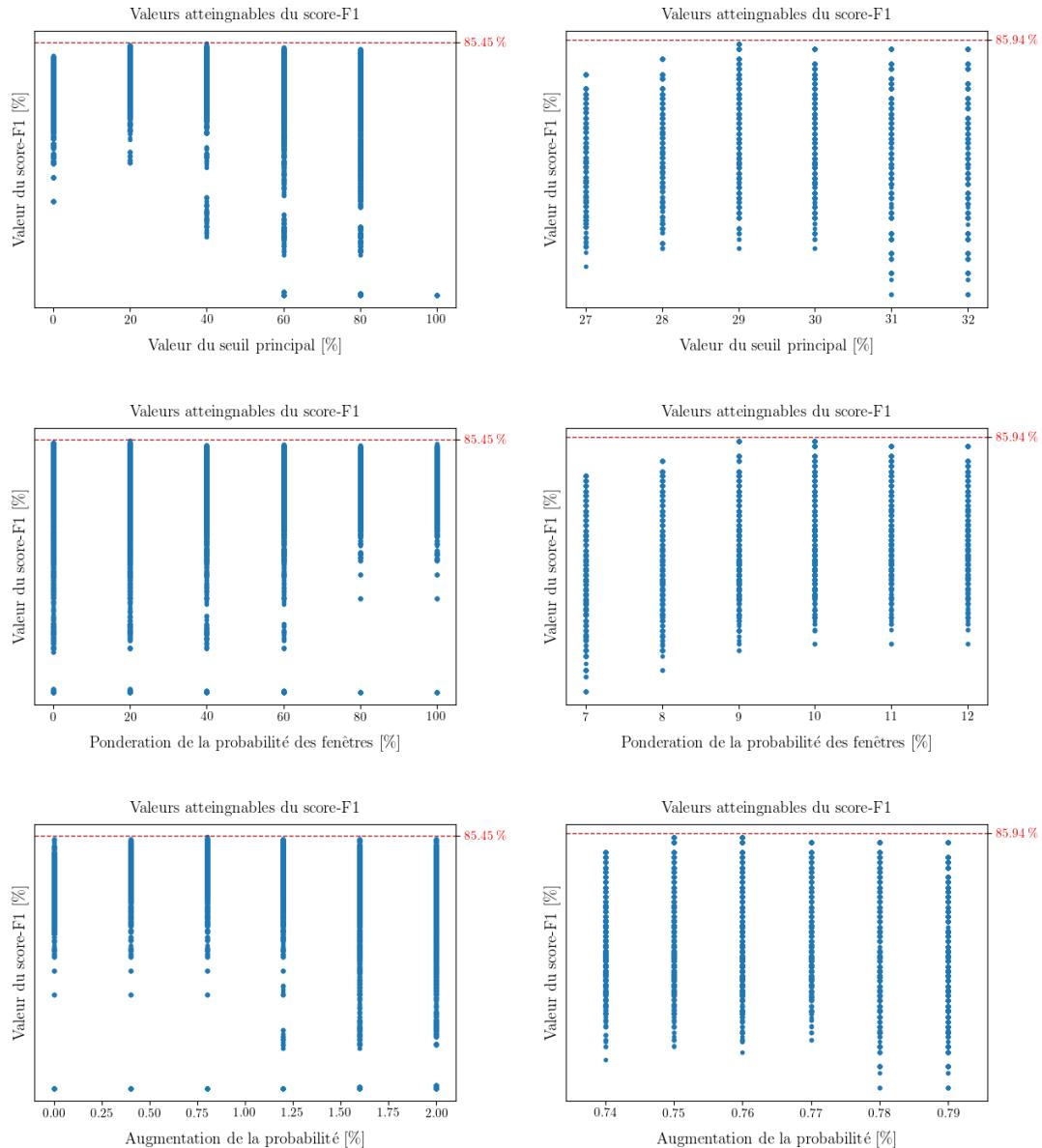


FIGURE A.3 – Influence des paramètres de seuil sur le score-F1

## Annexe A. Annexes

---

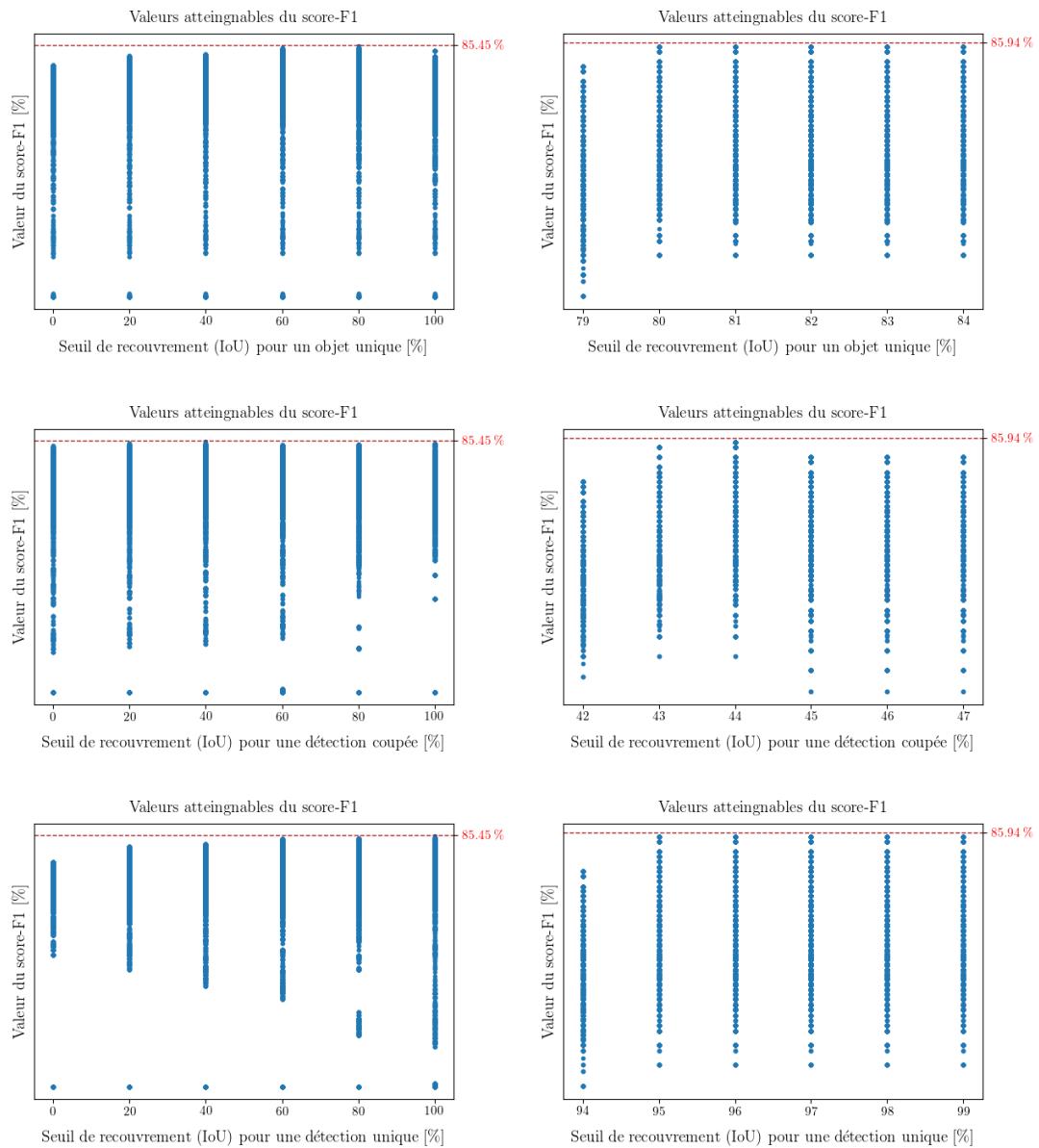


FIGURE A.4 – Influence des paramètres d'indices de Jaccard sur le score-F1

### A.3 Interactions sur le lieu d'étude

Séquence	Véhicule	Usager vulnérable	Confiance	Début	Fin	Distance minimale
1	sedan	personne	15	00:31.53	00:32.91	0.069
1	sedan	personne	3	00:33.51	00:33.60	0.132
1	sedan	personne	3	00:34.06	00:36.27	0.03
1	sedan	personne	3	00:37.50	00:37.54	1.107
1	sedan	personne	3	00:38.54	00:38.76	1.515
1	sedan	personne	4	00:42.55	00:44.83	0.233
1	sedan	personne	52	00:46.81	00:46.92	2.456
1	sedan	personne	15	01:09.06	01:09.16	2.983
1	sedan	personne	6	01:15.93	01:17.23	0.36
1	camion	personne	4	01:21.78	01:22.96	0.669
1	camion	personne	7	01:28.24	01:28.37	2.957
1	camion	personne	7	01:29.09	01:29.18	2.981
1	camion	personne	4	01:30.14	01:30.24	1.177
2	sedan	vélo	3	01:07.11	01:07.20	0.578
2	sedan	vélo	3	01:16.95	01:17.11	0.809
2	sedan	vélo	3	01:34.10	01:34.32	0.821
2	sedan	personne	3	01:37.11	01:37.17	0.12
2	sedan	personne	3	01:39.74	01:39.80	0.162
2	sedan	personne	3	01:43.67	01:43.81	0.981
2	sedan	vélo	3	01:46.61	01:46.77	0.558
2	sedan	vélo	3	01:54.79	01:54.99	0.802
2	sedan	vélo	3	01:58.48	01:58.61	0.569
2	sedan	vélo	3	02:09.20	02:09.29	0.61
2	sedan	vélo	4	02:51.44	02:51.61	0.818
2	sedan	personne	25	02:55.89	03:01.35	1.06
2	sedan	personne	4	03:04.32	03:05.71	0.964
4	sedan	personne	3	00:07.01	00:07.68	1.864
6	camionnette	personne	18	00:34.80	00:35.14	1.703
6	camionnette	personne	3	00:36.35	00:36.96	1.407
7	sedan	personne	8	00:02.07	00:02.08	1.837
7	sedan	personne	4	00:04.96	00:05.52	0.023
7	SUV	personne	4	00:06.21	00:06.59	2.199
7	SUV	personne	3	00:08.37	00:09.52	0.903
7	SUV	personne	4	00:10.61	00:10.64	2.905
7	sedan	personne	23	00:11.25	00:11.51	1.91
7	sedan	personne	3	00:12.38	00:12.41	2.766

TABLEAU A.1 – Résultats de l'analyse de la distance entre les usagers vulnérables et les véhicules. La couleur verte indique un vrai positif, la couleur bleue la présence des objets indiqués mais une distance réelle supérieure à 3 m, la couleur orange un faux positif avec un niveau de confiance inférieur à 3, et la couleur rouge un faux positif avec un niveau de confiance élevé. La confiance indique l'assurance de l'algorithme quand à la prédiction d'une distance inférieure à 3 m. Le faux positif en rouge sont donc plus graves que ceux en orange.



# Bibliographie

- [1] R. DI MARTINO, « Modelling and simulation of the dynamic behaviour of the automobile », Theses, Université de Haute Alsace - Mulhouse, jan. 2005. adresse : <https://tel.archives-ouvertes.fr/tel-00736040>.
- [2] N. KUDARAUSKAS, « Analysis of emergency braking of a vehicle », *Transport*, t. 22, juil. 2007. DOI : 10.1080/16484142.2007.9638118.
- [3] O. SEMICONDUCTOR. (2018). AR0230AT : CMOS Image Sensor, 2 MP, 1/3, adresse : <https://www.onsemi.com/products/sensors/image-sensors-processors/image-sensors/ar0230at>. (accessed : 14.10.2019).
- [4] ELP. (2019). ELP-USBFHD04H-L36 2MP 1080P, adresse : <http://www.webcamerausb.com/elpusbfd04hl36-2mp-1080p-cmos-ar0330-h264-mjpeg-yuy2-30fps-uvc-usb20-android-linux-windows-driverless-embedded-camera-module-mic-p-89.html>. (accessed : 14.10.2019).
- [5] S. CARDINAEL et A. MALCOURANT, « Conception d'un senseur intégré multimodale pour l'observation des routes », mém. de mast., Université catholique de Louvain, Louvain-la-Neuve, 2018.
- [6] R. M. GMBH. (2018). K-MD2 Engineering Sample, adresse : <https://www.rfbeam.ch/product?id=21>. (accessed : 14.11.2019).
- [7] E. GUERRERO-MENÉNZ, « Frequency-modulated continuous-wave radar in automotive applications. (English) », mém. de mast., Universitat Autònoma de Barcelona, Bellaterra, 2018.
- [8] *K-MD2 Engineering Sample*, Revision A, RFbeam Microwave GmbH, Schuppisstrasse 7, CH-9016 St. Gallen, 2018.
- [9] B.-S. KIM, Y. JIN, S. KIM et J. LEE, « A Low-Complexity FMCW Surveillance Radar Algorithm Using Two Random Beat Signals », *Sensors*, t. 19, p. 608, jan. 2019. DOI : 10.3390/s19030608.
- [10] T. FEUILLEN, C. XU, J. LOUVEAUX, L. VANDENDORPE et L. JACQUES, « Quantity over Quality : Dithered Quantization for Compressive Radar Systems », avr. 2019, p. 1-6. DOI : 10.1109/RADAR.2019.8835568.
- [11] NVIDIA. (2019). Jetson-Nano, adresse : <https://www.nvidia.com/fr-fr/autonomous-machines/embedded-systems/jetson-nano/>. (accessed : 21.11.2019).

## Bibliographie

---

- [12] R. P. FOUNDATION. (2019). Products, adresse : <https://www.raspberrypi.org/products/>. (accessed : 24.11.2019).
- [13] INTEL. (2019). Intel® Dual Band Wireless-AC 8265, adresse : <https://ark.intel.com/content/www/fr/fr/ark/products/94150/intel-dual-band-wireless-ac-8265.html>. (accessed : 24.11.2019).
- [14] P. SONG, N. ZHANG, H. ZHANG et F. GONG, « Blind Estimation Algorithms for I/Q Imbalance in Direct Down-Conversion Receivers », août 2018, p. 1-5. DOI : 10.1109/VTCFall.2018.8690669.
- [15] J. LÉGER, « Radar target classification based on micro-Doppler signature analysis. (English) », mém. de mast., Ecole polytechnique de Louvain, Université catholique de Louvain, Louvain-la-Neuve, 2016.
- [16] GATEWORKS. (2018). Linux Video4Linux2 API (v4l2), adresse : <http://trac.gateworks.com/wiki/linux/v4l2>. (accessed : 30.08.2019).
- [17] R. RASSHOFER et G. K, « Automotive Radar and Lidar Systems for Next Generation Driver Assistance Functions », *Advances in Radio Science - Kleinheubacher Berichte*, t. 3, mai 2005. DOI : 10.5194/ars-3-205-2005.
- [18] M. EVERINGHAM, L. VAN GOOL, C. WILLIAMS, J. WINN et A. ZISSERMAN, « The Pascal Visual Object Classes (VOC) challenge », *International Journal of Computer Vision*, t. 88, p. 303-338, juin 2010. DOI : 10.1007/s11263-009-0275-4.
- [19] Z.-Q. ZHAO, P. ZHENG, S.-t. XU et X. WU, « Object Detection with Deep Learning : A Review. (English) », *IEEE*, 2019. DOI : arXiv:1807.05511v2.
- [20] Z. ZOU, Z. SHI, Y. GUO et J. YE, *Object Detection in 20 Years : A Survey*, mai 2019.
- [21] J. REDMON, S. DIVVALA, R. GIRSHICK et A. FARHADI, « You Only Look Once : Unified, Real-Time Object Detection », juin 2016, p. 779-788. DOI : 10.1109/CVPR.2016.91.
- [22] J. REDMON et A. FARHADI, « YOLO9000 : Better, Faster, Stronger », juil. 2017, p. 6517-6525. DOI : 10.1109/CVPR.2017.690.
- [23] J. REDMON et A. FARHADI, « YOLOv3 : An Incremental Improvement », avr. 2018.
- [24] K. HE, X. ZHANG, S. REN et J. SUN, « Deep Residual Learning for Image Recognition », juin 2016, p. 770-778. DOI : 10.1109/CVPR.2016.90.
- [25] A. AB. (2019). Yolo-v3 and Yolo-v2 for Windows and Linux, adresse : <https://github.com/AlexeyAB/darknet>. (accessed : 06.12.2019).
- [26] T.-Y. LIN, G. PATTERSON, M. R. RONCHI et Y. CUI. (2019). Common objects in context, adresse : <http://cocodataset.org/#home>. (accessed : 06.12.2019).
- [27] M. EVERINGHAM, L. van GOOL, C. WILLIAMS, J. WINN et A. ZISSERMAN. (2017). Pascal VOC Dataset Mirror, adresse : <https://pjreddie.com/projects/pascal-voc-dataset-mirror/>. (accessed : 06.12.2019).
- [28] A. GEIGER, P. LENZ, C. STILLER et R. URTASUN. (2019). The KITTI Vision Benchmark Suite, adresse : <http://www.cvlibs.net/datasets/kitti/>. (accessed : 06.12.2019).

- [29] S. RUDER, « An overview of gradient descent optimization algorithms », sept. 2016.
- [30] K. ALEX, S. ILYA et E. HG, « Imagenet classification with deep convolutional neural networks », *Proceedings of NIPS, IEEE, Neural Information Processing System Foundation*, p. 1097-1105, jan. 2012.
- [31] C. LEE, G. OVERFITTING, R. CARUANA, S. LAWRENCE et L. GILES, « Overfitting in Neural Nets : Backpropagation, Conjugate Gradient, and Early Stopping », *Advances in Neural Information Processing Systems 13, NIPS 2000*, mar. 2001.
- [32] R. ROTHE, M. GUILLAUMIN et L. VAN GOOL, « Non-Maximum Suppression for Object Detection by Passing Messages between Windows », t. 9003, avr. 2015. DOI : 10.1007/978-3-319-16865-4\_19.
- [33] S. SMITH, P.-J. KINDERMANS et Q. LE, « Don't Decay the Learning Rate, Increase the Batch Size », nov. 2017.
- [34] M. MARRÓN-ROMERA, J. C. GARCIA GARCIA, M.-A. SOTELO, M. CABELLO, D. PIZARRO, F. HUERTA et J. CERRO, « Comparing a Kalman Filter and a Particle Filter in a Multiple Objects Tracking Application », nov. 2007, p. 1-6, ISBN : 978-1-4244-0830-6. DOI : 10.1109/WISP.2007.4447520.
- [35] M. KAVURI et B. KOLLA, « Performance Comparison of Detection, Recognition and Tracking Rates of the different Algorithms », *International Journal of Advanced Computer Science and Applications*, t. 10, p. 153, juil. 2019.
- [36] H. LINDELÖF BILSKI, *Heterogeneous Sensor Fusion : Verification and Optimization*, eng, Student Paper, 2017.
- [37] N. MAGDY, M. SAKR, T. ABDELKADER et K. ELBAHNASY, « Review on trajectory similarity measures », déc. 2015. DOI : 10.1109/IntelCIS.2015.7397286.
- [38] C. FEICHTENHOFER, A. PINZ et A. ZISSERMAN, « Detect to Track and Track to Detect », oct. 2017.
- [39] F. LIANG, V. DAS, N. KOSTYUK et M. HUSSAIN, « Constructing a Data-Driven Society : China's Social Credit System as a State Surveillance Infrastructure », *Policy and Internet*, t. 10, déc. 2018. DOI : 10.1002/poi3.183.



**UNIVERSITÉ CATHOLIQUE DE LOUVAIN**  
**École polytechnique de Louvain**

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | [www.uclouvain.be/epl](http://www.uclouvain.be/epl)