UCLouvain

École polytechnique de Louvain

Synchronization of Multimodal Data Flows for Real-Time AI Analysis

Synchronisation de flux multimodaux à des fins d'analyse en temps réel par l'IA

Auteur: **François Ledent**
Promoteurs: **Christophe Craeye, Benoît Macq**
Lecteurs: **David Bol, Gauthier Rotsart de Hertaing, Victorien Sonneville**
Année académique 2021–2022
Master [120] : Ingénieur Civil Électricien

Préface

Ces quelques pages sont le fruit de cinq belles années, riches et intenses. Aboutissement d'un agréable parcours, elles ouvrent l'horizon à de nouveaux engagements. Diversifié, le sujet plongera le lecteur au cœur de nombreux domaines : programmation haut et bas niveaux, traitement du signal et de l'image, algorithmes variés et introduction à l'intelligence artificielle. Ce texte couche sur le papier les principales réflexions et raisonnements fondamentaux qui menèrent, pas à pas, à la création d'un programme d'acquisition et de traitement bimodal. La constitution d'un jeu de données, spécifiquement dédié à l'entraînement de réseaux de neurones, lui confère sens et couleur.

J'exprime ma gratitude aux Pr. Craeye, Christophe et Macq, Benoît. Ceux-ci m'ont épaulé et conseillé ; leur motivation fut la mienne. Victorien Sonneville et Gauthier Rotsart de Hertaing m'ont eux accompagné plus étroitement tout au long de ce travail. Les discussions, aux sujets variés, furent enrichissantes et précieuses. Je les remercie pour leur engagement.

Je souhaite enfin témoigner ma reconnaissance à mes parents, pour leur présence attentionnée, et amis, sans qui la vie louvaniste n'aurait pas eu ce parfum particulier. J'adresse un merci spécial à celle qui partage mes heures depuis plusieurs années.

François Ledent
Louvain-la-Neuve, le 06 juin 2022.

Abstract

La perception de l'environnement mobilise les esprits depuis bien longtemps, particulièrement dans les domaines de la robotique et de l'électronique embarquée. Ce travail propose une approche bimodale alliant les facultés du radar aux atouts de la caméra CMOS et présente un programme autonome capable d'acquérir et traiter les images en provenance des deux capteurs, constituant des jeux de données spécifiquement dédiés à l'entraînement de réseaux de neurones. La fluidification de l'acquisition des mesures s'appuie sur les protocoles USB Video Class et TCP/IP, tire largement profit du parallélisme et contourne astucieusement les limites de `C`. Le traitement inclut la dérivation des graphes Doppler, ainsi que leur filtrage. Les méthodes d'interpolation et de compensation de mouvements servent à la génération des images appropriées. YOLO, CFAR et corrélation sont finalement étudiés lors de l'extraction du contenu et complètent la génération des jeux.

The perception of the environment has been a major research topic for a long time, especially in robotics and embedded electronics. This paper suggests a bimodal approach combining the advantages of the radar with the abilities of the CMOS camera. It provides an autonomous software designed to gather, process and assemble data from sensors into datasets dedicated to training of neural networks. An efficient acquisition process based on USB Video Class and TCP/IP protocols has been reached through multithreading technics and ingenious schemes to overcome `C` limitations. Once obtained, range-Doppler heatmaps are filtered. Interpolation and motion compensation are investigated to generate images at right times. YOLO, CFAR and a custom correlation method are finally applied to extract content from images and heatmaps. These conclude the dataset generating procedure.

Keywords | Embedded System, Multimodal Sensoring, Neural Network Dataset, Multithreading, Radar Signal Processing, EMA, YOLO, CFAR

Table des matières

Introduction **1**

1 Description du Dispositif **6**
1.1 Spécifications techniques de la caméra 7
1.2 Spécifications techniques du radar 8
1.3 Spécifications techniques du micro-contrôleur 10
1.4 Plan général du dispositif 11
1.5 Alimentation alternative pour des mesures sur le terrain 12

2 Récupération des Flux et Insertion des Estampilles **14**
2.1 Contexte et solution 15
2.1.1 Protocoles employés 15
2.1.1.1 TCP/IP 15
2.1.1.2 USB Video Class (UVC) 15
2.1.2 Structure générale de l'outil 16
2.1.2.1 Programme intermédiaire (motivation) 17
2.1.2.2 Fonctionnement de l'acquisition 17
2.2 Description des améliorations majeures 19
2.2.1 Multithreading 19
2.2.1.1 Parallélisme et indépendance 19
2.2.1.2 Protocole d'échange 20
2.2.2 Échange des horloges 21
2.2.2.1 Échange simple via TCP 21
2.2.2.2 Échange précis via le port Serial 21
2.2.3 Synchronisation de l'enregistrement 22
2.2.4 Reconnexion via time-out 22
2.2.5 Ajustement du nombre de buffers 23
2.3 Validation 25

3 Radar et Traitement des Trames **27**
3.1 Introduction aux radars . . . 27
3.1.1 Principe de fonctionnement . . . 27
3.1.2 Signaux en bande de base et bande passante . . . 28
3.1.3 Estimation de la distance . . . 29
3.1.4 Estimation de la vitesse et effet Doppler . . . 30
3.1.5 Dilemme Doppler . . . 30
3.2 Radar FMCW . . . 32
3.2.1 Modulation et formes d'ondes . . . 32
3.2.1.1 Signal reçu . . . 33
3.2.1.2 Échantillonnage . . . 34
3.2.2 Graphes Doppler . . . 35
3.2.2.1 Fast Fourier Transform . . . 35
3.2.2.2 Choix des paramètres . . . 39
3.2.2.3 Détermination des angles . . . 39
3.2.3 Zero padding . . . 41

4 Filtrage et Interpolation **44**
4.1 Filtrage des graphes Doppler . . . 44
4.1.1 Filtre à moyenne mobile (MA) . . . 45
4.1.1.1 Complexité . . . 46
4.1.1.2 Contenu fréquentiel . . . 46
4.1.2 Filtre triangulaire . . . 47
4.1.2.1 Complexité . . . 47
4.1.2.2 Contenu fréquentiel . . . 47
4.1.3 Filtre à moyenne mobile exponentielle (EMA) . . . 48
4.1.3.1 Complexité . . . 48
4.1.3.2 Contenu fréquentiel . . . 49
4.1.4 Comparaison et discussion . . . 49
4.1.4.1 Fréquence de coupure . . . 49
4.1.4.2 Dimensionnement des filtres . . . 49
4.1.4.3 Choix du filtre . . . 50
4.1.5 Validation . . . 52
4.2 Génération des images intermédiaires . . . 53
4.2.1 Mise en correspondance . . . 53
4.2.2 Interpolation . . . 53
4.2.2.1 Interpolation linéaire . . . 53
4.2.2.2 Interpolation de Lagrange . . . 55
4.2.3 Prédiction par estimation des mouvements . . . 57
4.2.3.1 Implémentation directe . . . 57
4.2.3.2 Exploitation de la compression H.264 . . . 57

5 Constitution du jeu de données **60**
5.1 Extraction du contenu des images 60
5.1.1 You Only Look Once (YOLO) 60
5.1.2 Validation 61
5.2 Extraction du contenu des graphes Doppler 63
5.2.1 Constant False Alarm Rate (CFAR) 63
5.2.2 Corrélation 63
5.2.3 Validation 64
5.3 Création du jeu de données 66
5.3.1 Implémentation logicielle 66
5.3.2 Parallélisme 67
5.3.3 Temps d'exécution 67
5.3.4 Résultat 69

Contribution **70**

Pistes d'amélioration **72**

Conclusion **74**

Annexe **77**
A.1 Spécifications techniques de la caméra, du radar et du Raspberry Pi 77
A.2 Spectre d'un peigne de Dirac 80
A.3 Déphasage entre les antennes 81
A.4 Comparaison des versions de YOLOv5 82
A.5 Détection des véhicules (séquences) 83
A.6 Comparaison CFAR et corrélation 84

Références **85**

Introduction

Présentée comme une des inventions majeures du XX^{e} siècle, l'automobile a été adoptée comme moyen de transport par excellence dans la plupart des régions du globe. Capable de transporter personnes et matériel en un temps raisonnable, elle s'est facilement imposée face à ses prédécesseurs, trop lents ou peu puissants. Toujours disponible et forte de sa polyvalence, elle envahit nos routes, villes et campagnes, contraints de s'adapter à ce brusque changement. Clé de voûte de notre mode de vie actuel, elle en devient banale et son omniprésence ne semble susciter aucun mécontentement, ni remise en question.

Pourtant, depuis son apparition, elle est la cause de nombreux accidents. Certains, spectaculaires, comme lors des Grands Prix de Formule 1 [1], font peut-être oublier le nombre et les victimes des accidents quotidiens, moins impressionnants. En 2020, près de trente mille accidents ont été recensés en Belgique, causant trois mille blessés sérieux et responsables de cinq cents décès [1]. Ces chiffres encouragent la quête d'une prise de conscience collective, mais aussi le développement de nouvelles technologies. Ces dernières mettent à la disposition des conducteurs une multitude d'informations et assistent leurs initiatives, via l'ABS (Anti-Blocking System) notamment. Elles permettent par ailleurs l'étude des comportements, par l'enregistrement de statistiques ou l'analyse en temps réel, et peuvent prévenir de situations dangereuses.

Assistant les équipes de terrain, les systèmes automatiques fixes ou embarqués sont déployés depuis plusieurs décennies. Ils sont constitués de deux éléments principaux : un capteur, habituellement une caméra, un radar ou un lidar, et une unité de calcul pour l'analyse des données collectées.

1. Le lecteur se souviendra de l'accident du pilote français Romain Grosjean au Grand Prix de Bahreïn en novembre 2020. Prisonnier de sa voiture en proie aux flammes, il s'en était finalement extrait, sain et sauf, après être resté plusieurs dizaines de secondes au cœur du brasier.

Les systèmes optiques ont été largement étudiés, certains articles remontant aux années quatre-vingt [2]. Ne contenant initialement qu'une unique caméra monoculaire [3], de nombreuses améliorations ont été apportées : résolution et fréquence d'acquisition (fps, pour *frames per second*) accrues, mais aussi vision stéréo [4] et recours aux caméras thermiques (à rayons infrarouges) [5].

Les radars, dont l'origine est antérieure à 1904 – année durant laquelle C. Hülsmeyer opéra la première démonstration à Cologne [6] –, ont eux aussi connu de nettes avancées depuis leur contribution durant la Seconde Guerre Mondiale. La fréquence des ondes émises n'a cessé de croître. On souligne par exemple sa progression au sein des applications automobiles, passant de 10 GHz en 1960 à 81 GHz, adopté comme standard ETSI (European Telecommunications Standards Institute) en 2004 [7]. Les hautes fréquences impliquent en effet une réduction de la taille des appareils et assurent une interférence mutuelle plus faible entre ceux-ci. Ce n'est cependant pas le seul progrès notable en ce qui concerne les radars. On peut citer l'acquisition comprimée (*compressed sensing* en anglais) [8][9], l'optimisation des formes d'onde [10] et la fusion des radars avec les outils de communications en un émetteur unique [11][12]. Comme l'indique *radio detection and ranging* dont est issu l'acronyme, les radars ont pour objectif la détection et la localisation d'objets. L'effet Doppler est en outre largement utilisé afin de déduire leur vitesse.

L'analyse des données récoltées par la caméra était historiquement réalisée par des outils de traitement d'images, tels que l'élimination de l'arrière-plan [13] et la détection de points saillants. On peut citer [14], où cette dernière est combinée à une minimisation aux moindres carrés de la différence entre les points saillants des images successives pour l'évaluation de la vitesse des véhicules. La détection d'arêtes est exploitée par [15], repérant les véhicules par leur ombre, plus sombre (hypothèse qui restreint fortement le champ d'application et ne permet pas d'extension à des cas plus complexes comme la nuit par exemple). L'article [16] applique également les moindres carrés, s'appuyant sur le changement de luminosité dû à la forme de la voiture (ce qui n'est donc pas extensible non plus). Ces deux travaux vérifient la symétrie des objets pour s'assurer qu'il s'agisse bien de véhicules. Cette démarche repose sur la présence d'une caméra *embarquée*, ce qui n'est pas le cas pour ce travail-ci. Un seuil adapté en temps réel pour l'extraction de l'arrière-plan, recourant à une caméra fixe, est exploré par [17]. D'autres algorithmes s'appuient sur le caractère aléatoire : des filtres de Kalman sont proposés dans [18][19][20]. Une connexion des voitures à Internet est mentionnée dans [21], où le traitement des données est opéré sur un serveur distant. L'algorithme CFAR semble être la seule méthode relatée dans la littérature pour l'analyse des données des radars. Il sera abordé au chapitre 5.

Force est de constater que l'intelligence artificielle (IA) détourne progressivement l'attention des chercheurs. Depuis une dizaine d'année, elle surclasse les méthodes traditionnelles et s'impose à bien des égards : classification, détection, segmentation furent les premiers exemples. La puissance de calcul et les bases de données disponibles depuis quelques années expliquent cette évolution. Divers travaux y font appel, comme [22] qui entraîne un classificateur SVM (*Support Vector Machine* en anglais) sur base d'une décomposition en ondelettes. Les radars profitent de ce nouvel élan et sont désormais capables d'ajuster eux-mêmes leurs paramètres internes [23], d'optimiser leurs formes d'ondes, de réduire le bruit de mesure ou encore de renforcer la lutte contre le brouillage [24].

Les systèmes actuels présentent cependant une limitation commune, et de taille : l'*unicité* du type de données sur lesquelles ils s'appuient. Ces ressources, le plus souvent des images RGB [2] ou trames d'un radar, démontrent chacune des avantages et inconvénients distincts, développés dans le prochain paragraphe, qu'il semble judicieux de combiner. Pourtant, la littérature ne contient que très peu d'articles relatant une telle association. On peut tout de même épingler [25][26][27][28]. Le premier, à l'instar des deux suivants, combine radar et caméra. Il expérimente un large réseau profond [3], résultat de l'entremêlement de deux plus petits : l'un pour les trames, l'autre pour les images. Le second papier présente une structure plus complexe composée de blocs distincts. Certains d'entre eux reposent sur des développements mathématiques spécifiques, d'autres ont recours à l'IA, comme un réseau convolutif (CNN, *Convolutional Neural Network* en anglais) pour l'analyse préliminaire des images. Un réseau antagoniste génératif (GAN, *Generative Adversarial Network* en anglais) est proposé par le troisième article, tandis que le dernier joint un lidar à la caméra et se sert de celui-ci pour identifier les zones d'intérêt. Ces zones sont mises en correspondance avec des parties de l'image, qui sont alors analysées par un réseau convolutif. L'apport du lidar est sans appel selon ses créateurs : il épargne au réseau l'étude de milliers de candidats potentiels, sélectionnant généralement moins d'une centaine de points d'attention. Les temps de calculs sont ainsi significativement réduits, maintenant tout de même un taux de rappel (pourcentage d'objets détectés par rapport au nombre total d'objets à détecter) supérieur à 95%. Ces échantillons témoignent de l'omniprésence de l'IA au cœur de l'ingénierie moderne.

2. Un des formats les plus courants pour la représentation d'images colorées est appelé RGB, pour *red*, *green*, *blue* en anglais. Ces trois couleurs permettent de générer toutes celles visibles par l'œil humain en ajustant leur pondération.

3. La dénomination anglaise *deep network* est habituellement employée en français pour désigner un réseau neural à plusieurs couches.

La diversité des capteurs garantit une couverture large et surtout plus robuste de la scène. Il est à ce titre essentiel d'être conscient de leur forces et faiblesses respectives. La caméra présente l'avantage de fournir des images faciles à manipuler et interpréter, lesquelles sont par ailleurs un atout pour la reconnaissance d'objets grâce aux réseaux convolutifs et base de données en plein essor. Elle est cependant tributaire de la luminosité et des conditions météorologiques : sa fiabilité est largement altérée la nuit ou en cas de fortes pluies. Enfin, efficace pour la détection des mouvements transversaux (mouvements perpendiculaires à l'axe de l'objectif), la caméra ne l'est pas autant pour l'évaluation des mouvements radiaux (mouvements parallèles à l'axe de l'objectif). Les radars et lidars mesurent quant à eux la distance radiale qui les séparent de l'objet. Ils sont de ce fait appropriés pour l'appréciation des mouvements radiaux, mais peinent à estimer ceux transversaux. Les radars ont d'autre part du mal à convaincre pour la reconnaissance des objets, souffrant certainement plus d'un manque de considération que d'une réelle impossibilité, comme le montrent certains travaux [29].

C'est dans ce contexte que s'inscrit ce projet. Celui-ci se distingue de la plupart des travaux antérieurs par la pluralité de ses senseurs, joignant les avantages de la caméra à ceux d'un radar. Mis en place il y a plusieurs années, il compte plusieurs contributions et a été imaginé comme suit : 1) récupération des flux, 2) extraction du contenu, 3) analyse des trajectoires, 4) anticipation des trajectoires futures à l'aide de l'IA et 5) finalité adaptée à la situation (statistiques ou intervention en temps-réel). Les étapes 1) à 4) ont été rapidement passées en revue par Alexis Duflot dans [30], ce qui a permis à Kevin De Sousa et Gauthier Rotsart de Hertaing de se concentrer davantage sur les points 3) et 4). Ceux-ci ont comparé diverses architectures : un filtre de Kalman, un Perceptron Multi-Couches (MLP) et un Transformer. Ils constatèrent des défaillances notables au sein du mécanisme d'acquisition et confirmèrent la nécessité d'un programme dédié et optimisé. Mieux compris, les protocoles de communication pourraient assurer un débit stable, sans pertes ni corruption. L'importance d'une meilleure synchronisation s'est également fait ressentir : les données de la caméra et du radar étaient jusqu'à présent associées à leur réception, considérant en quelque sorte les mesures enregistrées simultanément comme prises au même instant. Il existe pourtant un délai variable (et long) entre la capture et la réception des mesures du radar qui rend inadéquate une telle hypothèse.

Ces observations constituent l'objet de cette contribution-ci : fluidifier la récupération des flux par la bonne compréhension des protocoles ainsi que l'apport de techniques de programmation adaptées, et les synchroniser par l'insertion d'estampilles. Par la suite, au vu de la bonne progression du travail, s'ajouta un second objectif : la constitution de jeux de données destiné à l'entraînement de réseaux

de neurones. Ce manuscrit est divisé en 5 chapitres dans lesquels sont exposés les choix, réflexions et raisonnements qui menèrent à la réalisation d'un tel programme.

Le chapitre 1 présente brièvement le matériel et le montage, de sorte à immerger le lecteur dans le contexte de ce travail. Après avoir exposé les protocoles TCP/IP et USB Video Class, le chapitre 2 aborde l'acquisition des flux et l'insertion des estampilles. Le lecteur y découvre comment le multithreading peut garantir une latence minimale et l'indépendance des threads de récupération. Un peu de créativité permit en outre de relever plusieurs défis et de rendre l'outil plus commode pour l'utilisateur, lui offrant par exemple la possibilité de se connecter et déconnecter sans avoir à redémarrer le radar.

Les chapitres suivants se basent sur l'acquisition solide du chapitre 2 et sont consacrés au second objectif. Le chapitre 3 introduit le fonctionnement du radar et met en lumière la manière avec laquelle les graphes Doppler, ou graphes distance–vitesse, sont obtenus à partir de l'enveloppe complexe qu'il fournit. Une fois dérivés, les graphes sont filtrés. C'est le sujet du chapitre 4. On souhaite en plus former des paires images–graphes capturés *exactement* au même instant. Les capteurs ne garantissent évidemment pas une telle concordance, si bien qu'il est nécessaire de générer de nouvelles images : celles qui *auraient* été prises en même temps que les graphes. L'interpolation est explorée, tant dans sa version linéaire que dans celle de Lagrange, mais c'est l'estimation de mouvements qui procure les meilleurs résultats. Ceci est aussi discuté au chapitre 4. L'extraction du contenu des images est assurée par YOLO, un réseau de neurones conçu à cet effet en 2016, tandis que celle des graphes est opérée par CFAR, algorithme mentionné ci-dessus, et une méthode originale impliquant une corrélation. Les trois approches engendrent de très bons résultats (taux de rappel, précision), exposés et comparés au chapitre 5. Ce dernier relate enfin la manière dont s'agencent tous les thèmes des chapitres 3 à 5 au sein du logiciel. Quelques pistes d'amélioration clôturent ce TFE.

Chapitre 1

Description du Dispositif

Ce projet vise à mesurer l'impact de la multimodalité des données, c'est-à-dire la diversité à l'égard de leur nature, sur l'étude d'un environnement par l'intelligence artificielle. Son aboutissement requiert la constitution d'un système mobile et bon marché d'acquisition et d'analyse. Celui-ci doit être compact et son transport commode, afin d'être placé aisément aux abords des routes et carrefours. Idéalement autonome ou contrôlable à distance, il doit permettre la capture simultanée d'images et de trames radar, ainsi qu'assurer leur traitement. La possibilité d'une commercialisation future du dispositif n'est pas exclue. C'est pourquoi une attention particulière a été accordée à son coût.

La bimodalité est une des caractéristiques essentielles du dispositif proposé, qui comprend trois unités principales : les deux capteurs, la caméra et le radar, et le micro-contrôleur, le Raspberry Pi. La caméra a été achetée par S. Cardinael et M. Malcourant en 2018, le radar par C. Deleval et M. Lahy en 2017, tous anciens mémorants sur ce sujet. Comme le rapporte A. Duflot dans [30], le choix des composants n'est pas anodin et répond à plusieurs contraintes.

Premièrement, la résolution et le rapport signal-à-bruit des données doivent être suffisants pour une distinction précise et fiable des objets. La portée maximale attendue est 80 mètres. Deuxièmement, l'angle de capture doit garantir un champ de vision assez large qui comprend entièrement la route et ses abords, soit approximativement 30°. Troisièmement, la fréquence d'acquisition doit être relativement élevée pour s'assurer que l'hypothèse d'une vitesse constante entre deux images soit raisonnable. Partant du principe que l'accélération maximale d'une voiture est estimée à environ 1.2 fois l'accélération de la pesanteur [31][32], 11.8 captures par seconde sont nécessaires pour restreindre la variation de vitesse à moins de $1\ \mathrm{m} \cdot \mathrm{s}^{-1}$ entre deux apparitions. À ces conditions s'ajoutent celles mentionnées

dans le paragraphe précédent, à savoir la taille et le coût.

Les prochaines sections décrivent les caractéristiques de la caméra, du radar et du micro-ordinateur de ce projet. La caméra et le radar ont été choisis bien avant le lancement de ce travail-ci. Ils répondent aux attentes précédemment citées, mais certaines de leurs caractéristiques, qui semblaient moins importantes au premier abord, s'avèreront être contraignantes. Il en va ainsi pour le radar par exemple. N'enregistrant pas l'instant de capture des trames, il a rendu l'insertion des estampilles particulièrement périlleuse.

1.1 Spécifications techniques de la caméra

La caméra, représentée à la FIG.1.1, est commercialisée par ELP. Ses principales spécifications techniques sont rassemblées en annexe, dans la TABLE A.2. Elle contient un capteur CMOS AR0330 fabriqué par ON Semiconductor [1]. Celui atteint un taux de 30 images par seconde (ou fps, pour *frames per second*) avec une résolution FULL HD, c'est-à-dire 1920 × 1080, soit un peu plus de deux millions de pixels par image.

FIGURE 1.1 – Caméra contenant un capteur AR0330 et vendue sous forme de module prêt à l'emploi par ELP. Extrait de [33].

Le flux vidéo est disponible en format YUYV, MJPEG ou H.264 (nom usuel de la norme MPEG-4/AVC) et transmis via un câble USB. La puce est contrôlable via USB Video Class, protocole de gestion de périphériques électroniques vidéo. La section 2.1.1.2 détaille son fonctionnement.

Elle présente enfin quelques avantages supplémentaires, comme la gestion automatique de l'exposition. Cela facilite son emploi dans des environnements à luminosité changeante. Elle offre également un contrôle du contraste, de la netteté, de la teinte, de la saturation, et de la balance des blancs, qu'il peut être judicieux d'ajuster pour des prises de vue particulières.

1. Website : `https://www.onsemi.com`. Fiche technique disponible à l'adresse (après requête auprès du fabricant) : `https://www.onsemi.com/products/sensors/image-sensors/ar0330`.

1.2 Spécifications techniques du radar

Le radar fut développé par la société RFbeam Microwave [2]. Il contient une antenne émettrice et trois antennes réceptrices, disposées en rectangle comme le montre la FIG. 1.2. Un signal est émis sans interruption. Celui-ci est modulé en fréquence : la fréquence instantanée croit linéairement, retombant régulièrement à la fréquence initiale, ce qui produit un profil en dents de scie. Il s'agit d'une modulation FMCW (Frequency Modulated Continuous Wave). En français, on appelle cela un radar à ondes entretenues modulées en fréquence.

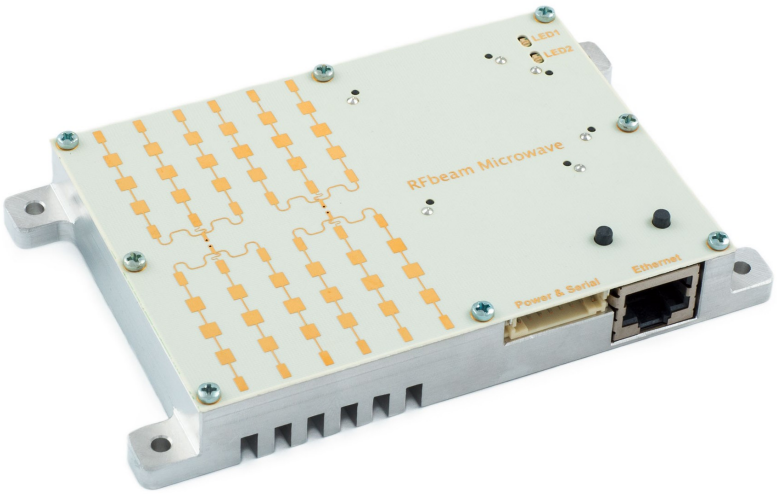

FIGURE 1.2 – Radar K-MD2 fabriqué par RFbeam Microwave. On distingue les 4 antennes sur la gauche. L'antenne émettrice se trouve dans le coin inférieur gauche. Extrait de [34].

La partie analogique est composée des antennes, des amplificateurs, des mélangeurs et de l'oscillateur. Ceux-ci sont visibles sur la partie droite de la FIG. 1.3. Pour le traitement des signaux digitaux, le radar utilise un SoC (System On Chip) Zynq XC7Z020. Ce dernier est monté sur une carte d'adaptation TE720 de la marque Trenz Electronic. Cette carte contient deux processeurs ARM Cortex-A9, et une partie programmable incluant le FPGA (Field-Programmable Gate Array). Un OS (Operating System), à savoir une distribution Linux, assure le bon fonctionnement du radar.

Le radar comprend deux mémoires : la mémoire volatile et celle non-volatile. La mémoire volatile, RAM (Random Access Memory) sur la FIG. 1.3, constitue la

2. `https://www.rfbeam.ch/product?id=21`

mémoire de travail. C'est elle qui est employée lorsque des données ou variables sont manipulées temporairement par les processeurs (pour être exact, ceux-ci intègrent également des caches L1 et L2) ou le FPGA. Son contenu est perdu lorsque le radar est mis hors tension. La commande `cat /proc/meminfo` indique que sa capacité est de 221,052 kB, soit 215 MB. Contrairement aux données de la mémoire RAM, l'information retenue dans la mémoire non-volatile, Flash sur la FIG. 1.3, est toujours disponible après une mise hors tension. Celle-ci sert de lieu de mémorisation des programmes et constantes relatifs à l'OS, ainsi que ceux et celles liés aux tâches exécutées au sein du radar. Une part restreinte peut être modifiée par l'utilisateur : c'est là que sera inséré le programme `prgrm_rdr.c` présenté au chapitre suivant.

Grâce au FPGA, au centre de la FIG. 1.3, le radar est capable de traiter lui-même les trames et ainsi générer les graphes distance–vitesse. Néanmoins, l'objectif restant de se tourner vers un radar plus simple et moins cher à long terme, seules les données brutes seront exploitées dans ce travail. Deux interfaces, Serial et Ethernet, permettent l'alimentation, le transfert des données et le contrôle du radar, dont la TABLE A.3 de l'annexe A.1 reprend les spécifications techniques principales. La fiche technique [35] comprend des données supplémentaires.

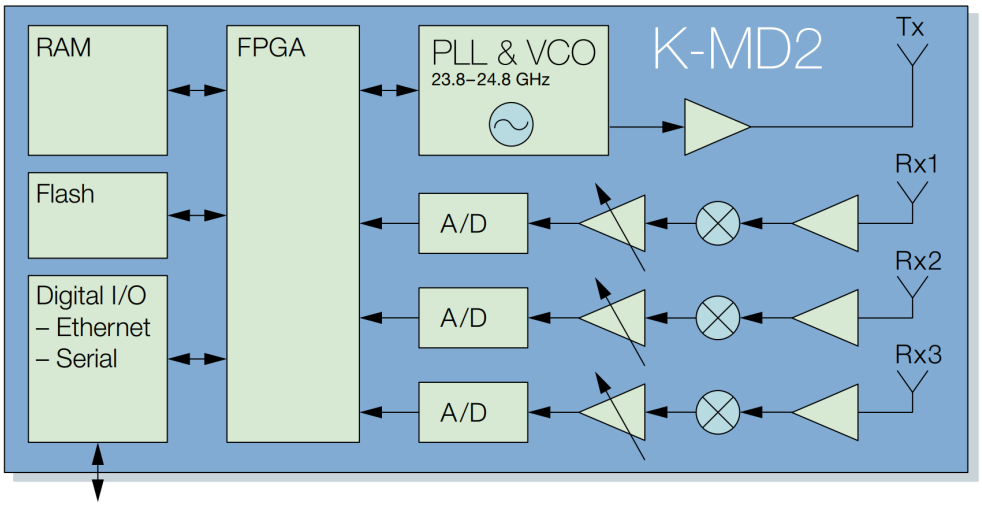

FIGURE 1.3 – Schéma fonctionnel du radar K-MD2. On y distingue le FPGA au centre. À gauche se trouvent la mémoire volatile (RAM), non-volatile (Flash) et les interfaces Ethernet et Serial. Les antennes et le traitement analogique se situent à droite. Image extraite de [35].

1.3 Spécifications techniques du micro-contrôleur

Le micro-contrôleur utilisé dans ce travail est un Raspberry Pi 4B. Comme le résume [36], le Raspberry Pi est un micro-contrôleur monocarte à processeur ARM de la taille d'une carte de crédit, conçu par des professeurs du département informatique de l'université de Cambridge dans le cadre de la fondation Raspberry Pi. Celle-ci est basée au Royaume-Uni et cherche à promouvoir et démocratiser l'accès à l'informatique [37]. Elle s'appuie pour cela sur des partenariats, des OS et logiciels libres de droit, ainsi qu'une large communauté bénévole.

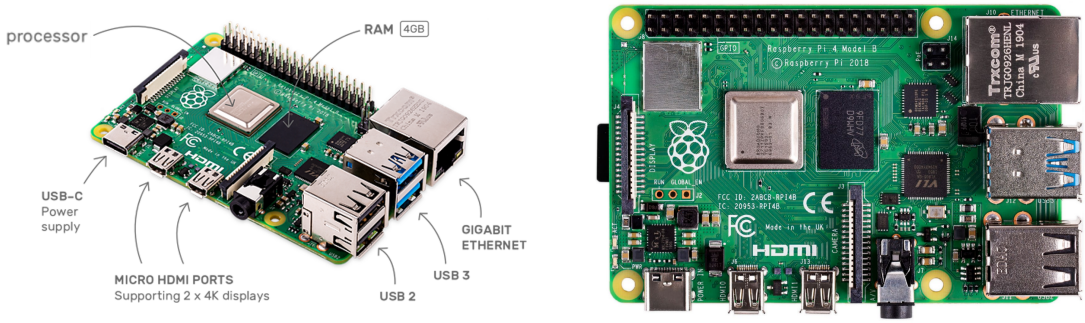

FIGURE 1.4 – Raspberry Pi 4B, doté de son processeur, sa mémoire vive et ses connecteurs. Gauche : adapté de [38]. Droite : extrait de [39].

Le Raspberry Pi, malgré son prix très abordable, offre une puissance de calcul satisfaisante grâce à son processeur quad-core Cortex-A72 (ARM v8) 64-bit @ 1.5 GHz. Il est également pourvu de quatre connecteurs USB, un connecteur USB-C (alimentation), deux connecteurs Micro-HDMI, un connecteur Ethernet et la sortie Jack 3.5 mm pour le son. Ceux-ci le rendent très polyvalent. Les 4 GB de mémoire vive (RAM) dont il dispose [3] sont largement suffisants pour l'utilisation d'IDE (Integrated Development Environments) [4], comme Visual Studio Code, et l'exécution des programmes développés dans le cadre de ce projet. Les quatre coeurs rendent possible un réel parallélisme, réduisant l'impact de l'OS sur les autres processus exécutés. Il est même possible de rendre l'OS et les programmes développés encore plus indépendants en associant un ou plusieurs coeurs à ce premier, et les coeurs restants aux seconds. Pour être exact, les coeurs ne sont pas parfaitement indépendants puisqu'ils partagent les mêmes périphériques (la

3. Le Raspberry Pi 4B est disponible avec 1, 2, 4 ou 8 GB de mémoire vive. Le prix varie légèrement avec la taille de la mémoire. Celui utilisé pour ce TFE dispose de 4 GB.

4. Un environnement de développement est un ensemble d'outils qui permet d'augmenter la productivité des programmeurs. Il comporte un éditeur de texte destiné à la programmation, un compilateur ou l'éditeur de liens ainsi qu'un débogueur.

mémoire vive notamment). Pour des programmes ne faisant pas massivement usage de ces périphériques, leur partage induit un impact très faible en comparaison avec un système qui ne contiendrait qu'un seul coeur.

Le Raspberry Pi a été conçu pour des applications diverses, employé comme serveur, routeur, stockage distant mais aussi en domotique et robotique. Il est prévu pour être déployé dans des environnements très variables et de nombreux dissipateurs de chaleur, ventilateurs, étuis, existent pour le protéger de la surchauffe et des conditions météorologiques difficiles. C'est pourquoi, au regard de ses performances, de sa polyvalence et des accessoires protecteurs disponibles sur le marché, il semblait être le micro-ordinateur embarqué adéquat.

1.4 Plan général du dispositif

La FIG. 1.5 propose un modèle simple pour l'alimentation, usant du réseau de distribution comme source. Il peut être employé pour le développement en laboratoire et pour les prises de vue proches d'un point d'accès au réseau électrique. On y retrouve les trois éléments principaux : la caméra, le radar et le Raspberry Pi. Les liaisons par lesquelles les données transitent sont marquées en gris et les protocoles rappelés : USB Video Class et TCP/IP.

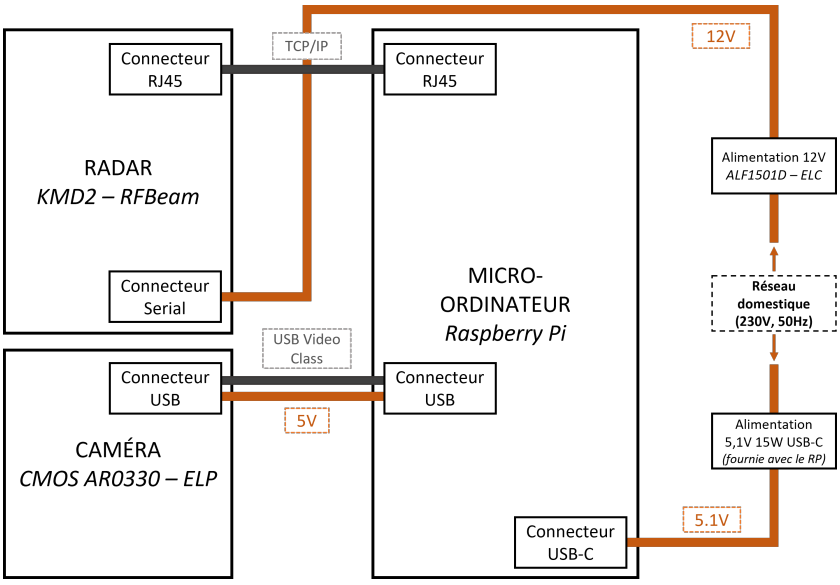

FIGURE 1.5 – Vue schématique des connexions entre les blocs et de leur alimentation. Les traits oranges et gris indiquent les flux de puissance et d'information (données, contrôle), respectivement.

La gestion de la puissance est indiquée en orange. Le Raspberry Pi est relié au réseau par l'intermédiaire de l'alimentation 5.1V-15W ad hoc. C'est lui qui, par

le biais de la connexion USB, approvisionne la caméra. Une alimentation permet la jonction entre le réseau et le radar. Cette configuration, illustrée en orange, garantit une tension et un courant constants et ininterrompus. L'alimentation est une ALF1501D commercialisée par ELC. La tension de sortie est variable, entre 10V et 15V (12V dans la cas présent) [40].

L'environnement de travail est visible ci-dessous, à la FIG. 1.6. Le chariot a été conçu avec l'aide précieuse et agréable de Souley Djadjandi, Benoît Hubert et Miloud Zitout. Merci pour leur disponibilité.

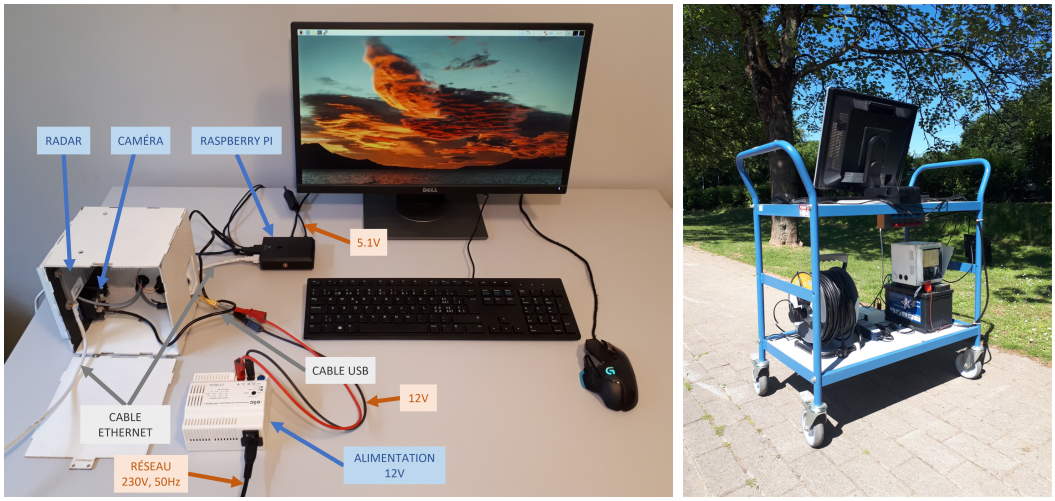

FIGURE 1.6 – Environnement de travail. Gauche : configuration pour le développement software à l'intérieur. Droite : chariot conçu pour les prises de mesure et la vérification du bon fonctionnement du programme sur le terrain.

1.5 Alimentation alternative pour des mesures sur le terrain

Puisque le dispositif doit pouvoir être déployé le long des routes, il est nécessaire de prévoir un plan d'alimentation adapté à ces environnements dépourvus d'accès au réseau. Pour cela, le schéma proposé dans la section précédente est légèrement modifié, considérant maintenant 2 batteries Lithium-Ion rechargeables comme source d'énergie. La tension à leurs bornes vaut, en valeur nominale, 11.1V. Ceci ne pose pas de problème pour le radar, qui supporte une tension d'alimentation comprise entre 10 et 16V. Chaque batterie dispose de 57.7 Wh (énergie totale), puisque sa capacité nominale vaut de 5200 mAh.

La tension des batteries correspond à celle du radar : tous deux peuvent être directement couplés. Le Raspberry Pi, lui, accepte une tension de 5.1V. Cela explique pourquoi un convertisseur Buck (abaisseur de tension) est intégré au système. Il réduit la tension de 11.1 à 5V.

Les vérifications d'usage ont été réalisées dans [30] pour des batteries similaires, dont la capacité était de 2200 mAh. Une autonomie de 2 heures y était estimée, ce qui signifie que le dispositif tel que présenté dans cette section, avec deux batteries de 5200 mAh, peut compter sur une autonomie de 4 heures et 45 minutes. Il est utile de noter que, si une telle autonomie n'est pas nécessaire, une unique batterie suffit. L'autonomie est dans ce cas réduite de moitié. Le schéma alternatif est illustré à la FIG. 1.7.

Les batteries, modèle 18650 3S29, sont conçues par la firme Ansmann. Bien que le produit y soit référencé, aucune fiche technique n'est disponible. Elle est en revanche accessible auprès des distributeurs [5]. Le convertisseur Buck, modèle SD-25A-5, est produit par Mean Well [6].

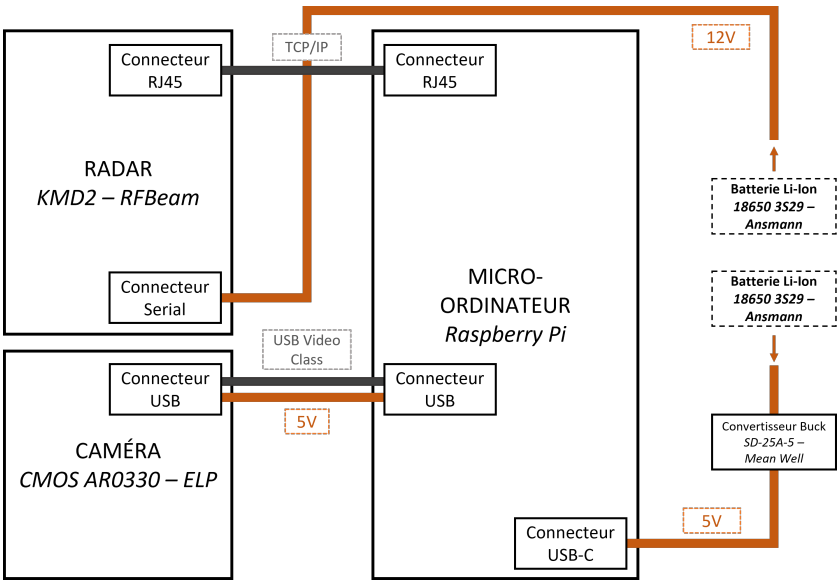

FIGURE 1.7 – Vue schématique des connexions entre les blocs et de leur alimentation (config. alternative dans le cas où aucun accès au réseau n'est possible). Les traits oranges et gris indiquent les flux de puissance et d'information (données, contrôle), respectivement.

5. https://asset.conrad.com/media10/add/160267/c1/-/gl/001547351DS01/datasheet-1547351-ansmann-3s2p-battery-pack-6x-18650-cable-li-ion-111-v-5200-mah.pdf

6. https://www.meanwell-web.com/en-gb/dc-dc-enclosed-converter-input-9-2-18vdc-output-sd--25a--5

Chapitre 2

Récupération des Flux et Insertion des Estampilles

Après une brève présentation des protocoles TCP/IP et USB Video Class, ce chapitre entame la description de l'outil. Celui-ci se compose de deux programmes principaux exécutés par le Raspberry Pi : `acquisition.c` et `traitement.py`. Les noms évocateurs indiquent qu'ils correspondent respectivement aux deux objectifs majeurs auxquels cherche à répondre cette contribution : la fluidification de l'acquisition des données par une compréhension approfondie des protocoles et la mise en place de techniques de programmation adaptées, ainsi que la formation de jeux d'entraînement pour des algorithmes d'IA grâce au traitement des données récoltées [1]. Le présent chapitre décrit l'organisation générale de l'outil puis se focalise sur l'acquisition, les chapitres suivants abordant le traitement.

L'instant de capture des trames et images est enregistré dans ce qu'on appelle des estampilles (ou *timestamps* en anglais). Leur rôle est essentiel pour la synchronisation des deux flux. En lisant attentivement la documentation de la caméra et du protocole qui la contrôle, on se rend compte qu'elle peut fournir les estampilles des images, rendant leur insertion triviale. C'est en revanche loin d'être le cas pour le radar, qui nécessita l'introduction d'un programme supplémentaire `prgrm_rdr.c` (exécuté par le radar) et plusieurs manipulations décrites ci-après. À celles-ci s'ajoutent quelques améliorations notables qui rendent l'outil plus commode à utiliser. Le nombre de buffers est finalement étudié, avant que la validation ne clôture ce chapitre, démontrant les bonnes performances du mécanisme.

1. Le lecteur intéressé peut consulter la contribution de Kevin De Sousa et Gauthier Rotsart de Hertaing [41] sur le sujet. Les réseaux qui y sont présentés souffraient d'un manque de données fiables et seront ré-entraînés à l'aide du jeu généré par le présent travail. Une nette amélioration des résultats est attendue par leur auteur.

2.1 Contexte et solution

2.1.1 Protocoles employés

2.1.1.1 TCP/IP

TCP/IP signifie TCP *sur* IP, où les deux acronymes correspondent à *Transmission Control Protocol* et *Internet Protocol*. Ils sont souvent assimilés aux couches Internet (IP) et de transport (TCP) du modèle qui porte leur nom, soit TCP/IP. Celui-ci structure la communication entre deux appareils distants en 4 couches successives, de l'application (Mozilla Firefox par exemple) au niveau physique (comme la tension dans le câble Ethernet). Celles-ci permettent des niveaux d'abstraction progressifs, de sorte que l'une ou l'autre puisse être modifiée sans qu'une révision générale ne soit requise.

Le rôle de la couche de transport consiste à découper les fichiers à transmettre en une succession de courts fragments. La particularité de TCP, c'est qu'il garantit l'intégrité et l'ordre des segments. Il leur adjoint pour cela un entête avec les informations nécessaires. À la réception, le protocole se charge lui-même de remettre les fragments dans l'ordre et sollicite l'expéditeur pour leur renvoi en cas de perte. Garantir l'intégrité du message facilite la gestion des transferts, mais induit un délai variable et important lors de pertes [2].

2.1.1.2 USB Video Class (UVC)

Le protocole USB Video Class (UVC) facilite la gestion de périphériques électroniques vidéo connectés aux ports USB. Il permet le transfert des images ainsi que le contrôle de la caméra (fréquence et format des images, type d'encodage ou encore nombre de buffers [3] employés pour la communication).

Pour recevoir les images de la caméra, le protocole prévoit la création de buffers et de deux queues, celle d'écriture et celle de lecture. Lorsqu'une image vient d'être enregistrée dans un buffer par la caméra, celle-ci le place dans la file de lecture qui regroupe tous les buffers remplis et prêts à être lus par le Raspberry Pi. Une fois que le contenu d'un buffer a été extrait par le micro-contrôleur, ce dernier rejoint la file d'écriture pour un nouveau cycle. Il s'agit d'une organisation *first in first out* (FIFO), le premier buffer intégrant une queue étant aussi le premier retiré.

2. Le protocole UDP, qui ne garantit ni l'intégrité ni l'ordre des messages et induit par là un délai nettement plus court que TCP, n'est pas disponible avec le radar utilisé. La garantie de l'intégrité des trames est en outre très pratique et évite d'en délaisser pour cause de pertes.

3. Le terme «buffer» peut être traduit par «mémoire-tampon» en français, mais la dénomination anglaise demeure la plus couramment employée.

2.1.2 Structure générale de l'outil

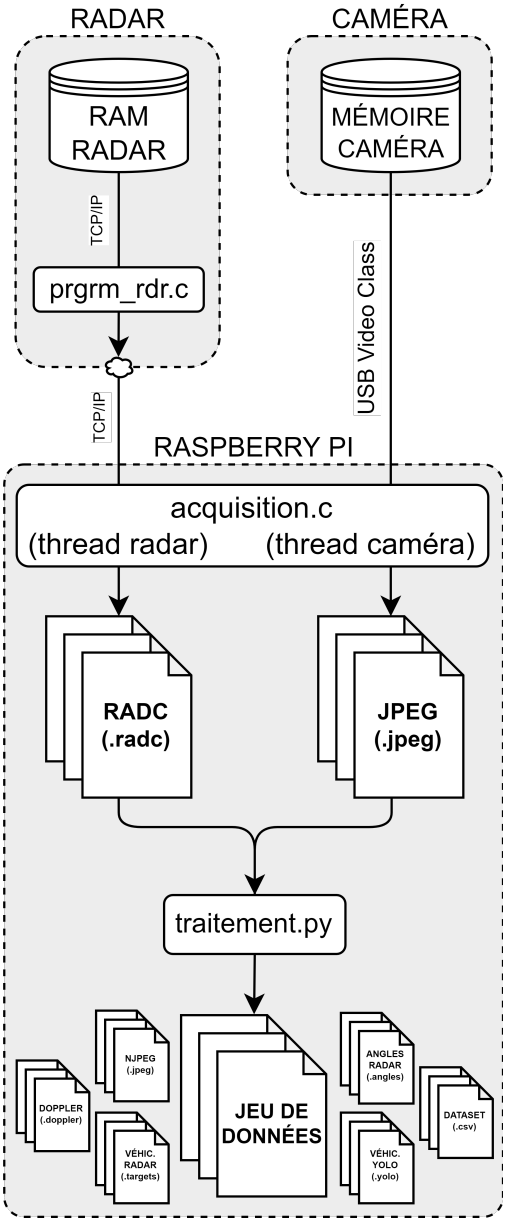

FIGURE 2.1 – Organisation générale de l'outil. On distingue deux phases : la récupération des flux (haut) et leur traitement (bas).

La structure générale de l'outil développé dans ce TFE est visible à la FIG. 2.1. Dans la partie haute a lieu la récupération des flux. Elle est responsable de l'enregistrement des trames et des images dans les dossiers `RADC` (pour *raw ADC values*) et `JPEG` du micro-contrôleur, le Raspberry Pi. Le traitement des mesures se trouve dans la partie inférieure et génère les jeux de données.

Les capteurs observent la scène et stockent en premier lieu les trames (radar) ou images (caméra) dans leur mémoire respective (cf. `RAM RADAR`, `MÉMOIRE CAMÉRA`). Elles sont ensuite envoyées au Raspberry Pi à l'aide des protocoles TCP/IP et UVC. C'est le programme `acquisition.c`, exécuté par le Raspberry Pi, qui communique de manière indépendante avec les deux capteurs. L'insertion des estampilles du radar nécessite l'ajout d'un programme intermédiaire `prgrm_rdr.c` exécuté au sein du radar. Le serveur qu'il met en place pour interagir avec `acquisition.c` est représenté par le symbôle *cloud* sur la figure ci-contre.

Une fois les trames et images enregistrées par le micro-contrôleur dans les dossiers `RADC` et `JPEG` respectivement, la mission de l'acquisition est accomplie et celle-ci laisse place au traitement, qui peut être réalisé en même temps que l'acquisition ou a posteriori. Ce dernier se chargera d'extraire les informations utiles et constituer les sets de données. Le code `traitement.py` est abordé dans la section 5.3.1.

2.1.2.1 Programme intermédiaire (motivation)

Le radar et le Raspberry Pi communiquent à l'aide d'une liaison TCP, qui, comme expliqué dans la section 2.1.1.1, introduit un délai dont la durée dépend du taux de pertes lors de la transmission : plus il y a de pertes, plus le nombre de retransmissions augmente et plus le délai s'allonge. Puisque l'instant de capture des trames n'est pas disponible et que les estampilles doivent être manuellement rajoutées, il est indispensable de les insérer au plus près de leur lieu (et instant) de capture, c'est-à-dire dans le radar directement. On évite ainsi les délais qui pourraient intervenir dans la communication TCP entre le radar et le Raspberry Pi. Bien sûr, la communication TCP entre le serveur interne du radar (appelé loopback [4]) et `prgrm_rdr.c` induit également un légère latence, mais celle-ci demeure sans commune mesure avec celle qui fut évitée. On peut en effet considérer la communication interne comme quasi-parfaite avec un taux d'erreur minime et, par conséquent, très peu de retransmissions. Concrètement, le programme `prgrm_rdr.c` se connecte au loopback et met lui-même en place un second serveur auquel se connecte `acquisition.c`. Le délai introduit lors de ce second échange est de moindre importance puisque les estampilles ont déjà été rajoutées.

2.1.2.2 Fonctionnement de l'acquisition

Le lecteur est désormais invité à se plonger au coeur de `prgrm_rdr.c` et `acquisition.c` (partie supérieure de la FIG. 2.1). Ces deux programmes sont conjointement responsables de l'acheminement des trames depuis le serveur interne du radar et des images depuis la caméra jusqu'à leur enregistrement dans la mémoire du Raspberry Pi. Un schéma-bloc, visible à la page suivante, résume leur fonctionnement et la manière dont les données transitent.

Le lecteur observe tout d'abord 5 threads : 2 pour `prgrm_rdr.c` et 3 pour `acquisition.c`. Le multithreading est abordé en 2.2.1. Un protocole d'échange entre threads d'un même programme a été mis au point, comme décrit dans la même section. Il assure la jonction entre les threads ①, ② de `prgrm_rdr.c` et ①, ③ de `acquisition.c`. L'insertion des estampilles est chose facile pour les images car la caméra agit comme un simple périphérique de l'ordinateur. Le radar possède en revanche son propre OS et, par conséquent, sa propre horloge sur laquelle est basé `prgrm_rdr.c`. Le Raspberry Pi doit connaître le décalage entre cette horloge et la sienne pour être capable d'interpréter les estampilles (cf. section 2.2.2). En plus, un système simple synchronise le début de l'enregistrement des trames et celui des images (en 2.2.3). Un mécanisme laisse enfin à l'utilisateur la possibilité de se connecter plusieurs fois sans devoir redémarrer le radar (en 2.2.4).

4. Le loopback est un serveur permettant l'interaction de processus en interne.

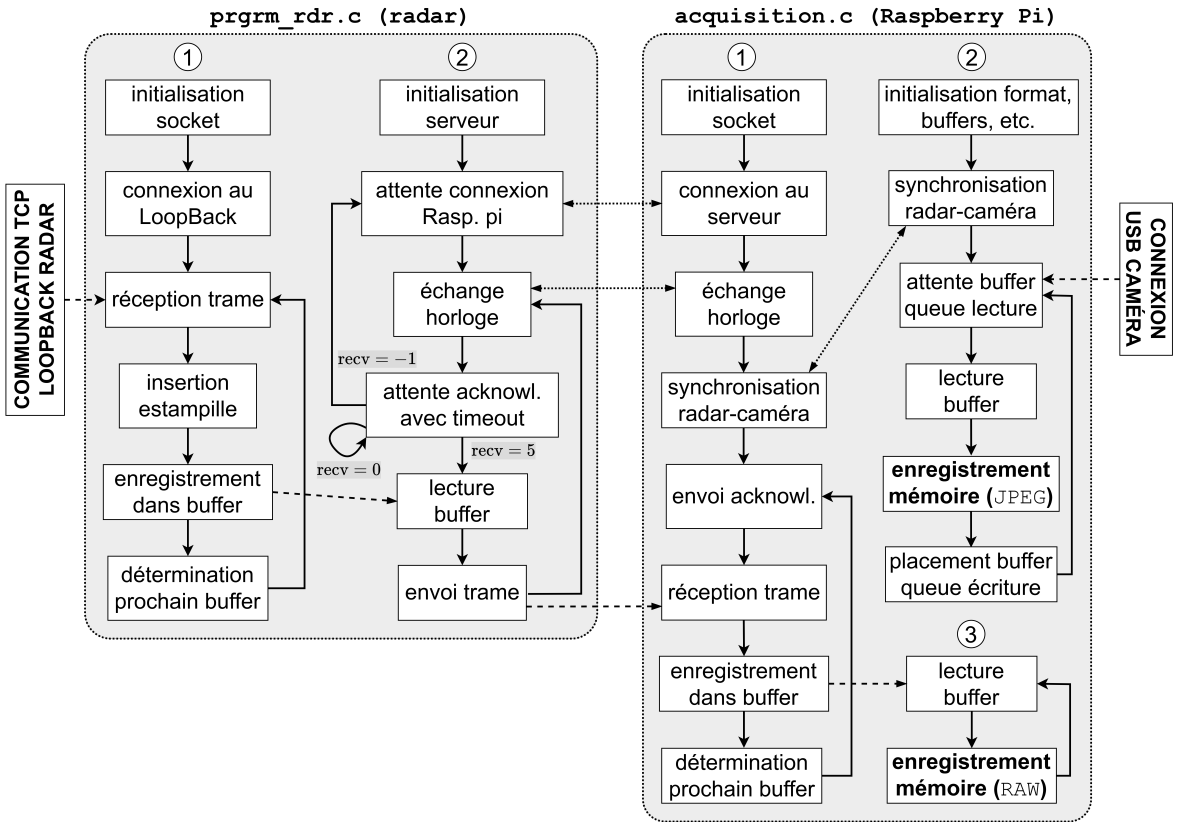

FIGURE 2.2 – Schéma-bloc de la partie «acquisition» (par opposition à celle du «traitement»). Les deux programmes principaux qui la composent sont `prgrm_rdr.c` (exécuté par le radar) et `acquisition.c` (exécuté par le Raspberry Pi). Les flèches discontinues indiquent la progression des trames/ images jusqu'à leur enregistrement dans les dossiers `RAW` (trames) et `JPEG` (images). Les flèches en pointillés relient les blocs qui communiquent expressément entre eux.

2.2 Description des améliorations majeures

2.2.1 Multithreading

Un programme consiste en une suite d'opérations élémentaires que le processeur, sans indication particulière, exécute séquentiellement, l'une après l'autre. Ce code pourrait par exemple être celui à gauche dans la FIG. 2.3. On observe que la séquence contient deux parties distinctes qui n'interagissent pas entre elles. Elles sont marquées par des couleurs différentes. Si on indique cette indépendance à l'ordinateur en définissant 2 threads (notés par ①, ②), il pourra associer chaque partie à un coeur différent, et ainsi clôturer l'exécution en $\max(2, 3) = 3$ unités de temps au lieu de 5.

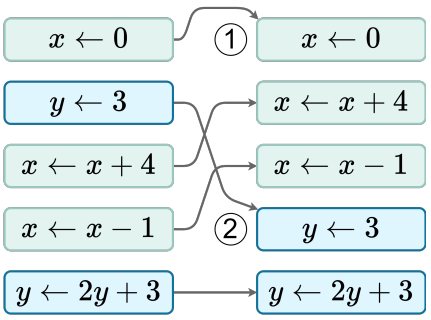

FIGURE 2.3 – Exemple d'exécution séquentielle puis parallèle.

2.2.1.1 Parallélisme et indépendance

Le principe de multithreading a été largement exploité lors de l'implémentation de l'outil, dont le schéma-bloc fut illustré à la FIG. 2.2, avec 2 threads pour `prgrm_rdr.c` et 3 pour `acquisition.c`. En réalité, ce n'est pas tant pour accélérer l'exécution que le parallélisme est primordial, mais plutôt pour rendre les parties indépendantes. Concentrons-nous en premier lieu sur `prgrm_rdr.c` qui contient 2 threads : ① assure la connexion avec le loopback, recevant les trames, leur insérant une estampille puis les stockant dans un buffer, tandis que ② communique avec le Raspberry Pi (par l'intermédiaire du thread ① de `acquisition.c`). Il est indispensable qu'un ralentissement de l'une ne se répercute pas sur l'autre. Elles doivent par contre avoir accès un espace de mémoire commun afin de partager les buffers employés pour le stockage temporaire. Ceci correspond précisément au rôle du parallélisme : il indique au processeur comment réaliser chaque tâche indépendamment, tout en conservant celles-ci au sein d'un même programme [5], ce qui leur permet d'interagir par le biais de variables communes (les buffers essentiellement). Pour ce rendre compte de cette indépendance, il suffit d'imaginer que l'opération $x \leftarrow x + 4$ de la FIG. 2.3 induise un long délai. L'exécution de ① est ralentie, mais cet événement est sans incidence sur celle de ②. Il en est de même pour les threads ①, ② de `prgrm_rdr.c` et ①, ②, ③ de `acquisition.c`.

5. Le partage de mémoire et l'interaction entre des programmes distincts sont particulièrement périlleux à cause des protections mises en place pour empêcher la corruption (in)volontaire des données de programmes tiers.

2.2.1.2 Protocole d'échange

Le souhait de tâches indépendantes mais qui puissent interagir peut paraître antagoniste. Il est important de comprendre qu'exécuter une tâche sans tenir compte de la seconde n'empêche pas d'utiliser des variables communes. On considère en effet que les temps d'écriture et lecture sont si courts que l'une et l'autre n'ont jamais lieu en même temps (des mécanismes de protection garantissent cette hypothèse). Ceci n'est en revanche plus possible lorsqu'il s'agit d'écrire et de lire des milliers de bytes (il y a 768 kB dans chaque trame). C'est pourquoi un protocole d'échange des buffers a été élaboré. Il doit garantir l'indépendance des tâches (l'une ne peut pas bloquer l'autre) et que la trame lue est toujours la plus récente (tant pis pour celles perdues [6]). Il assure la jonction entre les threads ①, ② de `prgrm_rdr.c` et ①, ③ de `acquisition.c`.

Pour ce faire, 3 buffers sont créés et 3 labels distribués : `LAST_EDITED`, `OTHER_A` et `OTHER_B` ; comme en témoigne la FIG. 2.4. Lorsque l'écriture d'un buffer est terminée – par le thread α dans cet exemple – (A), il lui assigne le label `LAST_EDITED`. Il intervertit pour cela le label du buffer en question avec celui du buffer qui portait la mention `LAST_EDITED` jusqu'à lors (B). Il détermine ensuite le buffer disponible, c'est-à-dire celui qui n'est pas en train d'être lu par le Raspberry Pi, et y enregistre la trame suivante (C). De l'autre côté, lorsque la lecture est accomplie (D), le thread qui en est responsable – le thread β dans cet exemple – n'a plus qu'à trouver le buffer marqué par `LAST_EDITED` (E) et initie la lecture du buffer correspondant (F).

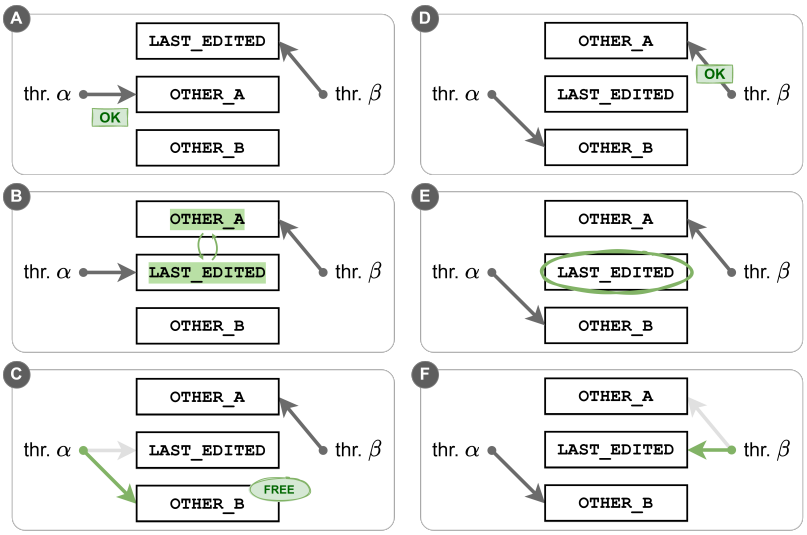

FIGURE 2.4 – Protocole mis sur pied pour l'échange de buffers. Il garantit l'indépendance des deux parties (l'une ne peut pas ralentir l'autre) et que le buffer lu correspond au dernier rempli, identifié par le label `LAST_EDITED`.

6. Puisque la communication est le facteur limitant, tout retard est impossible à résorber.

2.2.2 Échange des horloges

Le programme intermédiaire rajoute des estampilles au plus proche de leur lieu de capture, soit directement dans le radar, lequel n'a pas la même horloge que le Raspberry Pi. Son horloge est en effet réinitialisée à chaque mise hors tension et se met en route au démarrage du radar. On parle d'horloge monotone. Pour que le Raspberry Pi puisse interpréter l'estampille, il faut connaître à quel instant cela correspond pour lui, ce qui revient à déterminer le décalage t_0 entre l'horloge du radar et celle du Raspberry Pi. Deux méthodes sont proposées ci-dessous.

2.2.2.1 Échange simple via TCP

Une fois le Raspberry Pi connecté au serveur du radar, ce premier introduit une requête au second, qui lui envoie alors la valeur de son horloge. Même si l'échange semble rapide pour l'œil humain, plusieurs centaines de micro-secondes s'écoulent entre la requête et la réponse, notamment à cause du protocole TCP. Cependant, ce délai est fortement variable et il est possible d'en tirer profit : on peut essayer jusqu'à ce que le délai soit jugé suffisamment court. Dans ce travail, on considère qu'il doit être inférieur à 200 us, comme en atteste la FIG. 2.5. On garantit ainsi que l'erreur sur t_0 y est elle aussi inférieure. Elle se rapproche même de $200/2 = 100$ us si on suppose que l'aller et le retour sont de durée similaire.

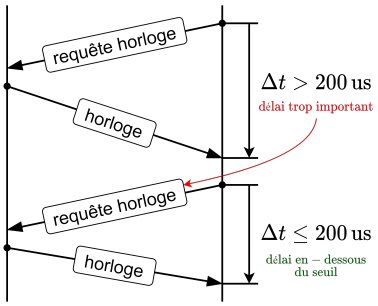

FIGURE 2.5 – Protocole d'échange de l'horloge.

Quelques échanges (< 100) suffisent généralement pour satisfaire la condition, ce qui est en réalité très rapide puisqu'ils ne durent pas plus de 300 us chacun. Pour être certain de ne pas bloquer indéfiniment, la limite de 200 us est relaxée tous les 2000 essais par un incrément de 20 us.

2.2.2.2 Échange précis via le port Serial

Pour l'utilisateur qui souhaiterait une estimation (encore) plus précise de t_0, il serait envisageable de combiner les ports Ethernet et Serial. En changeant l'état d'un des pins du port Serial ($0 \to 1$ ou $1 \to 0$) à l'instant où la valeur de l'horloge du radar est envoyée, le Raspberry Pi saurait exactement quand a eu lieu l'envoi de la réponse et recevrait sa valeur quelques instants plus tard par TCP.

2.2.3 Synchronisation de l'enregistrement

Grâce aux variables qu'ils partagent, les threads ① et ② de `acquisition.c`, respectivement responsables de la gestion de l'enregistrement des trames et des images dans la mémoire du Raspberry Pi, sont capables de se synchroniser avant le début de l'enregistrement. Cela permet de ne pas enregistrer de trames ou d'images seules, car elles seraient par la suite abandonnées lors de la mise en correspondance entre les deux flux.

Un mécanisme simple suffit. Deux booléens, un pour ①, l'autre pour ② sont initialisés à `0`. Une fois qu'un thread est prêt pour l'enregistrement, il met à jour le booléen qui lui est assigné, changeant sa valeur en `1`. La routine de synchronisation dans laquelle il entre ensuite vérifie si les deux booléens sont bien actifs. Si ce n'est pas le cas, elle patiente 200 ms puis vérifie à nouveau. Une fois que les deux threads ont atteint cette routine, et donc que les deux booléens sont actifs, elle s'achève et ceux-ci débutent leur enregistrement respectif.

2.2.4 Reconnexion via time-out

Un des défis qui exigea le plus de créativité fut certainement celui-ci : permettre à l'utilisateur de se connecter et se déconnecter autant de fois que souhaité et ce sans qu'il ne soit nécessaire de redémarrer le radar. Le thread ② de `prgrm_rdr.c`, qui maintient le serveur auquel se connecte le thread ① de `acquisition.c`, doit pour cela détecter la déconnexion de son client lorsqu'elle a lieu et guetter une potentielle reconnexion. Le langage `C` est cependant un langage bas niveau qui ne propose au premier abord aucune méthode d'alerte lorsque le client se déconnecte.

Pour contourner l'absence de fonctions adéquates dans les librairies de `C`, le système suivant a été mis sur pied. Avant d'envoyer chaque trame, le radar attend un message de validation (en l'occurrence «`READY`») de la part du Raspberry Pi. Si celui-ci est bien reçu, le radar sait que le Raspberry Pi est connecté et lui transmet la trame. S'il n'a rien reçu, il conclut que le Raspberry Pi n'est pas disposé à recevoir le message et patiente 200 us avant de tenter à nouveau. Enfin, il est aussi possible qu'une erreur se présente, ce qui se traduit par un renvoi de `-1` par la fonction de lecture des messages TCP. Le radar est par ce biais informé que le client est déconnecté. Il arrête alors l'envoi des trames et bascule vers l'attente de connexion pour permettre la reconnexion. Le mécanisme apparaît dans la FIG. 2.2 (cf. threads ② de `prgrm_rdr.c` et ① de `acquisition.c`).

Il n'est pas utile de vérifier le message de validation caractère par caractère : connaître le nombre de bytes reçus (baptisé `recv`) suffit. Ceci explique pourquoi

l'observation de `recv` $= 5$, `recv` $= 0$ ou `recv` $= -1$ garantit la distinction entre entre les 3 cas susmentionnés.

Pour forcer la génération d'une erreur en cas de déconnexion de l'hôte, il est important de fixer un court time-out (qu'on peut traduire par «durée d'expiration»). Il garantit que la fonction de lecture des messages TCP reverra une valeur endéans le délai choisi. Sans cela, elle attendrait indéfiniment un message du client quand celui-ci n'est plus connecté, à la place de retourner une erreur.

2.2.5 Ajustement du nombre de buffers

Le nombre de buffers à allouer pour le protocole USB Video Class a été optimisé. Il peut varier entre 1 et 32 (limite définie par la caméra). Pour évaluer l'impact de ce paramètre, appelé n ci-dessous, des essais ont été réalisés. Durant des enregistrement avec différentes valeurs de n, le nombre d'images réceptionnées a été échantillonné chaque seconde (sur base des instants indiqués par les estampilles). Le nombre d'images reçues par seconde est appelé taux, en *frame per second* (fps). La variation de celui-ci dans le temps peut être caractérisée à l'aide d'un histogramme car il permet d'avoir une idée précise de la répartition du taux sur l'ensemble d'une séquence. Dans la FIG. 2.6 sont résumés les histogrammes obtenus avec $n = 4$ (haut), $n = 18$ (milieu), $n = 32$ (bas) [7]. On observe deux facteurs : la stabilité et le taux moyen. Dans ce contexte, stabilité signifie absence de variation et peut être quantifiée par la variance du taux autour de sa moyenne, marquée en bleu. On souhaite en effet que les images reçues soient uniformément réparties afin de rendre leur interpolation, présentée à la section 4.2, plus précise.

On constate que la stabilité et le taux moyen s'améliorent tous les deux avec le nombre de buffers alloués (la moyenne croît, la variance décroît), ce qui correspond au résultat attendu. En effet, plus le nombre de buffers est important, plus les appareils évoluent librement l'un par rapport à l'autre. Pour s'en convaincre, discutons le cas critique $n = 1$. La caméra qui écrit dans les buffers et le Raspberry Pi qui les lit sont dépendants l'un de l'autre : puisqu'il y a qu'un seul buffer, ils ne peuvent pas travailler simultanément et l'un doit attendre que l'autre ait terminé avant de commencer. Avec $n = 2$, ils peuvent travailler simultanément mais le premier qui a terminé devra attendre le second (puisque les buffers passent de l'un à l'autre). En poursuivant ce raisonnement, on comprend que plus il y a de buffers, plus les processus seront indépendants et plus l'acquisition sera fluide, accroissant par cette occasion stabilité et taux moyen. $n = 32$ est dès lors le nombre de buffers retenu.

7. D'autres valeurs ont été testées mais ne sont pas présentées par souci de concision.

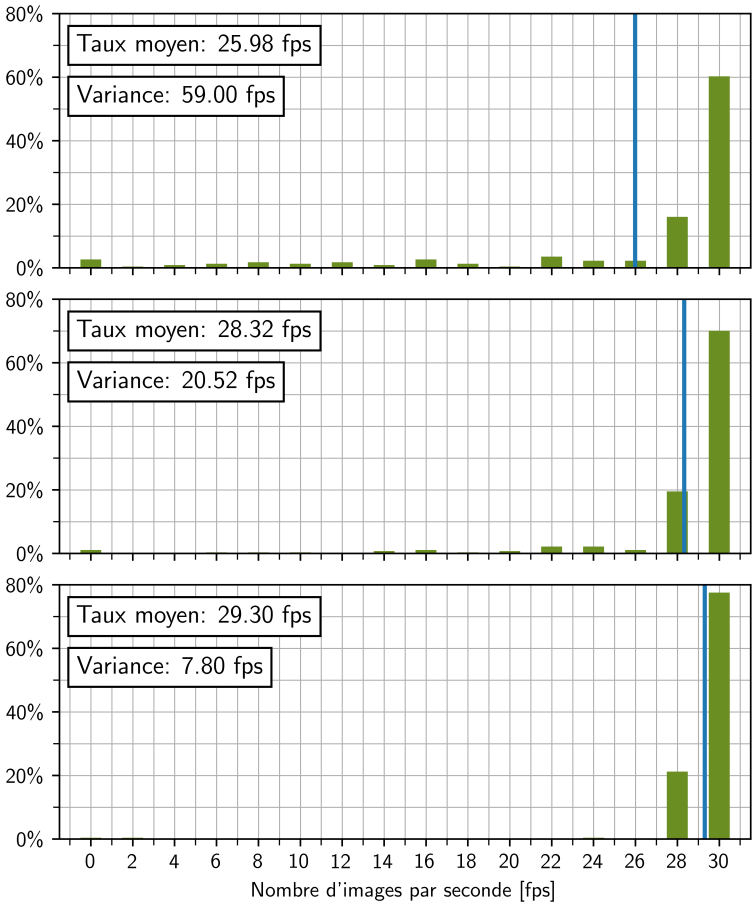

FIGURE 2.6 – Représentation des essais réalisés, avec 4 (haut), 18 (milieu) et 32 (bas) buffers. Le nombre d'occurrences (sur 220 sec.) est normalisé et exprimé en pourcentage. La ligne bleue représente la moyenne. On constate que la stabilité et le taux augmentent avec le nombre de buffers alloués. Celui-ci est cependant limité à 32 par la caméra.

Note : les valeurs proches de zéros sont obtenues les premières secondes de l'enregistrement. C'est en quelque sorte un «effet de bord».

2.3 Validation

Une fois tous les mécanismes établis vint le temps des premières prises de vue. Celles-ci ont confirmé le bon fonctionnement du système. L'une d'entre elles est affichée sur la FIG. 2.7. On retrouve le nombre de trames et d'images par seconde durant toute la durée de l'enregistrement (seuls ont été retirés les premiers échantillons pour éviter l'impact du démarrage), ainsi que les histogrammes récapitulatifs. Les objectifs sont clairs : se rapprocher aussi près que possible des taux théoriques, soit 30 fps pour les images et 12 fps pour les trames (le radar n'est pas capable d'en émettre davantage). Le résultat est très satisfaisant : le dispositif atteint 29.59 images et 11.69 trames par seconde.

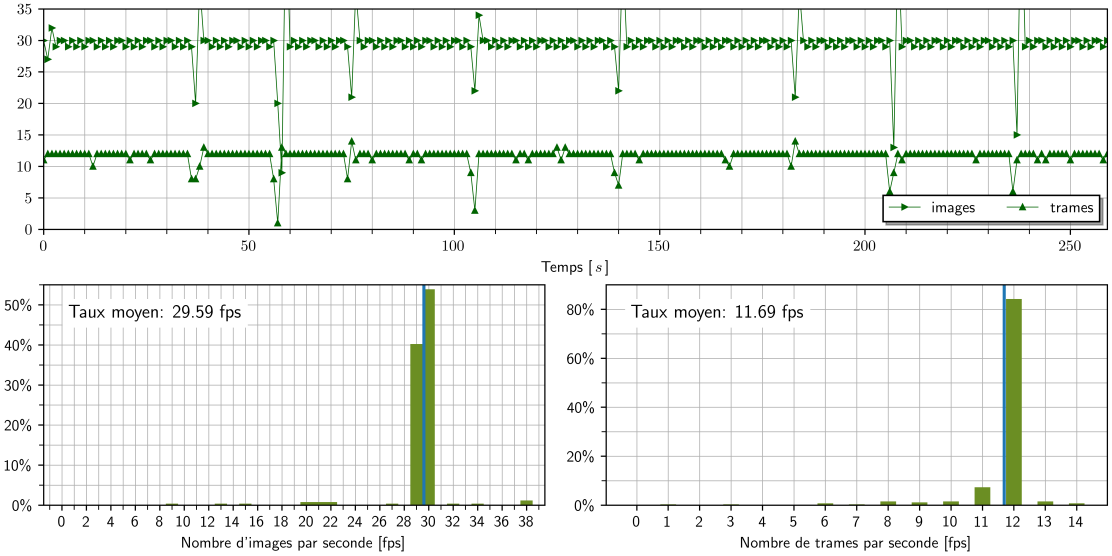

FIGURE 2.7 – Graphe de validation du système. Celui-ci atteint 29.59 fps et 11.69 fps pour les images et les trames, respectivement (moyenne sur la durée de l'enregistrement).

L'oscillation du nombre d'images est dû à la méthode de calcul, laquelle échantillonne ce dernier chaque seconde. Le taux de 29.59 fps entraîne donc 29 et 30 en alternance. On remarque par ailleurs quelques irrégularités . Ceci est dû au fonctionnement interne du Raspberry Pi et plus précisément à sa gestion de la mémoire. Cette précision est appuyée par la corrélation entre les sursauts des trames avec ceux des images sur la figure ci-dessus, et démontré par la vérification montrée à la FIG. 2.8 pour laquelle l'enregistrement a été désactivé. Pour jouir d'un enregistrement stable, l'utilisateur doit choisir adéquatement le micro-contrôleur pour éviter ces brefs ralentissements. Ce travail repose sur le Raspberry Pi, choisi avant ce constat et qui fournit malgré tout de bons résultats.

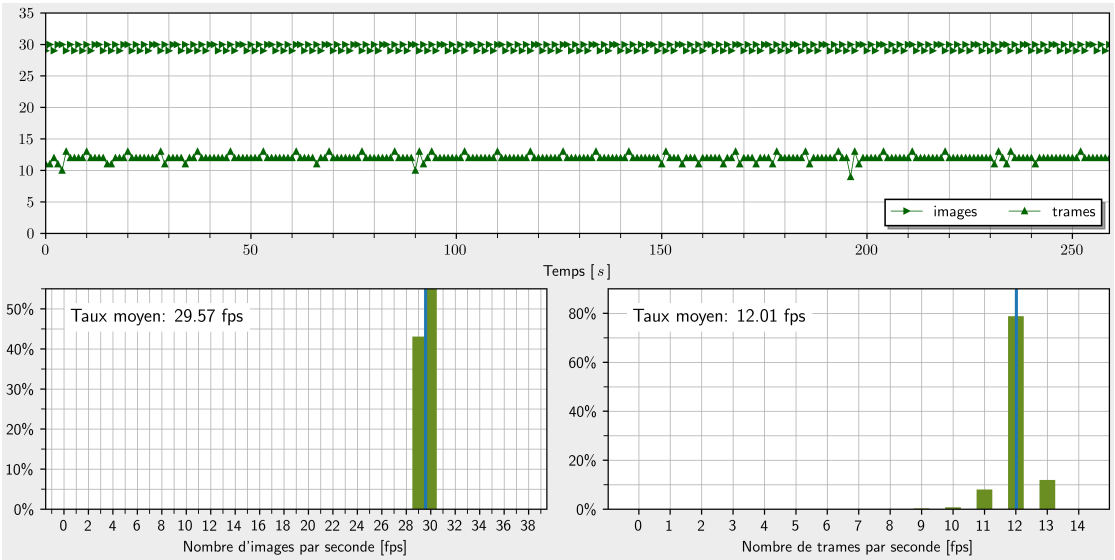

FIGURE 2.8 – Essai de validation sans enregistrement. Il permet de s'assurer que les quelques irrégularités constatées lors de la prise de vue sont bien dues à la gestion de la mémoire. Ce graphique valide du même coup le bon fonctionnement du système de récupération des flux et d'insertion des estampilles, qui peut se féliciter d'atteindre 12 trames et près de 30 images par seconde.

Chapitre 3

Radar et Traitement des Trames

Ce chapitre est dédié à la compréhension du fonctionnement du radar, et plus particulièrement la manière dont sont dérivés les graphiques distance–vitesse (généralement appelés graphes Doppler). Après une brève introduction aux concepts fondamentaux tels que la représentation des signaux en bande de base, l'effet Doppler ou le dilemme du même nom, le lecteur découvrira pourquoi l'application d'une FFT sur les mesures reçues du radar permet la génération des graphes. Les distance et vitesse maximales détectables sont ensuite discutées, desquelles sont déduits les paramètres internes choisis pour le radar. L'impact du remplissage par zéros est finalement abordé.

3.1 Introduction aux radars

3.1.1 Principe de fonctionnement

Les radars envoient des ondes électromagnétiques au moyen d'une antenne émettrice, à l'instar des LIDARs qui utilisent la lumière (ondes électromagnétiques appartenant au spectre visible). Ces ondes se déplacent en ligne droite à la vitesse de la lumière, soit $299\,792\,458\ \mathrm{m \cdot s^{-1}}$ (habituellement arrondi à $3 \cdot 10^8\ \mathrm{m \cdot s^{-1}}$) [1]. Lorsqu'elles rencontrent un obstacle, elles sont partiellement absorbées. Le reste subit une réflexion, en général dans plusieurs directions différentes. Une partie des ondes réfléchies repart ainsi vers l'antenne réceptrice, où elles sont captées et traitées.

1. Il faut en théorie tenir compte du milieu dans lequel évolue l'onde, puisque la vitesse c dépend de la permittivité ϵ et de la perméabilité μ de ce dernier : $c = 1/\sqrt{\epsilon\mu}$. En pratique, on considère que la vitesse dans l'air est égale à celle dans le vide, la variation n'étant que de quelques centièmes de pourcents.

La FIG. 3.1 illustre le fonctionnement d'un radar. De la puissance P_{T} à l'émetteur, S_* atteint l'obstacle par unité de surface, qui agit lui-même comme une source de puissance $P_{\mathrm{T}}^{\mathrm{equ.}}$ lors de la réflexion. La puissance P_{R} atteint finalement le récepteur. On modélise généralement le rapport entre S_* et $P_{\mathrm{T}}^{\mathrm{equ.}}$ par σ, appelé SER (surface équivalente radar), en mètres-carrés.

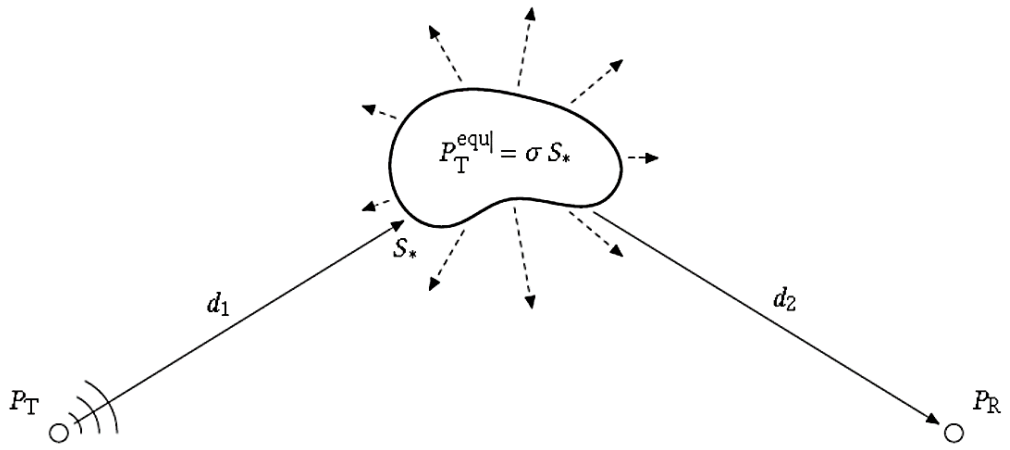

FIGURE 3.1 – Illustration du principe de fonctionnement d'un radar. L'onde est émise par l'émetteur T, à gauche. Elle rencontre un obstacle qui l'absorbe partiellement. Ce qui n'est pas absorbé est réfléchi dans toutes les directions, notamment vers le récepteur R, à droite. Extrait de [42].

3.1.2 Signaux en bande de base et bande passante

Tout signal $h(t)$ dont le spectre est étroit, c'est-à-dire qui s'étend sur un support fini et ne comprenant pas 0 [2], peut être décomposé selon ses composantes de Rice en phase $i(t)$ et en quadrature $q(t)$:

$$\begin{aligned} h(t) &= i(t)\cos(\omega t) - q(t)\sin(\omega t) \\ &= \mathfrak{Re}\left[(i(t) + jq(t))\, e^{j\omega t}\right] = \mathfrak{Re}\left[e(t)e^{j\omega t}\right] \end{aligned}$$

où l'enveloppe complexe $e(t)$ est introduite. Ce formalisme concis est surtout plus pratique, car il simplifie la représentation et la manipulation des signaux. Addition, soustraction, multiplication, déphasage, convolution sont autant d'opérations qui peuvent être ainsi réalisées sans tenir compte de la fréquence porteuse $f = \omega/2\pi$,

2. Cette condition est réalisée pour la plupart des signaux rencontrés, spécialement en télécommunications, car une composante à la fréquence nulle – donc constante – ne transmet pas d'information et induit une augmentation importante de la puissance consommée.

qui est souvent très élevée. Les ordinateurs sont les premiers à en bénéficier, car ils ne peuvent considérer que des signaux échantillonnés (le nombre d'échantillons étant directement proportionnel à la fréquence maximale du signal en raison du théorème de Nyquist).

Par exemple, dans le cas d'une impulsion non-modulée, on peut écrire :

$$x(t) = A \cdot \cos(\omega t + \phi) = \mathfrak{Re}\left[A \cdot e^{j\phi} e^{j\omega t}\right]$$

Et si on tient compte d'un train d'impulsions et non d'une seule :

$$x(t) = \sum_{i=-\infty}^{\infty} A \cdot \cos(\omega t + \phi)\,\mathrm{rect}_i = \mathfrak{Re}\left[\sum_{i=-\infty}^{\infty} A \cdot e^{j\phi}\,\mathrm{rect}_i\, e^{j\omega t}\right] \tag{3.1}$$

qui permet de considérer $e(t) = \sum_{i=-\infty}^{\infty} A \cdot e^{j\phi}\,\mathrm{rect}_i$. Ce résultat est utile dans la section 3.1.5. On observe que la phase ϕ apparaît sous la forme $e^{j\phi}$.

Sans en dire davantage à ce sujet, mentionnons qu'il est facile (terme discutable) de retrouver l'enveloppe complexe $e(t)$, autrement dit $i(t)$ et $q(t)$, d'un signal reçu grâce à un démodulateur cohérent. On peut ainsi considérer soit le signal $x(t)$ (bande passante), soit son enveloppe $e(t)$ (bande de base).

3.1.3 Estimation de la distance

La distance R séparant le radar d'un objet peut être estimée à partir du délai τ entre l'émission d'une impulsion et la réception de l'onde reflétée :

$$\tau = \frac{2R}{c} \qquad \Longrightarrow \qquad R = \frac{c\tau}{2} \tag{3.2}$$

où c représente la vitesse de la lumière, soit $3 \cdot 10^8\ \mathrm{m \cdot s^{-1}}$. L'onde parcourt un aller-retour, donc une distance de $2R$. Les radars estiment le temps séparant l'émission et la réception, et en déduisent la distance selon (3.2).

Cependant, plus la portée du radar est grande, plus la période T_{PRF} entre deux impulsions doit l'être aussi. Il ne serait sinon pas possible de déterminer si le signal reçu correspond à l'écho d'une impulsion k ou à celui de l'impulsion $k+1$, et on ne saurait par conséquent pas différencier un objet lointain d'un objet proche. C'est pourquoi

$$T_{\mathrm{PRF,min}} = \tau_{\max} = \frac{2R_{\max}}{c}$$

Le taux d'actualisation des données correspond à la fréquence des impulsions f_{PRF} et est inversement proportionnel à $T_{\mathrm{PRF,min}}$.

3.1.4 Estimation de la vitesse et effet Doppler

L'effet Doppler est un phénomène bien connu qui explique pourquoi la fréquence d'un objet en approche paraît plus aiguë que lorsqu'il s'éloigne.

Soit l'impulsion émise $x(t) = A \cdot \cos(\omega t + \phi) = A \cdot \cos(\theta(t))$. Lorsque la cible est en mouvement, la distance R est une fonction du temps, soit $R = R(t)$. On a aussi $v = \mathrm{d}R(t)/\mathrm{d}t$. En relativité Galiléenne, le signal reçu est de la forme

$$\begin{aligned} r(t) &= A \cdot \cos(\omega(t-\tau)+\phi) = A \cdot \cos\left(\omega\left(t - \frac{2R(t)}{c}\right) + \phi\right) \\ &= A \cdot \cos\left(\omega t - \frac{2\omega}{c}R(t) + \phi\right) = A \cdot \cos\left(\omega t + \tilde{\phi}(t)\right) = A \cdot \cos\left(\tilde{\theta}(t)\right) \end{aligned}$$

où $\tilde{\phi}(t) = -2\omega/c\ R(t) + \phi$ et $\tilde{\theta} = \omega t + \tilde{\phi}(t)$.

On retrouve la fréquence Doppler f_D telle que

$$f_\mathrm{D} = \frac{1}{2\pi}\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}t} = -\frac{1}{2\pi}\frac{2 \cdot 2\pi f}{c}\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}t} = -\frac{2f}{c}v = -\frac{2v}{\lambda} \tag{3.3}$$

où $\lambda = c/f$ est appelée longueur d'onde car c'est la distance parcourue par l'onde durant une période. À nouveau, la fréquence instantanée de $r(t)$ vaut

$$\tilde{f} = \frac{1}{2\pi}\frac{\mathrm{d}\tilde{\theta}}{\mathrm{d}t} = f + f_\mathrm{D}$$

et on constate que lorsque l'objet s'éloigne du radar ($v > 0$, $f_\mathrm{D} < 0$), la fréquence perçue $\tilde{f}$ décroît, tandis qu'elle croît dans le cas contraire ($v < 0$, $f_\mathrm{D} > 0$).

3.1.5 Dilemme Doppler

Décrite par (3.1), l'enveloppe complexe d'un train d'impulsions s'écrit

$$e(t) = \sum_{i=-\infty}^{\infty} A \cdot e^{j\phi}\,\mathrm{rect}_i$$

ce qui est en réalité la convolution d'une fenêtre rectangulaire avec un peigne de Dirac $p(t)$ (train de deltas de Dirac). Le spectre $E(\omega)$ de $e(t)$ est donc égal au produit de $\mathrm{Rect}(\omega)$ et $P(\omega)$ ($A \cdot e^{j\phi}$ n'a pas d'importance et est ignoré par la suite). Avec $\mathrm{sinc}(x) = \sin(x)/x$, $\mathrm{Rect}(\omega)$ et $P(\omega)$ valent :

$$\mathrm{Rect}(\omega) = T_\mathrm{imp}\,\mathrm{sinc}\left(\frac{\omega T_\mathrm{imp}}{2}\right) \qquad P(\omega) = \frac{2\pi}{T_\mathrm{PRF}}\sum_{m=-\infty}^{+\infty}\delta\left(\omega - \frac{2\pi m}{T_\mathrm{PRF}}\right)$$

Le produit $E(\omega) = \mathrm{Rect}(\omega)P(\omega)$ ressemble à celui illustré à la FIG. 3.2.

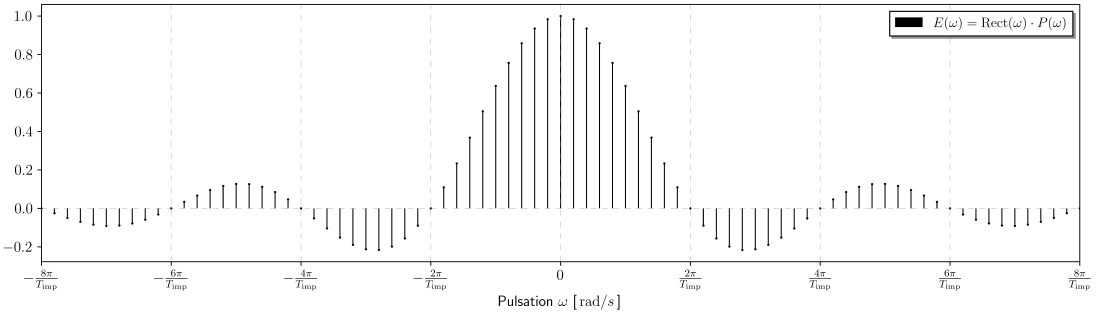

FIGURE 3.2 – Spectre de $E(\omega)$, résultat du produit de $\mathrm{Rect}(\omega)$, dû à la forme rectangulaire des éléments qui constituent l'enveloppe complexe, et de $P(\omega)$, dû à leur répétition périodique. L'espace entre deux raies vaut f_{PRF}.

Le signal reçu aurait alors la même enveloppe, à la différence près qu'elle aurait subi un décalage de f_{D} sur l'axe des fréquences. Pour déterminer f_{D}, les radars estiment (il faut se contenter d'une estimation, puisque le signal reçu est toujours bruité) la position des raies et la compare avec le spectre initial. On constate que dès lors, si la fréquence Doppler f_{D} dépasse l'écart entre deux raies (qui vaut $f_{\mathrm{PRF}} = 1/T_{\mathrm{PRF}}$), on est incapable de discerner cet important décalage d'un plus petit. En d'autres termes, tout décalage $\Delta f = k \cdot f_{\mathrm{PRF}} + f_0$ où $k \in \mathbb{Z}$ est interprété comme $\Delta f = f_0$. C'est pourquoi $|f_{\mathrm{D,max}}| = f_{\mathrm{PRF}}$. Le signe de la vitesse n'étant en outre pas connu, cet intervalle sert tant aux vitesses négatives que positives, d'où l'insertion d'un facteur 1/2. La vitesse maximale détectable vaut finalement

$$|v_{\mathrm{max}}| = \frac{1}{2}\frac{cf_{\mathrm{PRF}}}{2f} = \frac{cf_{\mathrm{PRF}}}{4f} \tag{3.4}$$

En se souvenant de (3.2), il apparaît que

$$R_{\mathrm{max}} \cdot |v_{\mathrm{max}}| = \frac{c}{2f_{\mathrm{PRF}}}\frac{cf_{\mathrm{PRF}}}{4f} = \frac{c^2}{8f}$$

Le produit $R_{\mathrm{max}} \cdot v_{\mathrm{max}}$ est indépendant de f_{PRF} : tout autre paramètre restant égal, l'augmentation de la portée par modification de f_{PRF} implique inévitablement la diminution de la vitesse maximale détectable, et réciproquement. C'est ce qu'on appelle le dilemme Doppler.

3.2 Radar FMCW

Comme énoncé précédemment, le radar utilisé dans ce travail est un radar FMCW (*frequency modulated continuous wave*). Il émet en permanence. En réalité, son signal est composé de blocs durant lesquels la fréquence augmente linéairement. Entre ces blocs, elle est constante, formant un profil en «dents de scie» pour la fréquence instantanée. Ce signal est illustré à la FIG. 3.3.

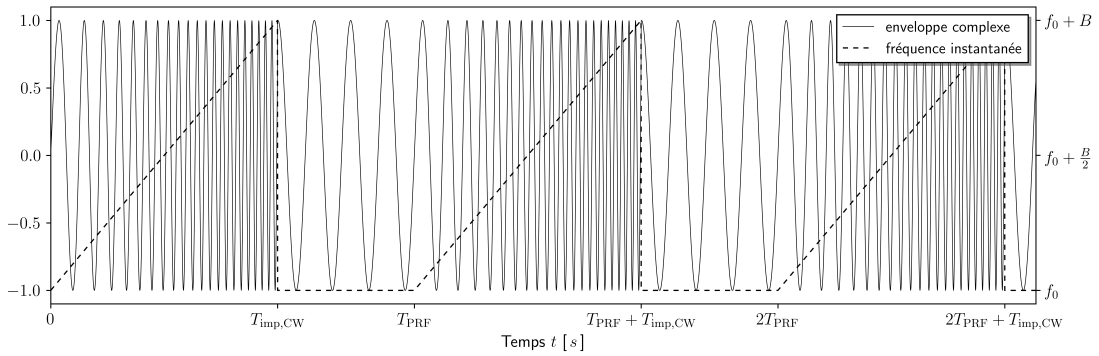

FIGURE 3.3 – Enveloppe complexe du signal émis par le radar FMCW, ainsi que sa fréquence instantanée. On y distingue les «dents de scie» (ou rampes). On définit la bande passante B comme étant l'intervalle parcouru par la fréquence instantanée f.

3.2.1 Modulation et formes d'ondes

Le radar module la porteuse en fréquence avec, à chaque impulsion,

$$f(t) = f_\text{c} + f_0 + \frac{B}{T_\text{imp}} t = f_\text{c} + f_0 + \mu t$$

où f_c est la fréquence de la porteuse (très élevée), $\mu = B/T_\text{imp}$ et $t \in [0, T_\text{imp}]$. Le signal modulé ressemble à

$$\begin{aligned} s(t) &= A \cdot \cos\left(2\pi \int_0^t f(t)\mathrm{d}t + \phi\right) = A \cdot \cos\left(2\pi \int_0^t f_\text{c} + f_0 + \mu t \mathrm{d}t + \phi\right) \\ &= A \cdot \cos\left(2\pi \left(f_\text{c} t + f_0 t + \frac{\mu}{2} t^2\right) + \phi\right) = \mathfrak{Re}\left[A \cdot e^{j\left(2\pi\left(f_\text{c} t + f_0 t + \frac{\mu}{2} t^2\right) + \phi\right)}\right] \\ &= \mathfrak{Re}\left[A \cdot e^{j\left(2\pi\left(f_0 t + \frac{\mu}{2} t^2\right) + \phi\right)} e^{j2\pi f_\text{c} t}\right] = \mathfrak{Re}\left[e_\text{s}(t)\, e^{j2\pi f_\text{c} t}\right] \end{aligned}$$

où l'enveloppe complexe $e_\text{s}(t)$ de $s(t)$ est introduite. Attention, on considère désormais toujours l'enveloppe complexe des signaux. Par exemple, $s(t)$ signifie $e_\text{s}(t)$.

Le signal émis $x(t)$ se compose de L rampes : $x(t) = \sum_{l=0}^{L-1} s\left(t - lT_\text{PRF}\right)$.

3.2.1.1 Signal reçu

Supposons que le radar dispose de K antennes réceptrices et observe M cibles. Le signal reçu $r(t)$ pour une impulsion est alors

$$r_{l,k}(t) = \sum_{m=0}^{M-1} \underbrace{a_m}_{(1)} \underbrace{s\left(t-\tau_m\right)}_{(2)} \underbrace{e^{j2\pi f_{\mathrm{D}} T_{\mathrm{PRF}} \cdot l}}_{(3)} \underbrace{e^{j\frac{2\pi}{\lambda} d_{\mathrm{s}} \sin(\theta_m) \cdot k}}_{(4)} + n(t) \tag{3.5}$$

où $n(t)$ est un bruit blanc gaussien additif qui est ignoré.

Objet et environnement

Le terme (1) est lié aux pertes dues au trajet de l'onde dans l'air, mais aussi au coefficient de réflexion de l'onde sur l'objet. Il dépend par conséquent de la SER définie précédemment. Mathématiquement, il applique une atténuation (terme réel) et un déphasage (terme complexe). Il est par conséquent complexe, d'amplitude strictement inférieure à 1 et a priori différent pour chaque cible m.

Time of arrival (TOA)

L'origine du terme (2) est évidente : le signal reçu correspond à celui émis décalé dans le temps puisque l'onde a dû parcourir deux fois la distance qui sépare le radar de la cible m. On peut par ailleurs développer (2) en

$$\begin{aligned} s\left(t-\tau_m\right) &= A \cdot e^{j\left(2\pi\left(f_0(t-\tau_m)+\frac{\mu}{2}(t-\tau_m)^2\right)+\phi\right)} \\ &= \underbrace{A \cdot e^{j\left(2\pi\left(f_0 t+\frac{\mu}{2}t^2\right)+\phi\right)}}_{s(t)} \underbrace{e^{-j2\pi\left(f_0\tau_m-\frac{\mu}{2}\tau_m^2\right)}}_{\text{ne dépend pas de } t} \underbrace{e^{-j2\pi\mu\tau_m t}}_{\text{dépend de } t} \\ &= s(t)\, e^{-j2\pi\left(f_0\tau_m-\frac{\mu}{2}\tau_m^2\right)}\, e^{-j2\pi\mu\tau_m t} \end{aligned} \tag{3.6}$$

Effet Doppler

On considère par hypothèse que les impulsions sont courtes : l'effet Doppler au sein d'une rampe est négligé. Il impacte en revanche significativement la phase des échos reçus d'une rampe à l'autre. Le déphasage entre deux d'entre elles vaut

$$\Delta\phi = \frac{2\pi}{\lambda} 2R = \frac{2\pi}{\lambda} 2vT_{\mathrm{PRF}} = 2\pi f_{\mathrm{D}} T_{\mathrm{PRF}}$$

où $f_{\mathrm{D}} = 2v/\lambda$, en adéquation avec (3.3). Ce dernier doit être multiplié par l pour la l^{e} rampe. On obtient le terme (3).

Direction of arrival (DOA)

Puisque toutes les antennes ne sont pas au-même endroit, la distance parcourue par l'onde pour leur parvenir n'est pas la même, comme décrit sur la FIG. 3.4. La cible est lointaine ($R \gg d_s$) : les rayons incidents sont (presque) parallèles. Le déphasage induit par la distance supplémentaire pour rejoindre A vaut

$$\Delta\phi = \frac{2\pi}{\lambda} \cdot \Delta = \frac{2\pi}{\lambda} \cdot d_\mathrm{s} \sin(\theta_m)$$

Ce dernier doit être multiplié par k pour la k^e antenne, formant le terme (4).

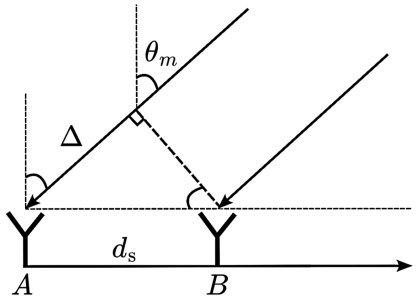

FIGURE 3.4 – Illustration d'un réseau d'antennes à 2 dimensions. Adapté de [43].

3.2.1.2 Échantillonnage

Le signal reçu $r_{l,k}(t)$ est multiplié par $s^*(t)$, qui est le signal conjugué de celui émis. On obtient ainsi, en intégrant (3.5) et (3.6) :

$$\begin{aligned} y_{l,k}(t) &= r_{l,k}(t) \cdot s^*(t) \\ &= \sum_{m=0}^{M-1} a_m \, s(t) \, e^{-j2\pi\left(f_0\tau_m - \frac{\mu}{2}\tau_m^2\right)} \, e^{-j2\pi\mu\tau_m t} \, e^{j2\pi f_\mathrm{D} T_\mathrm{PRF} \cdot l} \, e^{j\frac{2\pi}{\lambda} d_\mathrm{s} \sin(\theta_m) \cdot k} \, s^*(t) \\ &= \sum_{m=0}^{M-1} \tilde{a}_m \, e^{-j2\pi f_{\tau,m} t} \, e^{j2\pi f_\mathrm{D} T_\mathrm{PRF} \cdot l} \, e^{j2\pi f_{\vartheta,m} \cdot k} \end{aligned}$$

où $\tilde{a}_m = a_m \, e^{-j2\pi\left(f_0\tau_m - \frac{\mu}{2}\tau_m^2\right)} A^2$, $f_{\tau,m} = \mu\tau_m$ et $f_{\vartheta,m} = d_\mathrm{s} \sin(\theta_m)/\lambda$. Il faut pour cela remarquer que $s(t)s^*(t) = A\,A^* = |A|^2 = A^2$ puisque $A \in \mathbb{R}$. Ce signal est finalement échantillonné à la fréquence $f_\mathrm{s} = 1/T_\mathrm{s}$:

$$y_{l,k}[n] = \sum_{m=0}^{M-1} \tilde{a}_m \, e^{-j2\pi f_{\tau,m} T_\mathrm{s} \cdot \textcircled{n}} \, e^{j2\pi f_\mathrm{D} T_\mathrm{PRF} \cdot \textcircled{l}} \, e^{j2\pi f_{\vartheta,m} \cdot \textcircled{k}} \tag{3.7}$$

Cette expression est importante, puisqu'elle résume à elle seule le fonctionnement du radar FMCW. Elle constitue le point de départ du traitement appliqué aux données transférées par le radar, regroupées en trames comme expliqué ci-dessous.

Contenu des trames

Le radar envoie les échantillons $y_{l,k}[n]$ au moyen de trames TCP. Il y a 3 antennes réceptrices, chaque trame inclut 256 rampes et le signal est échantillonné

en 256 points par rampe : $k = \{0, \ldots, 2\}$, $l = \{0, \ldots, 255\}$ et $n = \{0, \ldots, 255\}$. Les coefficients $y_{l,k}[n]$ sont complexes et leurs parties réelle et imaginaire sont encodées sur 2 bytes. Les trames comprennent en conséquence $3 \cdot 256 \cdot 256 \cdot 2 \cdot 2 = 786\,432$ bytes au total, comme indiqué par la fiche technique.

Spectre micro-Doppler

On s'apercevra que les pics des graphes Doppler sont pour la plupart diffus : ceux-ci sont sujets à un étalement autour d'une fréquence centrale. Ce phénomène, baptisé spectre micro-Doppler, s'explique par la différence de vitesse apparente entre les points d'un même véhicule. Ce cas de figure est facile à percevoir avec les roues : la vitesse au point de contact avec le sol est nulle tandis qu'elle vaut le double de celle du véhicule au point le plus haut. L'étude du spectre micro-Doppler a été menée dans [41] par Kevin De Sousa et Gauthier Rotsart de Hertaing. Le lecteur est invité à s'y référer.

3.2.2 Graphes Doppler

3.2.2.1 Fast Fourier Transform

Dans l'expression (3.7), les indices n, l, k sont mis en évidence : $y_{l,k}$ est une somme pondérée de fonctions oscillant dans les 3 directions de l'espace (n, l, k). On va néanmoins omettre la direction k pour laquelle le nombre d'échantillons est trop faible [3]. Pour déterminer $f_{\tau,m}$ et f_{D}, on cherche à connaître les fréquences représentées selon n et l, respectivement. Puisqu'il s'agit d'un signal échantillonné, il est question de la DFT (Discrete Fourier Transform), dont l'implémentation la plus efficace est appelée FFT (Fast Fourier Transform). Le nombre d'échantillons à sa sortie est égal au nombre en entrée, soit 256×256. On obtient des graphiques distance–vitesse, qu'on appelle généralement graphes Doppler et dont la FIG. 3.5 est un exemple réalisé expérimentalement.

Sans entrer dans les détails, l'échantillonnage d'un signal à la fréquence f_{s} est équivalent à le multiplier par un peigne de Dirac. Cela revient à convoluer son spectre par celui du peigne (sans oublier le facteur $1/2\pi$ sans importance), dont la déduction du spectre est détaillée à l'annexe A.2. Ceci implique que le spectre est rendu périodique, de période f_{s}. L'observation d'une seule période $[-f_{\mathrm{s}}/2, f_{\mathrm{s}}/2]$ ou $[0, f_{\mathrm{s}}]$ résume par conséquent l'ensemble du spectre. Les 256 échantillons sont répartis uniformément sur cet intervalle.

3. Les 3 antennes sont disposées en «L», ce qui donne 2 échantillons pour les axes horizontal et vertical : bien trop peu pour une détection de pics.

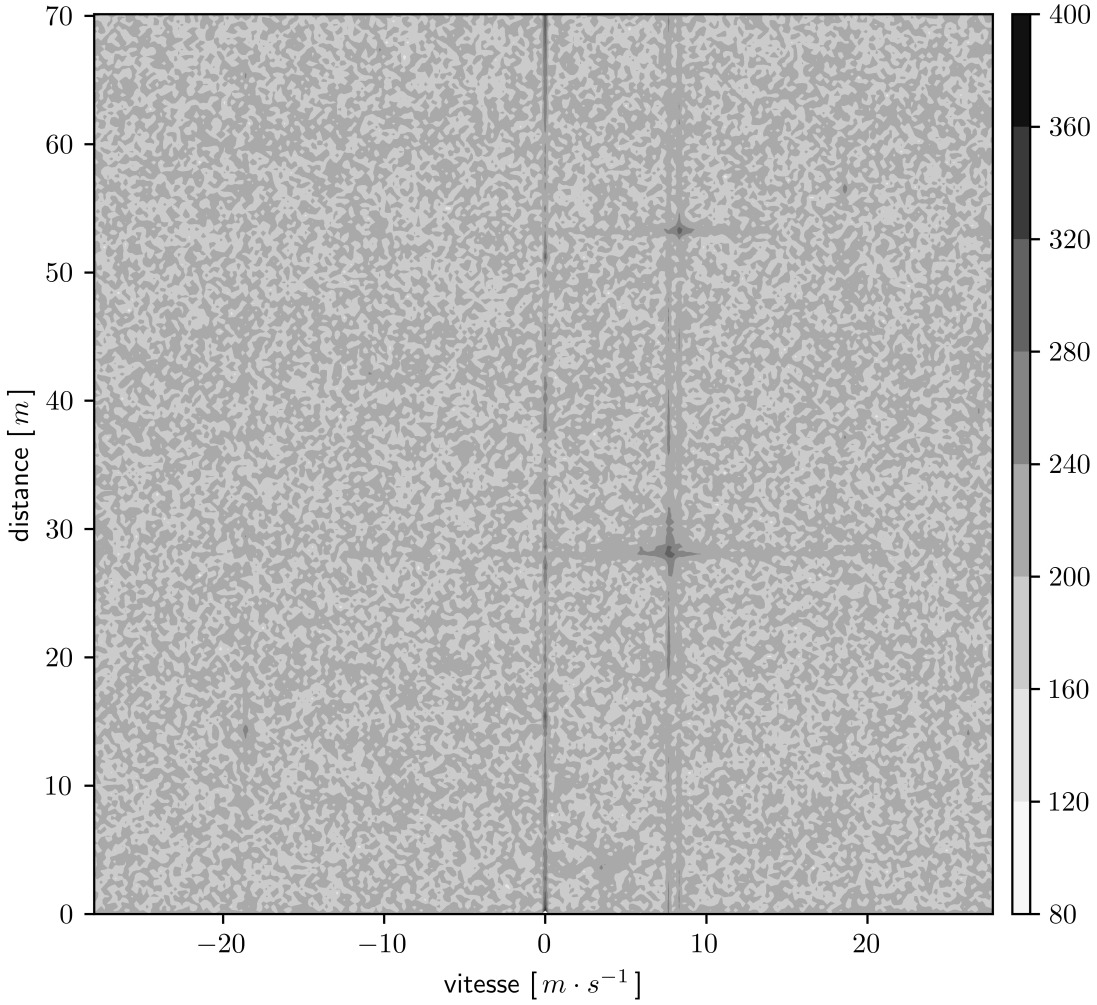

FIGURE 3.5 – Exemple de graphe Doppler (graphique distance–vitesse) obtenu après l'application de la FFT. On observe les 2 véhicules (aux alentours de 28 et 54 mètres respectivement), la ligne centrale (qui est une non-idéalité du radar) et 2 points fantômes à peine visibles (aux alentours de 15 et 55 mètres, aussi une non-idéalité du radar). Amplitude exprimée en décibels (dB).

Résolution en distance

En se rappelant de (3.2) et que $f_\tau = \mu\tau$, on sait que

$$\Delta R = \frac{c\Delta\tau}{2} = \frac{c}{2}\frac{\Delta f_\tau}{\mu} = \frac{c}{2}\frac{T_{\text{imp}}}{B}\frac{1}{T_{\text{enr}}} = \frac{c}{2B}\frac{T_{\text{imp}}}{T_{\text{enr}}} = \frac{c}{2B}\frac{N+M}{N}$$

où T_{enr} représente la période d'enregistrement. En effet, comme indiqué dans la fiche technique, le radar émet en réalité $N + M$ échantillons par rampe. Les M premiers échantillons sont mis de côté pour éliminer les discontinuités, de sorte que $T_{\text{imp}} = (N + M)/f_{\text{s}}$ et $T_{\text{enr}} = N/f_{\text{s}}$. Dans ce travail, $N = 256$ et $M = 4$.

Distance maximale détectable

Comme précisé précédemment, le spectre d'un signal échantillonné est périodique, de période f_{s} en fréquence (ou ω_{s} en pulsation). Cela signifie qu'un élément à la pulsation ω_τ se retrouvera en $\omega_\tau \pm i \cdot \omega_{\text{s}}$ avec $i \in \mathbb{Z}$, comme représenté sur la FIG. 3.6. Puisque le spectre est périodique, on doit considérer un certain intervalle et supposer que son contenu en provient initialement. Puisque la distance est forcément positive (le radar n'observe que devant lui), f_τ l'est aussi et l'intervalle de prospection le plus adéquat est $[0, \omega_{\text{s}}]$ [4].

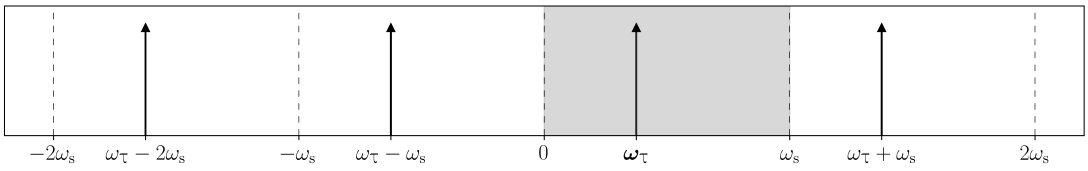

FIGURE 3.6 – Répétition du spectre initialement localisé en ω_τ. L'intervalle de prospection $[0, \omega_{\text{s}}]$ est choisi pour la distance.

On a ainsi $\omega_\tau \geq 0$, ce qui est se résout sans surprise en $R \geq 0$, et $\omega_\tau \leq \omega_{\text{s}}$ où $\omega_{\text{s}} = 2\pi N/T_{\text{enr}}$, formant

$$2\pi\omega_\tau = 2\pi\mu\tau = 2\pi\frac{B}{T_{\text{imp}}}\frac{2R}{c} \leq 2\pi\frac{N}{T_{\text{enr}}}$$

On trouve finalement

$$R \leq \frac{c}{2B}\frac{N+M}{N}N = \frac{c}{2B}(N+M)$$

4. Il est important de garder l'hypothèse à l'esprit lors de l'interprétation du graphe : si un véhicule est à une distance telle que la pulsation correspondante est $\omega_{\text{s}} + \Delta\omega$, la pulsation apparente est $\Delta\omega$ et l'estimation de la distance n'est pas correcte.

Ce résultat est bien égal à $N \cdot \Delta R$. Cependant, le contenu en ω_s est le même que celui en 0 par périodicité. Il faut donc considérer

$$R_\mathrm{max} = (N-1) \cdot \Delta R = \frac{c}{2B} \frac{N+M}{N} (N-1)$$

Résolution en vitesse

En employant (3.3), on trouve que

$$\Delta v = \frac{\lambda}{2} \frac{\Delta f}{T_\mathrm{PRF}} = \frac{\lambda}{2 T_\mathrm{PRF} N_\mathrm{r}}$$

où $\Delta f = 1/N_\mathrm{r} T_\mathrm{PRF}$ car la fréquence d'échantillonnage f_s est $1/T_\mathrm{PRF}$ dans ce cas, et N_r points sont obtenus sur l'intervalle de norme f_s.

Vitesse maximale détectable

Tout comme pour la distance, la vitesse maximale est contrainte par la fréquence d'échantillonnage. Néanmoins, contrairement à la première, le signe de la vitesse n'est pas connu et l'intervalle de prospection devient $[-f_\mathrm{s}/2, f_\mathrm{s}/2]$ [5]. La FIG. 3.7 illustre la situation.

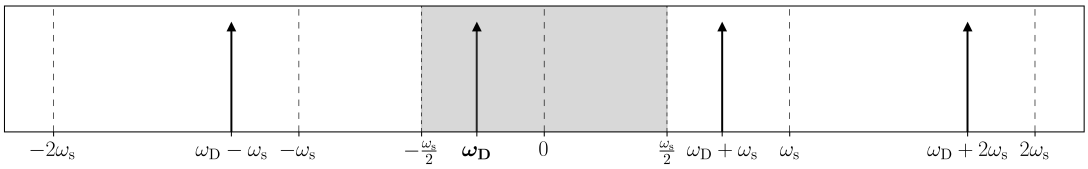

FIGURE 3.7 – Répétition du spectre initialement localisé en ω_D. L'intervalle de prospection $[-\omega_\mathrm{s}/2, \omega_\mathrm{s}/2]$ est choisi pour la vitesse.

Cette fois-ci, $|\omega_\mathrm{D}| \leq \omega_\mathrm{s}/2$ où $\omega_\mathrm{s} = 1/T_\mathrm{PRF}$:

$$|\omega_\mathrm{D}| = \left| 2\pi \frac{2}{\lambda} v \right| \leq \frac{1}{2} 2\pi \frac{1}{T_\mathrm{PRF}} \qquad\qquad |v_\mathrm{max}| = \frac{\lambda}{4 T_\mathrm{PRF}}$$

On retrouve l'expression (3.4) dérivée en 3.1.5, ainsi que $|v_\mathrm{max}| = N_\mathrm{r}/2 \cdot \Delta v$. En tenant compte du fait que le contenu en $-\omega_\mathrm{s}/2$ est identique à celui en $\omega_\mathrm{s}/2$, on obtient

$$|v_\mathrm{max}| = \left(\frac{N_\mathrm{r}}{2} - 1 \right) \cdot \Delta v = \left(\frac{N_\mathrm{r}}{2} - 1 \right) \frac{\lambda}{2 T_\mathrm{PRF} N_\mathrm{r}}$$

5. Dans une rue à sens unique ou tout autre contexte dans lequel le signe de la vitesse est connu, l'intervalle investigué peut être décalé sur $[0, \omega_\mathrm{s}]$, doublant la vitesse maximale détectable.

En tenant compte du délai entre les rampes, de la fréquence d'horloge f_{clk} du radar et du nombre de périodes d'horloge par échantillon S_{clk}, on calcule

$$T_{\text{PRF}} = \frac{S_{\text{clk}}\,(N+M) + \text{delay}}{f_{\text{clk}}}$$

Le lecteur attentif constatera que cela concorde précisément avec les relations fournies dans la fiche technique.

3.2.2.2 Choix des paramètres

Une bande passante totale B de 554 MHz et un délai de 1147 périodes d'horloge ont été retenus. Ils permettent d'atteindre une résolution en distance

$$\Delta R = \frac{3 \cdot 10^8}{2 \cdot 554 \cdot 10^6} \frac{256+4}{256} = 0.275 \text{ m}$$

une portée maximale

$$R_{\max} = (256-1) \cdot 0.275 = 70.125 \text{ m}$$

une résolution en vitesse

$$\Delta v = \frac{38461538 \cdot 0.012426}{2 \cdot 256 \cdot (12 \cdot (256+4) + 1147)} = 0.219 \text{ m} \cdot \text{s}^{-1} = 0.788 \text{ km} \cdot \text{h}^{-1}$$

et une vitesse maximale (en valeur absolue)

$$v_{\max} = \left(\frac{256}{2} - 1\right) \cdot 0.219 = 27.813 \text{ m} \cdot \text{s}^{-1} = 100.125 \text{ km} \cdot \text{h}^{-1}$$

qui satisfont les contraintes annoncées dans le chapitre 1. La valeur de la bande passante et du délai ont été enregistrées dans la partition inscriptible `/mnt/` de la mémoire du radar directement (dans le fichier `/mnt/sensor_server.conf`) via une connexion Telnet.

3.2.2.3 Détermination des angles

La FFT ne permet pas une mesure précise de la position des cibles à cause du faible nombre d'antennes : K antennes sur un axe assurent tout au plus le calcul de $K-1$ angles. Dans notre cas, puisqu'il y a $K=2$ antennes dans chaque direction, le radar ne pourrait traiter qu'un seul véhicule. On peut en revanche mettre de côté la méthode analytique de la FFT au profit d'une recherche exhaustive.

L'équation (3.7) indique que l'emplacement de l'antenne impacte la phase du signal reçu. En s'inspirant du terme (4) de (3.5), on définit $e^{j\phi_k} = e^{j\frac{2\pi}{\lambda}\delta_k}$ où δ_k correspond à la différence de distance entre la cible et l'antenne $k = 0, 2$ par rapport à l'antenne 1 prise comme référence. C'est une extension du terme DOA,

exposé en 3.2.1.1, à un repère à 3 dimensions. Celui utilisé est affiché à la FIG. 3.8. Les angles θ, η qui y figurent laissent ensuite place à $\gamma = \sin(\theta)\sin(\eta)$, $\xi = \cos(\theta)$ afin d'alléger les notations. L'annexe A.3 justifie ce changement, vérifiant par la même occasion le résultat suivant :

$$\phi_0 = \frac{2\pi}{\lambda}\delta_0 = -\frac{2\pi}{\lambda}\xi\Delta y$$
$$\phi_2 = \frac{2\pi}{\lambda}\delta_2 = -\frac{2\pi}{\lambda}\gamma\Delta x$$

où $\Delta x = 0.022$ m et $\Delta y = 0.040$ m représentent les espacements horizontal et vertical entre les antennes. Pour évaluer les coordonnées γ, ξ (desquelles seront dérivés les angles θ, η), un grand nombre de possibilités $\gamma, \xi \in [-1, 1]$ sont vérifiées, en évaluant

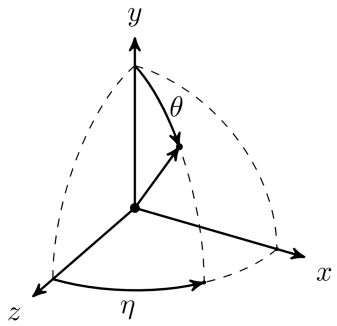

FIGURE 3.8 – Repère choisi pour la localisation des objets dans l'espace. θ est appelé angle d'élévation, tandis que η désigne l'angle azimutal.

$$\kappa = a_0 \cdot e^{j\frac{2\pi}{\lambda}\xi\Delta y} + a_1 + a_2 \cdot e^{j\frac{2\pi}{\lambda}\gamma\Delta x}$$

où chaque valeur complexe $a_k = \tilde{a}_m\, e^{j\frac{2\pi}{\lambda}\delta_k}$ est issue du graphe Doppler associé à l'antenne k au point d'attention.

Lorsque des nombres complexes sont de norme fixe, celle de leur somme est maximale s'ils sont alignés ; ce qui signifie qu'il sont de même phase. En d'autres termes, trouver $\hat{\gamma}$ et $\hat{\xi}$ tels que $\hat{\kappa}(\hat{\gamma}, \hat{\xi}) = \operatorname{argmax}(|\kappa|)$ garantit que les déphasages ϕ_k sont annulés, et donc connus. Cette démarche souffre malheureusement d'un phénomène d'ambiguïté. En effet, puisque $e^{j\phi} = e^{j(\phi+2\pi)}$, il apparaît que $e^{j2\pi/\lambda\gamma\Delta x} = e^{j2\pi/\lambda\Delta x(\gamma+\lambda/\Delta x)}$: γ est périodique de période $\lambda/\Delta x$ et il existera pour cette raison $\lfloor 2\Delta x/\lambda \rfloor$ ambiguïtés pour γ. Le même raisonnement tient bien sûr pour ξ. La FIG. 3.9 montre la norme de κ pour $\hat{\gamma}$, $\hat{\xi}$ balayant $[-1, 1]^2$.

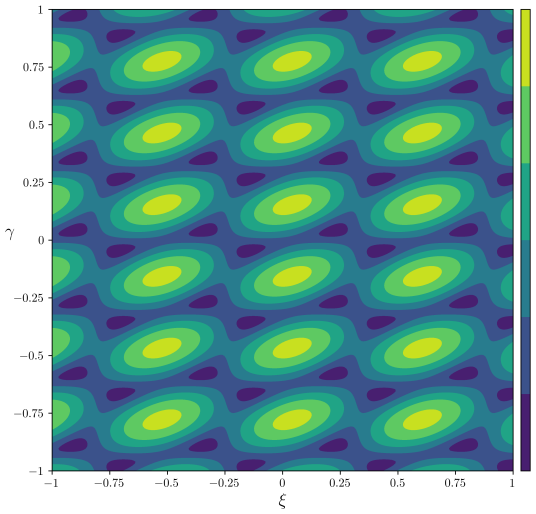

FIGURE 3.9 – Représentation de $|\kappa(\gamma, \xi)|$ pour $\gamma, \xi \in [-1, 1]$. Les ambiguïtés sont visibles et le lecteur peut y vérifier leur nombre, avec $\lambda = 0.0124$ m.

À défaut d'information supplémentaire, a priori (par l'étude du contexte) ou provenant de la caméra, il n'est pas possible de savoir quel angle choisir. C'est pourquoi, pour générer le set de données au chapitre 5, il a été décidé de fournir les valeurs $\hat{\gamma} \in [0, \lambda/\Delta x]$ et $\hat{\xi} \in [0, \lambda/\Delta y]$. L'opérateur doit garder cette incertitude à l'esprit et exploiter l'information que lui fournit son analyse des trajectoires pour valider les angles ou les adapter en prenant $\hat{\gamma} + i \cdot \lambda/\Delta x$ et $\hat{\xi} + j \cdot \lambda/\Delta y$ $(i, j \in \mathbb{Z})$. Les valeurs numériques des ambiguïtés lui seront transmises à cette effet. Une fois les ambiguïtés levées, il pourra appliquer $\theta = \arccos(\xi)$ et $\eta = \arctan(\gamma/\sqrt{1-\gamma^2-\xi^2})$.

3.2.3 Zero padding

Le spectre d'une séquence est continu en la variable Ω appelée pulsation normalisée. Puisqu'il est périodique de période 2π, on n'en représente qu'une seule période, habituellement $[-\pi, \pi]$ ou $[0, 2\pi]$. De ce spectre inconnu, la FFT permet d'obtenir N échantillons. Puisqu'on ne connaît le spectre de la séquence qu'en N points, la précision de l'estimation la position des maxima est limitée. Certains objets sont même partiellement (voire totalement) invisibles sur le graphe. Une manière bien connue d'améliorer la résolution d'une FFT consiste à augmenter artificiellement la taille du signal par l'ajout de zéros dans chaque direction (2 dans notre cas) avant de la réaliser.

En multipliant la dimension N pour atteindre $N' = \beta N$, où $\beta = 3$ par exemple, on accède au contenu fréquentiel localisé tous les $2\pi/N' = 1/3 \cdot 2\pi/N$: 3 fois plus de points sont connus. L'estimation des maxima du spectre est donc meilleure et certains objets, qui n'étaient pas détectés auparavant, ne passent plus inaperçus. Le *zero padding* ou «remplissage par zéros» est ainsi un moyen de rendre visibles des objets dissimulés ! En revanche, cela augmente significativement le temps d'exécution de la FFT, dont la complexité est $\mathcal{O}(N \log(N))$. Il ne faut donc pas choisir le facteur β trop élevé, au risque de rendre l'exécution déraisonnablement longue. L'outil laisse à l'utilisateur le choix de β, qui est pris à 1 dans ce travail (sauf mention contraire).

Le remplissage par zéro entraîne de surcroît un autre changement important : il accentue l'effet d'oscillation autour des points du graphe. Cet effet, qui n'est pas très marqué lorsque $\beta = 1$, est dû à la finitude des indices n et l. Les variables d'échantillonnage n, l (k est sans importance) sont en effet limitées par $n = \{0, \ldots, 255\}$ et $l = \{0, \ldots, 255\}$. Cela revient à supposer qu'une séquence originale $o_{l,k}[n]$, pour laquelle $n, l \in \mathbb{Z}$, a été multipliée par des fenêtres rectangulaires $\mathrm{rect}[n]$ et $\mathrm{rect}[l]$ de longueur N et L respectivement, formant celle $y_{l,k}[n]$. Étudions l'impact sur le spectre selon n, la méthode tenant pour l également.

Puisque la séquence $y_{l,k}[n]$ résulte de la multiplication de $o_{l,k}[n]$ par rect$[n]$, son spectre est égal à la convolution de $O_{l,k}(e^{j\Omega})$ avec Rect$(e^{j\Omega})$ (à un facteur $1/2\pi$ près, négligé). Le spectre Rect$(e^{j\Omega})$ est établi dans le chapitre suivant et vaut

$$\text{Rect}\left(e^{j\Omega}\right) = \frac{1}{N} e^{-j\frac{\Omega}{2}(N-1)} \frac{\sin\left(\Omega\frac{N}{2}\right)}{\sin\left(\frac{\Omega}{2}\right)}$$

Un pic en $\Omega = \Omega_0$, modélisé par $O(e^{j\Omega}) = \delta(\Omega - \Omega_0)$, induit que $Y_{l,k}(e^{j\Omega}) = \text{Rect}(e^{j(\Omega-\Omega_0)})$. Graphiquement, on observe qu'une copie de l'oscillation de Rect$(e^{j\Omega})$ a été centrée en Ω_0, comme mis en évidence sur la FIG. 3.10. Il en est de même pour tout autre point non-nul de $O_{l,k}(e^{j\Omega})$.

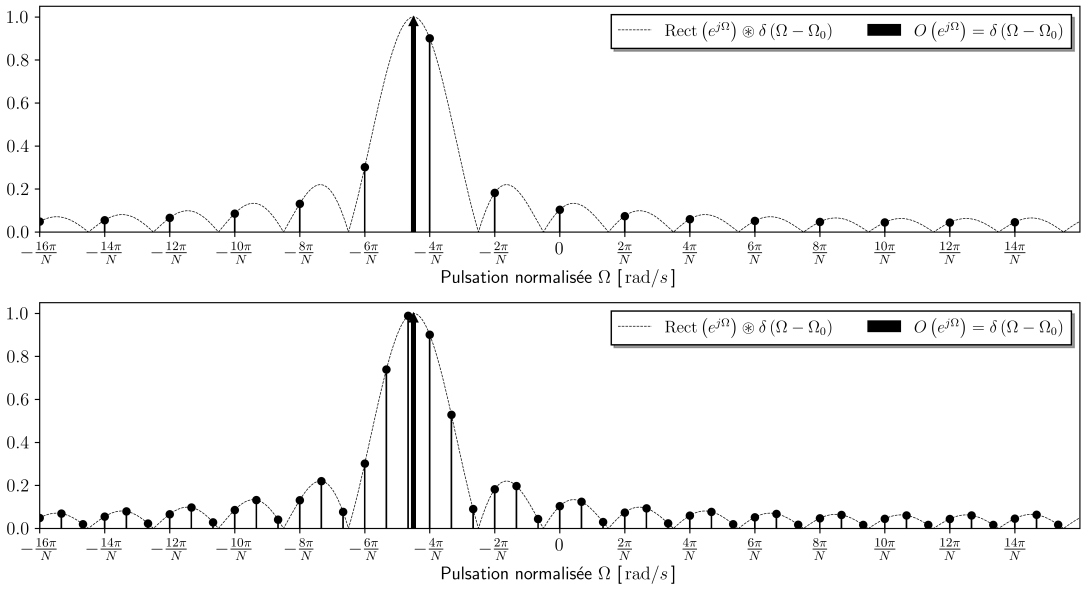

FIGURE 3.10 – Effet du sur-échantillonnage sur le résultat de la FFT avec $N = 16$ et $\beta = 3$ et $\Omega_0 \approx -0.884$ rad/s. Un plus grand nombre d'échant. améliore l'estimation du pic, mais accentue le profil des lobes secondaires. Représentation de l'amplitude.

Il est enfin utile de préciser que ce procédé permet d'accéder à plus de points du spectre *résultant*. Il améliore l'estimation des pics de ce dernier, et rend visibles des objets qui ne l'étaient pas. Cela dit, il n'augmente en rien la résolution en distance ou vitesse, car celle-ci est déjà présente dans le spectre résultant : deux pics séparés par une distance inférieure à la résolution, qui n'a pas changé, ne pourront pas être discernés pour autant et apparaîtront toujours comme un seul large pic. Ceci est représenté à la FIG. 3.11, tandis que la FIG. 3.12 compare deux graphes Doppler, l'un sans sur-échantillonnage et l'autre avec.

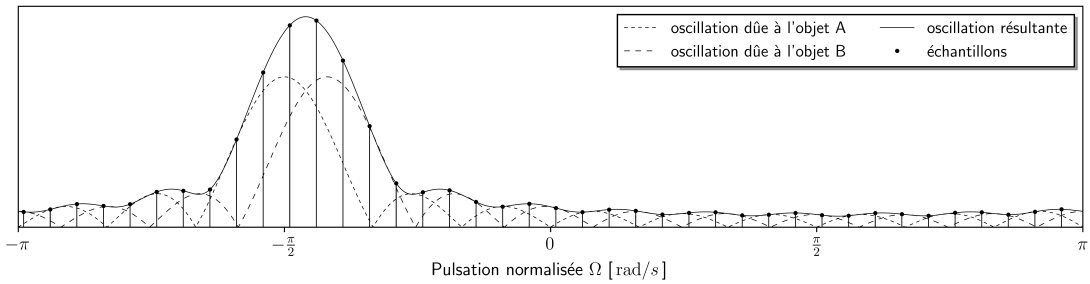

FIGURE 3.11 – Le sur-échantillonnage permet de mieux connaître la forme du spectre observable. On peut ainsi mieux localiser le ou les maxima de la *superposition* des éléments sous-jacents. La résolution en distance ou vitesse n'est malgré cela pas meilleure : les deux pics originaux, trop proches, sont toujours confondus.

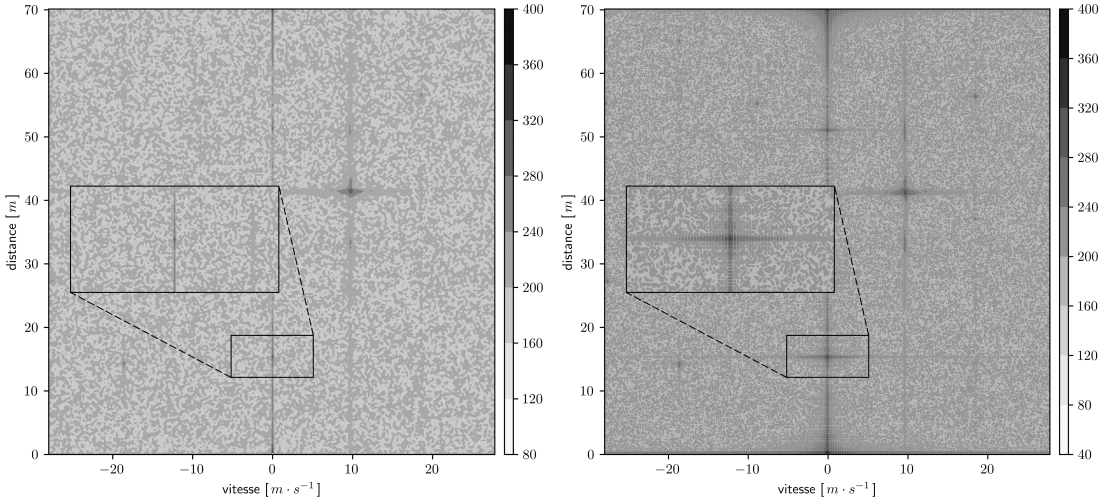

FIGURE 3.12 – Comparaison d'un graphe sans (gauche) et avec (droite) sur-échantillonnage. On remarque que l'objet encadré est peu visible dans le graphe du haut, alors qu'il apparaît clairement avec $\beta = 6$. Les oscillations sont en revanche plus marquées et peuvent complexifier la recherche des objets dans le graphe.

La compréhension approfondie du contenu de ce chapitre a été appuyée par la littérature. On peut citer [44], [45], [46], [47], [48], [49] et [50].

Chapitre 4

Filtrage et Interpolation

Ce chapitre détaille le filtrage des graphes Doppler et l'interpolation des images. Les graphes Doppler produits au chapitre précédent, dont la FIG. 3.5 est un exemple, révèlent en effet qu'une partie du bruit est constante. Elle se compose d'une fine ligne verticale au centre de l'image, ainsi que de points fantômes qui laissent penser à la présence de véhicules, à tort. Il est particulièrement important de les retirer afin de ne pas induire la détection des objets, abordée au chapitre suivant, en erreur. La seconde partie de ce chapitre est dédiée à l'interpolation des images. On souhaite en effet créer des paires comprenant un graphe Doppler et une image RGB capturés concurremment. Puisque les outils d'acquisition n'assurent pas une telle synchronisation, il faut soit se contenter de celles enregistrées juste avant ou juste après, soit générer de nouvelles images sur base de celles dont on dispose. Les images reconstituées doivent correspondre au mieux à celles qui *auraient* été prises aux instants adéquats ; c'est-à-dire aux mêmes instants que les graphes Doppler.

4.1 Filtrage des graphes Doppler

On cherche à débarrasser le signal de ses composantes constantes (ou presque). En d'autres termes, on vise à lui retirer son contenu spectral proche de 0 Hz. Ça tombe bien : la théorie des filtres n'en est plus à ses balbutiements et nombre d'idées ont déjà été présentées dans la littérature. Sont retenus dans ce travail : le filtre à moyenne mobile (MA, pour *moving average*), le filtre triangulaire et le filtre à moyenne mobile exponentielle (EMA, pour *exponential moving average*), qui paraissent les plus populaires. Ce dernier se démarque, particulièrement dans l'industrie pour sa (très) faible complexité. Ceci dit, plutôt que de l'appliquer par mimétisme, il paraît légitime d'envisager chacun des trois filtres et d'en analyser la complexité, ainsi que la qualité du filtrage (effet passe-bas).

Soit la séquence $t[n] \in \mathbb{R}^{N \times L}$, $n \in [-\infty, \infty]$ des graphes Doppler, où n représente l'indice des trames dont l'origine $n = 0$ n'a pas d'importance. L'attention peut être focalisée sur un point particulier $t_{i,j}$ de chaque trame, de la même manière qu'on suivrait l'évolution d'un seul pixel d'image en image au sein d'une bande vidéo. Ce point forme lui-même une séquence : $t_{i,j}[n]$, $n \in [-\infty, \infty]$. La séquence $t[n]$ peut ainsi être perçue comme $N \times L$ séquences parallèles, juxtaposées :

$$t[n] = \begin{bmatrix} t_{0,0}[n] & \dots & t_{0,L-1}[n] \\ \vdots & \ddots & \vdots \\ t_{N-1,0}[n] & \dots & t_{N-1,L-1}[n] \end{bmatrix}$$

Appliquer un filtre $h[n]$ à $t[n]$ est équivalent à l'appliquer à chaque $t_{i,j}[n]$:

$$y[n] = t[n] \circledast h[n] = \begin{bmatrix} t_{0,0}[n] \circledast h[n] & \dots & t_{0,L-1}[n] \circledast h[n] \\ \vdots & \ddots & \vdots \\ t_{N-1,0}[n] \circledast h[n] & \dots & t_{N-1,L-1}[n] \circledast h[n] \end{bmatrix} = \begin{bmatrix} y_{0,0}[n] & \dots & y_{0,L-1}[n] \\ \vdots & \ddots & \vdots \\ y_{N-1,0}[n] & \dots & y_{N-1,L-1}[n] \end{bmatrix}$$

où $y_{i,j}[n] = t_{i,j}[n] \circledast h[n]$. Une fois $y[n]$ obtenu par passage de $t[n]$ dans le filtre $h[n]$ passe-bas, on le soustrait à $t[n]$ pour former la séquence $t'[n] = t[n] - y[n] = t[n] - h[n] \circledast t[n] = (\delta[n] - h[n]) \circledast t[n]$. Étudions maintenant les filtres $h[n]$ susnommés.

4.1.1 Filtre à moyenne mobile (MA)

Le filtre à moyenne mobile de taille P est particulièrement simple : les P dernières valeurs du signal sont moyennées. On écrit

$$h[n] = \begin{cases} \frac{1}{P} & n = 0, \dots, P-1 \\ 0 & \text{ailleurs} \end{cases}$$

Il s'agit d'un filtre à réponse impulsionnelle finie (FIR, pour *finite impulse response*). Comme son nom l'indique, la sortie du filtre est finie : elle s'annule définitivement à partir d'un certain temps après excitation par une brève impulsion. Dans le cas d'un filtre MA, ce délai vaut P. La réponse impulsionnelle est représentée à la FIG. 4.1

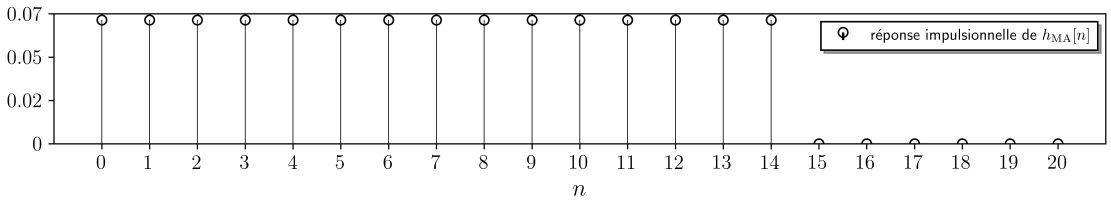

FIGURE 4.1 – Séquence $h[n]$ pour le filtre MA avec $P = 15$.

4.1.1.1 Complexité

On remarque que

$$\begin{aligned} y[n] &= \frac{1}{P}\sum_{p=0}^{P-1} t[n-p] = \frac{1}{P}\sum_{p=0}^{P-1} t[n-p-1] + \frac{1}{P}\left(t[n] - t[n-P]\right) \\ &= y[n-1] + \frac{1}{P}\left(t[n] - t[n-P]\right) \end{aligned}$$

ce qui signifie qu'à chaque itération, 2 additions et 1 multiplication, soit 3 opérations arithmétiques, suffisent. Il faut cependant garder à l'esprit qu'il s'agit d'opérations complexes. Chaque addition ou multiplication (car $1/P \in \mathbb{R}$) [1] en cache donc deux, portant le total à 6 opérations arithmétiques élémentaires. On multiplie enfin celui-ci par $N \times L$ puisqu'il s'agit des graphiques $t[n]$ contenant $N \times L$ valeurs.

En ce qui concerne la mémoire, il est nécessaire de conserver P graphes dans une mémoire tampon. Cela représente $N \times L \times 8 \times P = 512P$ kB pour $N = L = 256$. Le facteur 8 est dû aux parties réelle et imaginaire, occupant chacune l'équivalent d'un entier (4 bytes).

4.1.1.2 Contenu fréquentiel

La réponse en fréquence peut être calculée ainsi :

$$\begin{aligned} H\left(e^{j\Omega}\right) &= \sum_{n=-\infty}^{+\infty} h[n]e^{-j\Omega n} = \sum_{n=0}^{P-1} \frac{1}{P} e^{-j\Omega n} = \frac{1}{P}\frac{1-e^{-j\Omega P}}{1-e^{-j\Omega}} \\ &= \frac{1}{P}\frac{e^{-j\Omega\frac{P}{2}}}{e^{-j\frac{\Omega}{2}}}\frac{e^{j\Omega\frac{P}{2}} - e^{-j\Omega\frac{P}{2}}}{e^{j\frac{\Omega}{2}} - e^{-j\frac{\Omega}{2}}} = \frac{1}{P}e^{-j\frac{\Omega}{2}(P-1)}\frac{\sin\left(\Omega\frac{P}{2}\right)}{\sin\left(\frac{\Omega}{2}\right)} \end{aligned}$$

grâce à $\sum_{i=0}^{I-1} a^i = (1-a^I)/(1-a)$ et l'identité $2j\sin(x) = e^{jx} - e^{-jx}$.

La norme ξ des coefficients du filtre est telle que le gain DC $H(e^{j0})$ soit unitaire :

$$H\left(e^{j0}\right) = \sum_{n=-\infty}^{+\infty} h[n]e^{j0} = \sum_{n=0}^{P-1} \xi = P\xi = 1$$

soit $\xi = 1/P$. Si cette condition n'était pas respectée, la composante DC serait sur- ou sous-évaluée et, indirectement, mal retirée dans les images.

1. Les multiplications complexes dissimulent en réalité 4 produits et 2 additions, selon $(a+jb)(c+jd) = (ac-bd) + j(ad+bc)$. Si $b = 0$, seules 2 multiplications sont requises.

4.1.2 Filtre triangulaire

La réponse du filtre triangulaire est finie : il s'agit aussi d'un filtre FIR. La norme de ses coefficients croît puis décroît linéairement. La séquence s'énonce

$$h[n] = \begin{cases} \zeta - \left| n - \frac{P-1}{2} \right| \Delta & n = 0, \ldots, P-1 \\ 0 & \text{ailleurs} \end{cases}$$

où P est impair. ζ et $\Delta = 2a/(P+1)$ sont tels que le gain DC soit unitaire :

$$\begin{aligned} H\left(e^{j0}\right) &= \sum_{n=-\infty}^{+\infty} h[n] e^{j0} = \sum_{n=0}^{P-1} h[n] = \zeta + 2 \cdot \sum_{n=1}^{(P-1)/2} n\Delta \\ &= \zeta + 2 \cdot \sum_{n=1}^{(P-1)/2} n \frac{2\zeta}{P+1} = \zeta \left(1 + \left(1 + \frac{P-1}{2}\right) \frac{P-1}{P+1}\right) = 1 \end{aligned}$$

La réponse impulsionnelle est illustrée ci-dessous :

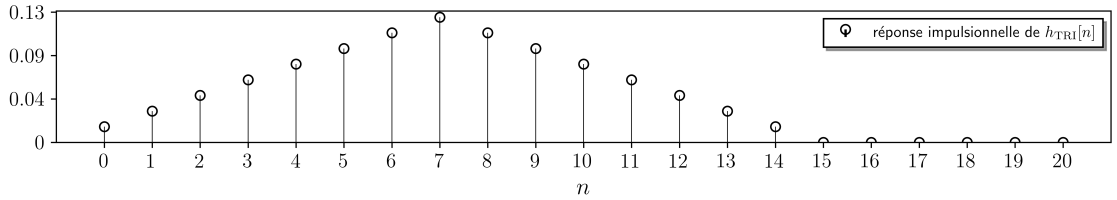

FIGURE 4.2 – Séquence $h[n]$ pour le filtre triangulaire avec $P = 15$.

4.1.2.1 Complexité

Chaque itération requiert P multiplications et $P-1$ additions pour le calcul de la sortie $y[n]$. On atteint ainsi $2P-1$ opérations arithmétiques complexes, et $6P + 2(P-1) = 2(4P-1)$ opérations arithmétiques élémentaires. Ce résultat est à multiplier par $N \times L$.

Pour la mémoire, la conclusion est la même que le filtre MA : P graphes à conserver, ce qui représente $512P$ kB.

4.1.2.2 Contenu fréquentiel

Dériver l'expression de $H(e^{j\Omega})$ à partir de la définition n'est pas chose facile à cause des termes $n(\cdot)^n$. On peut en revanche considérer $h[n]$ comme le résultat de la convolution d'une fenêtre rectangulaire $w[n]$ avec elle-même. Pour que $h[n]$ soit de taille P impair, $w[n]$ doit être de taille $P' = (P+1)/2$. La norme ξ de ses coefficients doit garantir que le sommet de la convolution, qui vaut $P'\xi^2$, coïncide

avec $\zeta : \xi = \sqrt{\zeta/P'}$. Le spectre d'une telle fenêtre a été calculé en 4.1.1.2, où il faut changer la taille P en P' et le terme $1/P$ en $\sqrt{\xi/P'}$:

$$W\left(e^{j\Omega}\right) = \sqrt{\frac{2\zeta}{P+1}} e^{-j\frac{\Omega}{2}\frac{P-1}{2}} \frac{\sin\left(\Omega\frac{P+1}{4}\right)}{\sin\left(\frac{\Omega}{2}\right)}$$

Une convolution en temporel est synonyme d'une multiplication en fréquentiel : le spectre de $t[n] = w[n] \circledast w[n]$ vaut

$$H\left(e^{j\Omega}\right) = W\left(e^{j\Omega}\right) W\left(e^{j\Omega}\right) = \frac{2\zeta}{P+1} e^{-j\Omega\frac{P-1}{2}} \frac{\sin^2\left(\Omega\frac{P+1}{4}\right)}{\sin^2\left(\frac{\Omega}{2}\right)}$$

4.1.3 Filtre à moyenne mobile exponentielle (EMA)

Contrairement aux deux précédents, la réponse impulsionnelle du filtre EMA n'est pas finie. Elle tend vers 0, sans jamais l'atteindre. On parle de filtre IIR (*infinite impulse response*). Avant de définir les coefficients $h[n]$, jetons un oeil à l'équation de récurrence qui le constitue. La sortie $y[n]$ est liée à l'entrée $x[n]$ par

$$\begin{aligned} y[n] &= \alpha x[n] + (1-\alpha)\, y[n] \\ &= \sum_{m=-\infty}^{+\infty} \alpha\,(1-\alpha)^m\, x[n-m] = h[n] \circledast x[n] \end{aligned} \tag{4.1}$$

On conclut, comme le montre la FIG. 4.3, que

$$h[n] = \begin{cases} \alpha\,(1-\alpha)^n & n \geq 0 \\ 0 & \text{ailleurs} \end{cases}$$

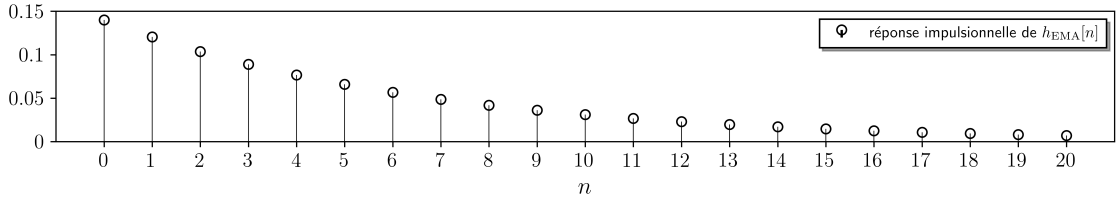

FIGURE 4.3 – Séquence $h[n]$ pour un filtre EMA avec $\alpha = 0.2$.

4.1.3.1 Complexité

L'équation de récurrence (4.1) inclut 3 opérations arithmétiques : 2 multiplications et 1 addition, soit 2 additions et 4 multiplications réelles puisque $\alpha \in \mathbb{R}$. On obtient $8\,(N \times L)$ opérations élémentaires.

Le filtre EMA ne nécessite que très peu de mémoire : seule la séquence $y[n]$ doit être enregistrée d'itération en itération, donc 512 kB quel que soit α.

4.1.3.2 Contenu fréquentiel

Le spectre du filtre $h[n]$ se calcule comme suit :

$$\begin{aligned} H\left(e^{j\Omega}\right) &= \sum_{n=-\infty}^{\infty} h[n]e^{-j\Omega n} = \sum_{n=0}^{+\infty} \alpha\left(1-\alpha\right)^n e^{-j\Omega n} \\ &= \alpha \sum_{n=0}^{+\infty} \left(\left(1-\alpha\right)e^{-j\Omega}\right)^n = \frac{\alpha}{1-\left(1-\alpha\right)e^{-j\Omega}} \end{aligned}$$

Puisqu'on veut un filtre stable, il faut $0 \leq \alpha \leq 1$. Le gain DC unitaire est toujours respecté car, pour tout α,

$$H\left(e^{j0}\right) = \frac{\alpha}{1-\left(1-\alpha\right)e^{j0}} = \frac{\alpha}{\alpha} = 1.$$

4.1.4 Comparaison et discussion

4.1.4.1 Fréquence de coupure

On appelle fréquence de coupure la fréquence à partir de laquelle le contenu ne passe plus à travers un filtre. En réalité, ce n'est pas possible de réaliser un filtre idéal et la coupure n'est pas si précise : les fréquences avant celle de coupure sont *peu* affectées, tandis que celles après le sont *nettement plus*. On considère généralement qu'une fréquence se trouve dans la zone coupée si l'amplitude d'une sinusoïde à cette même fréquence est réduite de plus de moitié. Autrement dit, la fréquence de coupure f_{cp} est telle que

$$\left|H\left(j\omega_{\text{cp}}\right)\right| = \frac{1}{2} \qquad \longleftrightarrow \qquad 10 \cdot \log_{10}\left(\left|H\left(j\omega_{\text{cp}}\right)\right|\right) = -3 \text{ dB}$$

On considère dans ce travail qu'un signal stationnaire plus de deux secondes est à retirer. À raison de 12 graphes par seconde, soit $f_e = 12$ Hz, la fréquence de coupure Ω_{cp} vaut

$$\Omega_{\text{cp}} = 2\pi F_{\text{cp}} = 2\pi \frac{f_{\text{cp}}}{f_e} = \frac{2\pi}{2 \cdot 12} = 0.26 \text{ rad/s}$$

4.1.4.2 Dimensionnement des filtres

La résolution numérique de $H(e^{j\Omega_{\text{cp}}}) = 1/2$ pour chacun des trois filtres fournit les paramètres rassemblés dans la TABLE 4.1.

	sol. num.	P, α
Filtre MA	14.506	$P = 15$
Filtre triangulaire	20.342	$P = 21$
Filtre EMA	0.140	$\alpha = 0.14$

TABLE 4.1 – Paramètres des filtres pour $\Omega_{\text{cp}} = \pi/12$ rad/s.

La réponse des filtres est affichée à la FIG. 4.4.

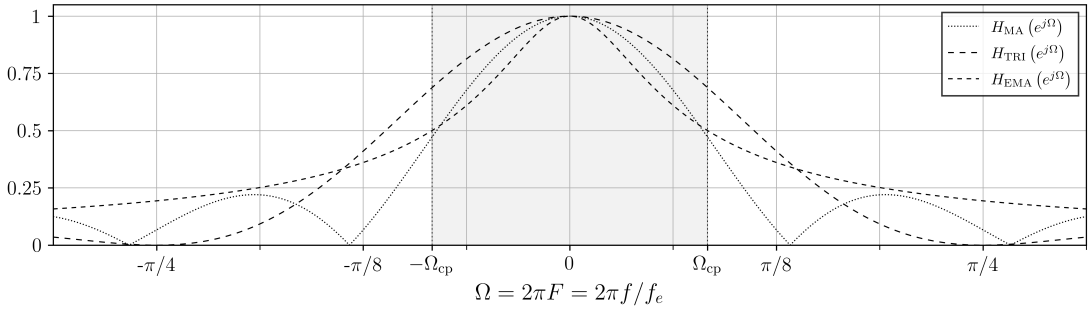

FIGURE 4.4 – Réponse fréquentielle des filtres dimensionnés pour $\Omega_{\text{cp}} = 2\pi/24$, ce qui correspond au retrait du contenu oscillant à une période supérieure à 2 secondes. À cause des arrondis, puisque $P \in \mathbb{N}$, les filtres s'intersectent en $\Omega \approx 0.24$ rad/s. Cela ne pose pas de problème. Représentation de l'amplitude.

4.1.4.3 Choix du filtre

Effet passe-bas

Pour quantifier cette analyse, introduisons une métrique $\gamma(H(e^{j\Omega}), \Omega_{\text{cp}})$ telle qu'elle pénalise les basses fréquences atténuées et les hautes fréquences qui ne le sont pas. Elle tient compte du fait que le spectre d'un signal réel est symétrique (évaluation des intégrales sur la moitié du domaine) :

$$\gamma\left(H\left(e^{j\Omega}\right), \Omega_{\text{cp}}\right) = \int_0^{\Omega_{\text{cp}}} 1 - \left|H\left(e^{j\Omega}\right)\right| \mathrm{d}\Omega + \int_{\Omega_{\text{cp}}}^{\pi} \left|H\left(e^{j\Omega}\right)\right| \mathrm{d}\Omega$$

Une méthode d'intégration numérique fournit les résultats rassemblés dans la TABLE 4.2. Le tableau confirme la tendance visible à la FIG. 4.4 : l'effet passe-bas du filtre triangulaire est meilleur que celui des deux autres.

	terme 1	terme 2	total
Filtre MA	0.05	0.23	0.28
Filtre triangulaire	0.05	0.07	0.12
Filtre EMA	0.06	0.41	0.47

TABLE 4.2 – Métrique γ appliquée aux trois filtres étudiés pour les paramètres de la TABLE 4.1.

Complexité algorithmique

Si on tient compte des paramètres de la TABLE 4.1, la complexité algorithmique (nombre d'opérations arithmétiques élémentaires nécessaires) de chaque filtre est telle que résumée dans la TABLE 4.3. Celle du filtre triangulaire est 27.6 fois supérieure à celle des filtres MA et EMA.

	en général	paramètre	cas étudié
Filtre MA	$6NL$	/	$6NL$
Filtre triangulaire	$2\,(4P-1)\,NL$	$P = 21$	$166NL$
Filtre EMA	$6NL$	/	$6NL$

TABLE 4.3 – Nombres d'opérations élémentaires pour l'application des trois filtres étudiés avec les paramètres de la TABLE 4.1.

Espace mémoire requis

L'espace mémoire requis pour chaque filtre est indiqué dans la TABLE 4.4. Par rapport au filtre MA, cet espace est 15 fois plus faible pour le filtre EMA, 21 fois par rapport au filtre triangulaire.

	en général [kB]	paramètre	cas étudié [kB]
Filtre MA	$512P$	$P = 15$	7 680
Filtre triangulaire	$512P$	$P = 21$	10 752
Filtre EMA	512	/	512

TABLE 4.4 – Espace mémoire requis pour l'application des trois filtres étudiés avec les paramètres de la TABLE 4.1.

Discussion

Sur base de la comparaison dressée ci-dessus, il a été choisi d'implémenter les filtres triangulaire et EMA. L'utilisateur a ainsi la possibilité d'obtenir un meilleur effet passe-bas ou d'opter pour une exécution plus rapide. Le filtre EMA a été choisi comme filtre par défaut.

Les résultats doivent cependant être nuancés : la différence entre les graphes en sortie des deux filtres est faible et parfois même à peine perceptible pour l'œil humain. Plus surprenant : la différence de temps n'est pas aussi flagrante que le laisserait penser la TABLE 4.3. Un test réalisé sur 15 000 itérations révèle qu'il faut compter 3.324 ms par échantillon pour le filtre EMA et 4.438 ms pour le

filtre triangulaire. Ceci s'explique sans doute par l'emploi de Python, qui est un langage haut niveau interprété. Cela implique de nombreuses routines communes en arrière-plan (comme le garbage collector, par exemple), masquant l'incidence de la partie propre à chaque méthode. La différence de temps d'exécution serait probablement nettement plus marquée en `C`.

4.1.5 Validation

La FIG. 4.5 expose deux exemples, pris au hasard parmi plusieurs prises de vue, et valide le bon fonctionnement des filtres, qui remplissent leur rôle : la ligne centrale et les deux points fantômes sont effacés.

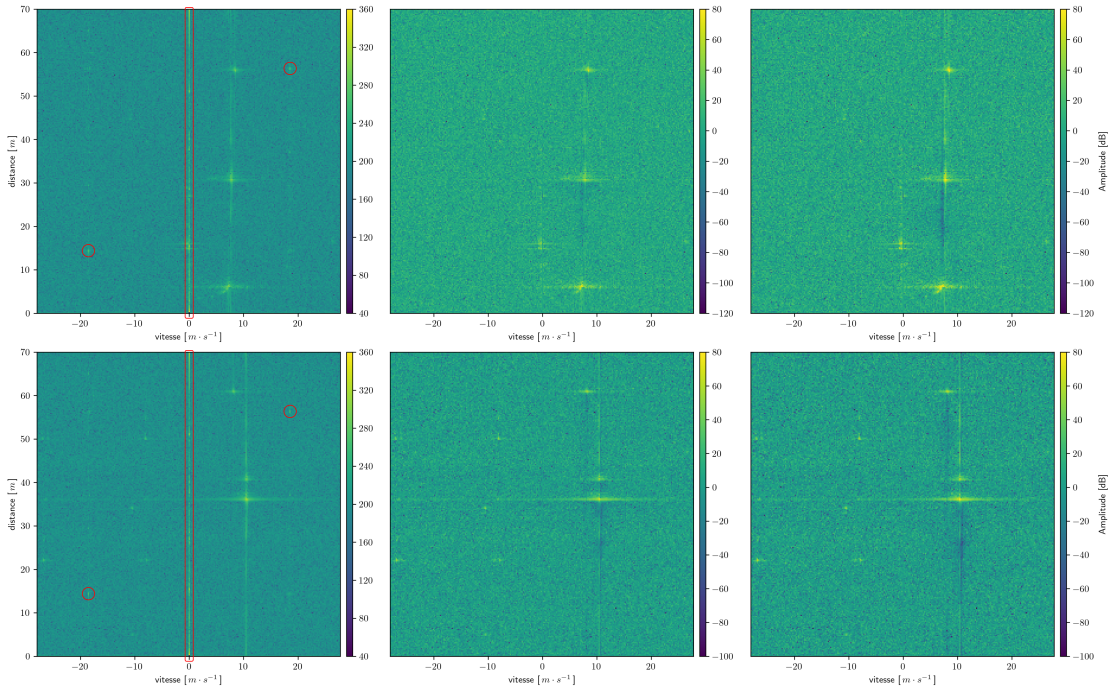

FIGURE 4.5 – **Gauche :** graphe Doppler sans filtre. **Milieu :** sortie du filtre EMA. **Droite :** sortie du filtre triangulaire. On peut vérifier que ceux-ci retirent la ligne centrale et les deux points fantômes (marqués en rouge sur le graphe sans filtre).

4.2 Génération des images intermédiaires

Il faut maintenant trouver quelles images associer aux graphes Doppler. Pour cela, trois approches différentes sont investiguées : une mise en correspondance directe, l'interpolation (linéaire, Lagrange), et l'estimation de mouvements.

L'alignement des séquences, mentionné à plusieurs reprises dans la suite de ce document, est l'étape qui détermine l'emplacement des trames du radar par rapport au flux des images. Pour l'illustrer, supposons que la prise de vue consiste en n_{tr} trames et n_{im} images. L'algorithme d'alignement des séquences fournira un vecteur v de taille n_{tr} où $v_i = j$ signifie que les images j et $j+1$ ont été prises juste avant et juste après la trame i, respectivement.

4.2.1 Mise en correspondance

Une manière simple et efficace d'associer une image à chaque graphe est de sélectionner l'image dont l'estampille est la plus proche de celle de celui-ci. L'implémentation est bien sûr particulièrement simple : l'alignement des séquences réalisé en amont fournit déjà l'indice des images dont les estampilles encadrent (avant et après) celle de chaque graphe.

Si l'estampille d'un graphe vaut t_{gr} et que l'alignement des séquences indique que les images j et $j+1$ ont été prise juste avant et juste après, la seule comparaison de $|t_{\text{gr}} - t_{\text{im},j}|$ et $|t_{\text{gr}} - t_{\text{im},j+1}|$ indique quelle image choisir. À raison de 30 fps, l'erreur introduite est strictement inférieure à $1/2 \times 1/30 = 16.7$ ms. Ceci suffit déjà à bon nombre d'applications.

4.2.2 Interpolation

4.2.2.1 Interpolation linéaire

L'interpolation linéaire se base sur deux valeurs connues et consiste à tracer la droite qui les joint. On sélectionne alors le point de la droite à l'abscisse souhaitée. Plus formellement, on peut écrire que si on connaît les points (x_0, y_0), (x_1, y_1) et qu'on cherche la valeur en $x \in [x_0, x_1]$, on trace la droite

$$f(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}\,(x - x_0) \tag{4.2}$$

En considérant le flux d'images régulier, on peut considérer que $\Delta x = x_1 - x_0$ est constant et écrire (4.2) sous la forme

$$f(x) = y_0 + (y_1 - y_0)\frac{x - x_0}{\Delta x} = y_0 + (y_1 - y_0)\delta x = y_0\,(1 - \delta x) + y_1 \delta x$$

où $\delta x = (x - x_0)/\Delta x$. Pour une meilleure compréhension, ré-écrivons cette relation avec les variables appropriées :

$$y = (1 - \delta t)\, y_{\mathrm{im},j} + \delta t\, y_{\mathrm{im},j+1} \tag{4.3}$$

où $\delta x = (t_{\mathrm{gr}} - t_{\mathrm{im},j})/\Delta t$ et $\Delta t = 1/30$. L'interpolation linéaire est donc en réalité une pondération par $1 - \delta t$ et δt des images j et $j + 1$ entourant le graphe. La FIG. 4.6 montre une image obtenue par (4.3). On observe que le résultat est satisfaisant (avec $\delta t \approx 0$ ou $\delta t \approx 1$). D'autres évaluations ont cependant révélé que ce n'était pas le cas lorsque $\delta t \approx 1/2$: la contribution de chaque image apparaît dans l'image produite. Ce phénomène sera illustré à la FIG. 4.10, page 59.

FIGURE 4.6 – Image produite (centre) par interpolation linéaire des images j (haut) et $j + 1$ (bas) avec $\delta t = 0.891$ s. Le résultat est satisfaisant.

4.2.2.2 Interpolation de Lagrange

L'interpolation de Lagrange construit un polynôme de degré supérieur à 1 en utilisant plus de points : n points établissent un polynôme de degré (au plus) $n-1$. Pour constituer ce polynôme, il faut commencer par déterminer ceux dits «de Lagrange» qui seront associés à chaque abscisse. On définit un polynôme de Lagrange pour l'abscisse i parmi les abscisses $0, \ldots, n-1$ comme :

$$l_i(x) = \prod_{j=0, j\neq i}^{n-1} \frac{x - x_j}{x_i - x_j} \tag{4.4}$$

qui peut s'interpréter ainsi : le numérateur assure que le polynôme $l_i(x_j)$ s'annule aux abscisses des points $j \neq i$, tandis que le dénominateur garantit que $l_i(x_i) = 1$. En d'autres termes, $l_i(x_j) = \delta_{ij}$ pour tous $0 \leq i, j \leq n-1$. $l_i(x)$ est en outre de degré au plus $n-1$. Chaque polynôme de Lagrange $l_i(x)$ est ensuite pondéré par la valeur y_i. En intégrant (4.4), la fonction d'interpolation s'énonce :

$$y(x) = \sum_{i=0}^{n-1} l_i(x) \cdot y_i = \sum_{i=0}^{n-1} \prod_{j=0, j\neq i}^{n-1} \frac{x - x_j}{x_i - x_j} \cdot y_i \tag{4.5}$$

Dans ce travail, quatre images sont exploitées : les deux précédentes et les deux suivantes. En supposant l'écart Δt constant entre les images reçues de la caméra et en normalisant par ce dernier, on peut se ramener au cas suivant : on connaît les images aux abscisses $\{t_0, t_1, t_2, t_3\} = \{-1, 0, 1, 2\}$ et on cherche celle pour $t \in [0, 1]$.

Les polynômes de Lagrange sont respectivement :

$$l_0(x) = -1/6\left(x^3 - 3x^2 + 2x\right) \qquad l_1(x) = 1/2\left(x^3 - 2x^2 - x + 2\right)$$
$$l_2(x) = -1/2\left(x^3 - x^2 - 2x\right) \qquad l_3(x) = 1/6\left(x^3 - x\right)$$

et il ne reste plus qu'à appliquer (4.5) avec

$$\{y_0, y_1, y_2, y_3\} = \{y_{\mathrm{im},j-1}, y_{\mathrm{im},j}, y_{\mathrm{im},j+1}, y_{\mathrm{im},j+2}\}$$

La FIG. 4.7 montre une image obtenue par (4.5). À l'instar de l'interpolation linéaire, le résultat est satisfaisant pour $\delta t \approx 0$ ou $\delta t \approx 1$, tandis que les essais mettent au jour des limitations de la méthode lorsque $\delta t \approx 1/2$. Remarquons que l'intérêt des interpolations linéaire et de Lagrange est limité puisqu'elles fonctionnent lorsque $\delta t \approx 0$ ou $\delta t \approx 1$, ce qui revient à appliquer, à une légère variation près, la méthode de mise en correspondance élémentaire discutée au point 4.2.1.

Bornes

Contrairement à l'interpolation linéaire, où $y \in [0, 255]$ si $y_{\text{im},j}, y_{\text{im},j+1} \in [0, 255]$, le résultat de l'interpolation de Lagrange peut excéder les limites de cet intervalle. Il est donc primordial d'y être attentif. Pour y remédier, soit les valeurs sont mises à l'échelle via $y_{i,j} \leftarrow (\max(y) - \min(y))/255 \cdot (y_{i,j} - \min(y))$, soit $y_{i,j} \leftarrow 0$ si $y_{i,j} < 0$ et $y_{i,j} \leftarrow 255$ si $y_{i,j} > 255$ (option choisie). Cela n'a que peu d'incidence car les essais ont montré qu'au maximum 0.03% des pixels étaient concernés.

FIGURE 4.7 – Image produite (centre) par interpolation de Lagrange des images j (haut) et $j+1$ (bas) avec $\delta t = 0.891$ s. Les images $j-1$ et $j+2$ ne sont pas affichées. Le résultat est satisfaisant, mais ne diffère que très peu de celui de l'interpolation linéaire.

4.2.3 Prédiction par estimation des mouvements

Pour des objets fixes, une interpolation peut avoir du sens (on pense par exemple à un changement de luminosité). On a constaté que cette technique n'était malheureusement pas concluante. La raison est simple : pour recréer une image sur base de celles qui l'entourent, il est nécessaire d'analyser son contenu, car les objets mobiles impliquent quant à eux des changements de couleur brusques pour les pixels. Dans le domaine du traitement du signal, on distingue pour cette raison l'arrière-plan de l'avant-plan (ou premier-plan). L'arrière-plan regroupe par convention les éléments fixes, le reste appartient à l'avant-plan.

En quoi l'analyse des mouvements peut-elle mener à la génération des images intermédiaires ? Supposons que le vecteur de déplacement d (par rapport à une image de référence qu'on suppose en $t = t_0$ quelconque) d'un objet $\mathcal{O}$ soit connu aux temps t_A et t_B. Au temps t tel que $t_A \leq t \leq t_B$ et $\delta t = (t - t_A)/(t_B - t_A)$, on pourra calculer $d_B = (1 - \delta t)\, d_A + \delta t\, d_B$, à l'instar de l'interpolation linéaire développée en 4.2.2.1 (ou tout autre forme d'interpolation). Elle est ici justifiée par le caractère continu du vecteur de déplacement d, contrairement au profil discontinu que les objets mobiles font subir à la couleur des pixels.

4.2.3.1 Implémentation directe

La compensation de mouvements jouit d'un certain engouement autour du domaine audiovisuel en pleine expansion. L'augmentation artificielle du taux d'images intéresse en effet beaucoup le 7$^{\text{e}}$ art, car elle rend entre autres plus fluides les mouvements de la caméra et améliore les effets de ralenti. Il existe pour cette raison une variété certaine de logiciels, certains libres d'autres non, offrant la possibilité d'appliquer les techniques de compensation de mouvements à une suite d'images. Dans le cas présent, c'est `FFmpeg`, un logiciel libre de conversion et de traitement multimédia, qui a été employé. Il est basé sur des méthodes dites «classiques», comme celles présentées dans [51]. Le résultat obtenu est visible à la FIG. 4.10. D'autres approches sont également prometteuses, comme l'étude du flux optique (*optical flow* en anglais) à laquelle contribue l'algorithme de Lucas-Kanade [52].

4.2.3.2 Exploitation de la compression H.264

En prenant du recul, une idée semble encore plus prometteuse. Pourquoi ne pas exploiter l'estimation de mouvements déjà disponible, à la place de la (re-)calculer ? Eh oui, cette estimation est en fait réalisée au sein de la caméra CMOS et sert à l'encodage de vidéos pour le protocole H.264. On peut apercevoir le bloc en bas à gauche dans la FIG. 4.9. Le protocole est expliqué en quelques mots ci-dessous.

Le protocole H.264 utilise des images de référence régulièrement espacées, qui sont encodées isolément. La compression ne tient alors compte que de la redondance spatiale, interne à l'image, d'où le nom I pour *intra-frame*. Il code ensuite les images P (pour *predicted frame*) en utilisant une prédiction en avant par l'estimation des mouvements sur base des images I. Les images B qui se trouvent entre celles I et P sont enfin encodées en employant une estimation de mouvement en avant (*forward estimation*) à partir de la dernière image I et une seconde estimation de mouvement en arrière (*backward estimation*) à partir de l'image P suivante. L'ordre d'encodage des images ne correspond donc pas à leur position dans la séquence, comme illustré par la FIG. 4.8. Ceci induit un léger délai puisqu'il faut patienter jusqu'à l'obtention de l'image de référence suivante pour encoder celles intermédiaires. Toutefois, exploité ou non, l'encodage est de toute façon réalisé par la caméra, induisant le délai, et il sera regrettable de pas profiter de la compression qui en résulte.

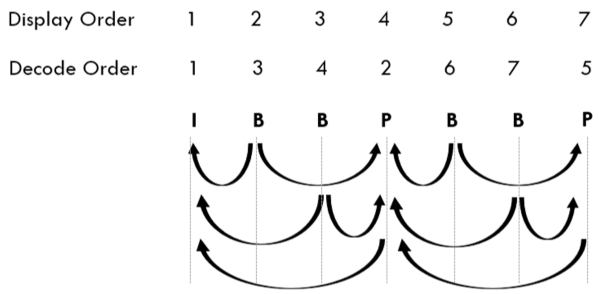

FIGURE 4.8 – Comparaison entre la position des images dans la séquence et l'ordre dans lequel elles sont encodées lors d'une compression H.264. Les flèches indiquent les images prises comme référence pour l'estimation des mouvements. Extrait de [53].

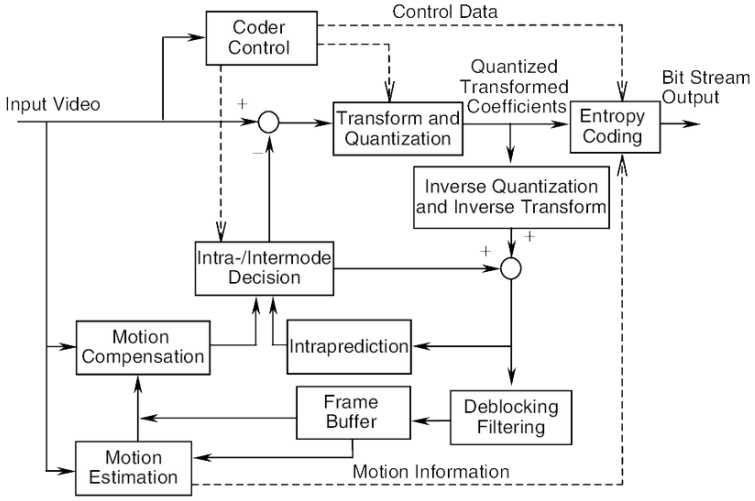

FIGURE 4.9 – Description schématique d'un encodeur H.264. Le bloc «Motion Estimation» est en bas à gauche. On voit que les vecteurs qu'il renvoie sont transmis à l'encodage entropique. Extrait de [54].

La FIG. 4.9 indique clairement que l'estimation des mouvements est transmise dans la sortie de l'encodeur. Elle permet de ne transmettre que ces vecteurs, et non l'ensemble des pixels, ce qui réduit drastiquement la quantité d'information à

envoyer. Le principe proposé est donc très simple : il s'agit de recréer un décodeur artisanal, qui serait non seulement capable de décoder la séquence originale, mais aussi de générer les images intermédiaires aux temps souhaités en interpolant les vecteurs de mouvements reçus, comme mentionné ci-dessus. C'est une piste d'amélioration encourageante pour ce travail.

FIGURE 4.10 – **Haut :** interp. linéaire. **Milieu :** interp. de Lagrange. **Bas :** comp. de mouvements.

Le résultat obtenu par estimation et compensation des mouvements est nettement meilleur. La superposition visible sur les deux premières images disparaît pour cette dernière, laissant la place à l'objet au centre ; ce qui coïncide avec la scène qui *aurait* été capturée à ce moment-là.

Chapitre 5

Constitution du jeu de données

Ce cinquième et dernier chapitre discute l'extraction du contenu des images et des graphes Doppler, ainsi que l'implémentation logicielle de tous les éléments abordés aux chapitres 3 et 4. Il donne sens aux raisonnements et recherches menés jusqu'à présent et naissance aux jeux de données, aboutissement du TFE. Ces jeux sont principalement destinés à l'entraînement de réseaux de neurones et participeront dans ce cadre à l'élaboration d'un article planifié pour octobre 2022.

5.1 Extraction du contenu des images

5.1.1 You Only Look Once (YOLO)

La classification d'images est un domaine en pleine expansion qui connaît un essor important depuis une dizaine d'années. De nombreux réseaux de neurones furent développés, démontrant un taux de succès toujours plus important ($> 82\%$ [1] de nos jours [55]). Dans ce travail, c'est l'algorithme You Only Look One (YOLO) qui assure la détection des objets présents sur les images. Il s'agit d'un algorithme proposé par J. Redmon, S. Divvala, R. Girshick et A. Farhadi en 2016 [56].

YOLO est un réseau neuronal convolutif qui, contrairement aux classificateurs employés pour la détection, ne nécessite qu'une seule observation de l'image ; particularité dont il tient son nom. De l'aveu de ses auteurs, l'algorithme est inspiré de GoogLeNet et contient 24 couches de convolution suivies de 2 couches entièrement connectées. Le réseau est affiché à la FIG. 5.1. Après avoir découpé l'image en un quadrillage de $S \times S$ blocs, il génère B cadres pour chacun d'entre eux. Chaque cadre correspond à une prédiction quant à la position d'un objet éventuel

1. $> 82\%$ pour Top-1 (une seule prédiction) et $> 95\%$ pour Top-5 (5 prédictions possibles).

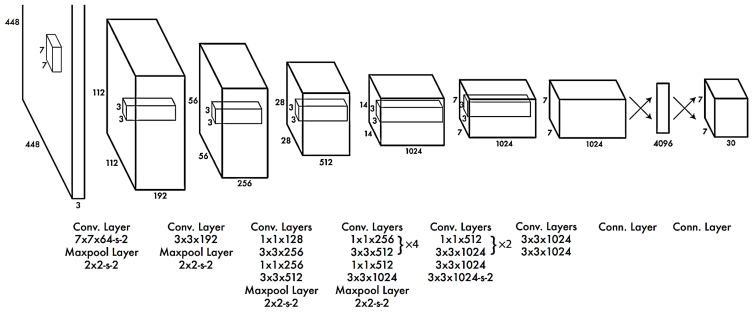

FIGURE 5.1 – Réseau de détection de YOLO. Celui-ci contient 24 couches de convolution suivies de 2 couches entièrement connectées. Extrait de [56].

et est décrit par les 5 composantes (x, y, w, h, p), où (x, y) spécifie la position du centre du cadre par rapport au bloc considéré, (w, h) indique la largeur et hauteur du cadre par rapport à l'image et $p \in [0, 1]$ correspond au degré de certitude. À chaque bloc est aussi associé un vecteur contenant une estimation de la probabilité pour chacune des C classes, indépendamment du nombre de cadres B. Par exemple, avec $S = 7$, $B = 2$ et $C = 20$ comme sur la FIG. 5.1, la prédiction est un tenseur de taille $7 \times 7 \times (2 * 5 + 20) = 7 \times 7 \times 30$. L'algorithme est résumé à la FIG. 5.2.

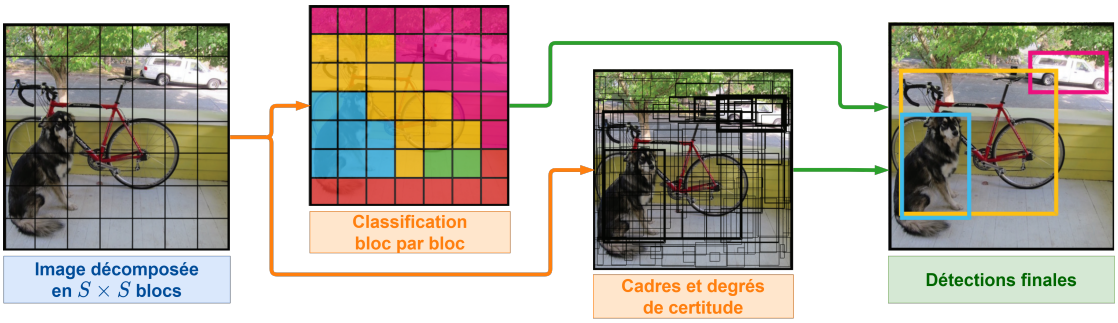

FIGURE 5.2 – Étapes clés de l'algorithme YOLO. Adapté de [56].

5.1.2 Validation

Plusieurs librairies proposent une implémentation Python de YOLO, comme `YOLOv5` d'`Ultralytics`. Plusieurs tailles de réseaux sont disponibles, avec des précisions et temps d'exécution variables [2] publiés en [57]. Pour se restreindre à un temps d'exécution raisonnable, la version `YOLOv5n` est privilégiée. Le programme permet tout de même à l'utilisateur d'employer celle qu'il souhaite.

Un seuil peut être fixé, définissant le degré de certitude au-delà duquel les éléments trouvés sont conservés. À l'inverse, les objets dont le degré de certitude est

2. Les principaux résultats ont été retranscrits dans l'annexe A.4.

plus faible seront ignorés. Ce seuil a été défini à 0.25, maintenant le taux de rappel (pourcentage d'objets détectés parmi tous ceux qu'il était possible de détecter) relativement haut, au risque de restreindre légèrement la précision (détections correctes parmi toutes les détections réalisées). Les faux positifs (détections erronées) pourront être repérés lors du traçage opéré par l'utilisateur. Les classes non-pertinentes ont en outre été retirées [3].

Le résultat est visible à la FIG. 5.3 qui regroupe 3 images d'une même séquence. Les labels affichés sont aussi enregistrés dans des fichiers `.txt` et seront adjoints au jeu de données sous la forme (x, y, w, h, p, c).

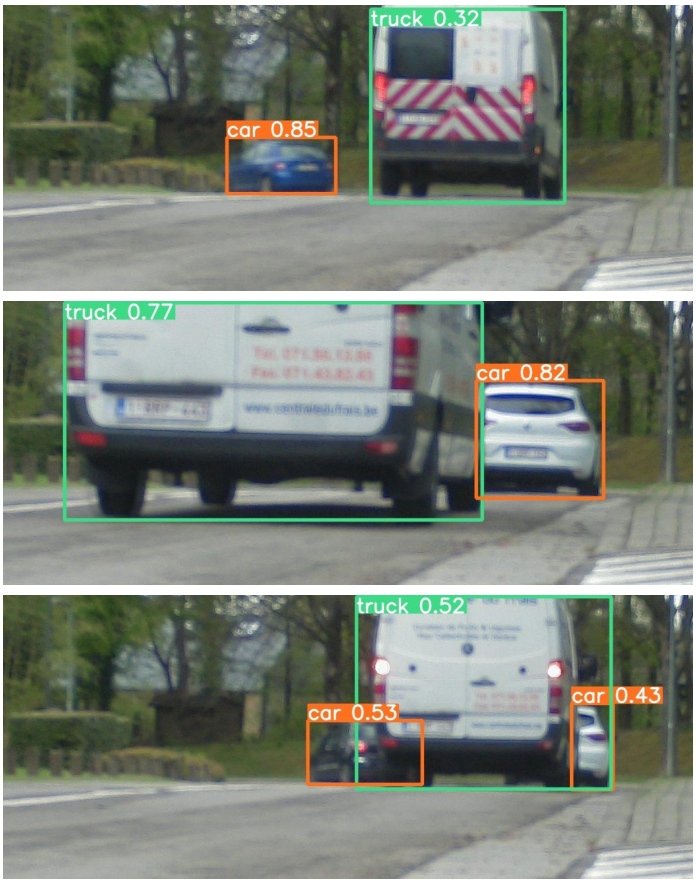

FIGURE 5.3 – Résultat de la détection et classification de l'algorithme YOLO dans sa version `YOLOV5n` proposée par `Ultralytics`. Seules les détections dont le degré de certitude est supérieur à 0.25 sont conservées.

3. Le réseau a été entraîné sur le jeu de données *Common Objects in Context* (COCO) qui contient des classes variées, comme *pizza*, *éléphant* ou *bateau*, qui ne sont pas pertinentes.

5.2 Extraction du contenu des graphes Doppler

5.2.1 Constant False Alarm Rate (CFAR)

Constant False Alarm Rate (CFAR) est un algorithme adaptatif pour le repérage de pics au sein de signaux dont l'usage est régulier dans la littérature (par ex. [58], [59]). Dans sa version la plus simple, le point du graphe $G(v, R)$ est comparé à la moyenne λ des cellules avoisinantes. On considère qu'un pic se trouve en (v, R) si $G(v, R) \geq T\lambda$ où T est un seuil fixé. Pour être plus résistant aux variations du bruit, une variante a été implémentée : *Greatest-Of* CFAR (GO-CFAR). La valeur $G(v, R)$ est cette fois comparée aux moyennes locales des points se trouvant à gauche (λ_{G}), à droite (λ_{D}), au-dessus (λ_{H}) et en-dessous (λ_{B}) du point considéré. Un pic est identifié si $G(v, R) \geq T \max(\lambda_{\mathrm{G}}, \lambda_{\mathrm{D}}, \lambda_{\mathrm{H}}, \lambda_{\mathrm{B}})$.

FIGURE 5.4 – Noyaux employés pour le calcul des moyennes λ_{G}, λ_{D}, λ_{H}, λ_{B}.

Chaque moyenne peut être calculée efficacement par convolution du graphe G avec un noyau h_i tel que ceux représentés à la FIG. 5.4. Les valeurs non-nulles de ces noyaux s'étendent sur une bande de largeur m à une distance allant de k à l du point central : elles valent $(m(l-k))^{-1}$. On applique ensuite le critère susmentionné avec $\lambda_{\mathrm{G}} = G \circledast h_{\mathrm{G}}$, …, $\lambda_{\mathrm{B}} = G \circledast h_{\mathrm{B}}$. Les valeurs de k, l, m, T ont été trouvées expérimentalement grâce aux données récoltées sur le terrain.

5.2.2 Corrélation

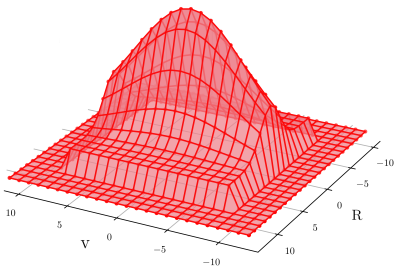

FIGURE 5.5 – Noyau h exploité pour la corrélation.

Pour éviter les 4 convolutions de CFAR, une seconde méthode a été mise sur pied. Elle est motivée par le constat que l'allure des pics ne varie généralement pas beaucoup et s'appuie sur la corrélation entre le voisinage de chaque point (v, R) avec un noyau h dont l'aspect rappelle celui des maxima à détecter, visible à la FIG. 5.5. Il favorise les amas de points dont l'allongement est plus important selon v (spectre micro-Doppler). La partie extérieure est responsable de soustraire la moyenne des cellules avoisinantes (ses valeurs sont négatives). Elle est normalisée à 1 en valeur absolue, comme la partie centrale (de sorte qu'une zone plate du graphe fournisse 0 en sortie du filtre). Puisque le noyau est symétrique selon v et R, la corrélation est équivalente à une convolution, qui est alors comparée à un seuil.

Un filtre gaussien a été appliqué en amont pour réduire l'impact du bruit. Celui-ci est de la forme $h'(x,y) = \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)/\tilde{h}$ où $\tilde{h}$ est tel que $\sum_{x,y} h'(x,y) = 1$. L'expérience a montré que $\sigma = 3$ était un bon compromis entre réduction du bruit et altération des pics. Justifiée par le caractère associatif de la convolution, la combinaison du filtre gaussien h' avec le noyau h de la corrélation a permis de réduire de moitié le temps d'exécution (23.0 → 10.9 ms, mesuré sur 4000 iter.).

5.2.3 Validation

Les résultats des deux méthodes sont très satisfaisants. Deux exemples sont affichés ci-dessous (FIG. 5.6). Les seuils ont été déterminés expérimentalement. Ils sont proportionnels à la valeur maximale de sorte à s'adapter aux variations d'intensité d'un graphe à l'autre. Le lecteur est invité à consulter l'annexe A.6 : les séquences de la FIG. A.16 illustrent le suivi que permettent les algorithmes lors du passage de véhicules. L'outil est capable de fournir jusqu'à plus de 85 paires *avec détection* pour un véhicule circulant à 30 km/h (7.3 s dans le champ de vision).

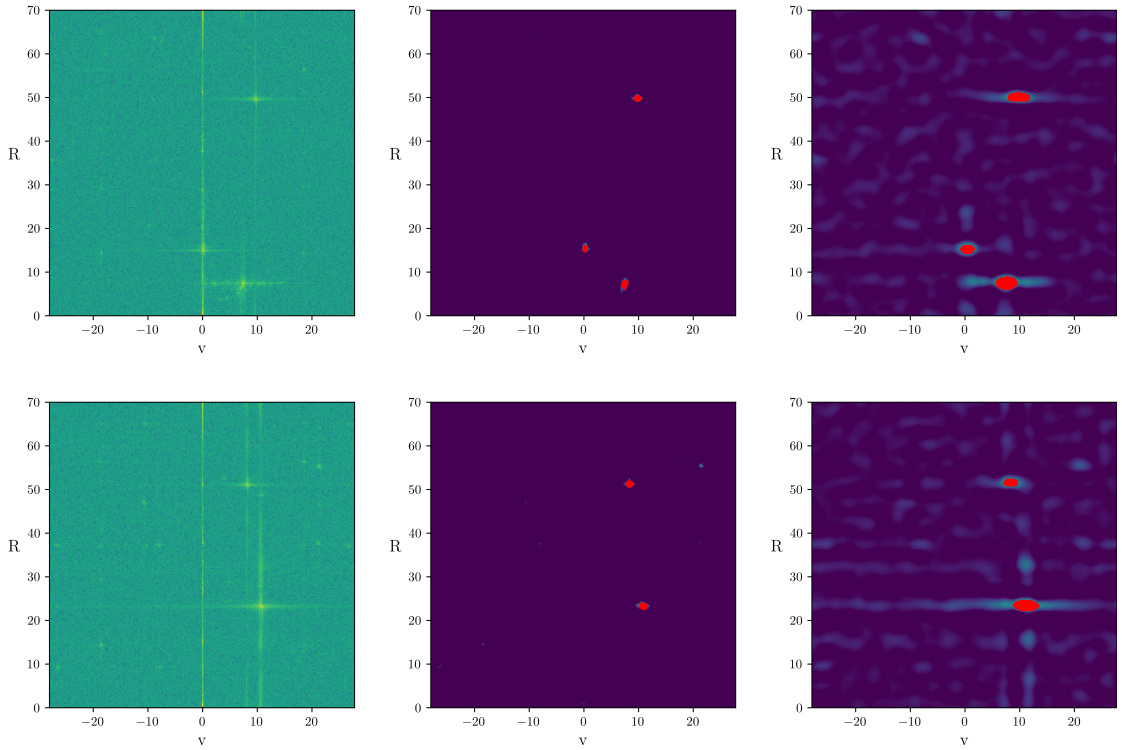

FIGURE 5.6 – Résultats de l'extraction du contenu des graphes Doppler. **Gauche :** graphe Doppler (brut). **Milieu :** méth. CFAR. **Droite :** méth. de la corrélation. Les cibles détectées sont marquées par les points rouges.

Une évaluation a été menée dans le but de comparer objectivement les deux méthodes. Les séquences étudiées comptabilisent 1418 apparitions de véhicules réparties parmi pas moins de 1368 graphes. Les résultats sont disponibles en annexe (cf. TABLE A.6) et synthétisés dans la TABLE 5.1. Le taux de rappel (pourcentage d'apparitions détectées par rapport au nombre total d'apparitions) est très bon pour les deux méthodes, celui de CFAR étant très légèrement supérieur. Quant à la précision (pourcentage de détections correctes), elle dépasse 95% pour les deux algorithmes (CFAR : 100%, CORR : $100-2.54 = 97.46\%$). Une série de 20 tests (3600 itérations), indépendante de l'expérience ci-dessus, confirme la différence de temps d'exécution : la méthode CORR. (10.16 ms/iter.) est 5.25 fois plus rapide que CFAR (53.42 ms/iter.). On peut ainsi conclure que les deux méthodes fournissent des résultats très satisfaisants et souligner les performances remarquables de CFAR lors de ces tests.

APPR	1418	100%	FP	
CFAR	1299	91.61%	0	0%
CORR	1264	89.14%	33	2.54%

TABLE 5.1 – Synthèse des résultats. **APPR**=nbr. d'appar. (nbr. de graphes sur lequel un ou plusieurs véh. sont visibles), **CFAR**=nbr. d'appar. détectées par l'alg. CFAR, **CORR**=nbr. d'appar. détectées par la méth. de la corrélation, **FP**=faux positifs (détections erronées).

Conformément à l'analyse ci-dessus, les performances de CFAR devancent (de peu) celles de la corrélation, tandis que l'usage de cette dernière est nettement plus rapide. En fonction de ses objectifs, l'utilisateur de l'outil peut choisir d'employer l'une ou l'autre méthode. Pour favoriser la précision et le taux de rappel, CFAR est établi comme méthode par défaut dans ce travail.

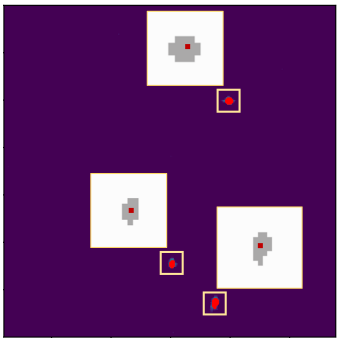

FIGURE 5.7 – Cibles isolées et loc. en leur point séculaire respectif.

La séparation des cibles est assurée par un algorithme conçu à cet égard. Partant d'un point unique, il applique des dilatations binaires [4] successives au nuage reçu de CFAR/CORR jusqu'à ce que le résultat ne varie plus : il a ainsi rassemblé tous les points associés à une même cible (supposés appartenir au même ensemble connexe). Le point spéculaire (point, parmi ceux considérés, dont l'intensité sur le graphe Doppler est maximale) est déterminé et enregistré comme position de la cible. Si des pics détectés par CFAR/CORR n'ont pas encore été associés à une cible, l'algorithme s'exécute à nouveau sur base d'un des points non encore associés.

4. Opération appartenant au domaine des morphologies mathématiques. Employée avec un noyau carré 3×3, elle revient à convertir en `1` tous les `0` adjacents à un `1`.

5.3 Création du jeu de données

5.3.1 Implémentation logicielle

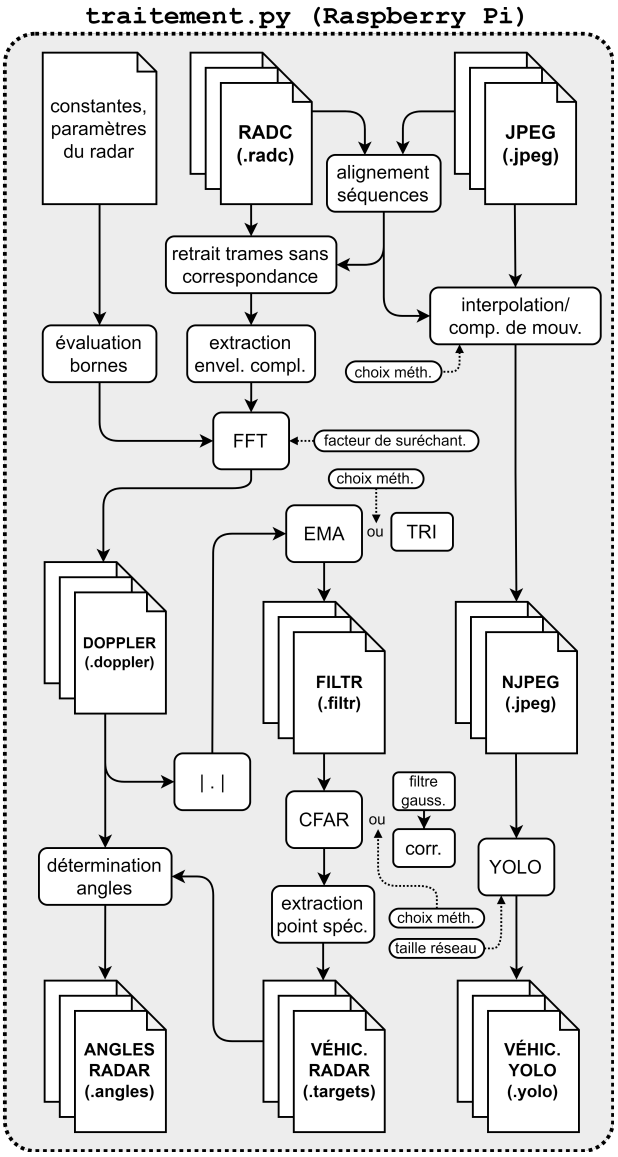

FIGURE 5.8 – Structure de `traitement.py`. Ce programme assure la création du jeu de données sur base des données des capteurs récoltées par `acquisition.c`.

La FIG. 5.8 montre comment les différents sujets abordés aux chapitres 3, 4 et 5 ont été intégrés et interagissent au sein du programme `traitement.py`. Celui-ci génère le jeu de données, ou *data set*[5], sur base des mesures récoltées par `acquisition.c`.

Ce data set est constitué des graphes Doppler (`.doppler`), des images correspondantes issues de l'interpolation ou de la compensation de mouvements (`.jpeg`), ainsi que des fichiers `.targets`, `.angles` et `.yolo` qui reprennent la position des cibles dans les graphes et images. Pour faciliter l'import et l'export des données, celles-ci sont directement enregistrées sous forme d'objets Python. Le programme `convert.py` est prévu pour combiner les fichiers `.targets`, `.angles` et `.yolo` en format `.csv` dans le cas où l'utilisateur souhaiterait employer un autre langage de programmation.

Sur la FIG. 5.8, les flèches en pointillés marquent les paramètres pour lesquels l'utilisateur peut faire un choix.

5. Dénomination anglaise habituellement employée en français pour désigner un ensemble de données servant à leur étude, ou l'entraînement d'algorithmes.

5.3.2 Parallélisme

Pour tirer profit de chaque cœur du Raspberry Pi et faire en sorte qu'ils puissent tous participer à l'exécution, il est important de paralléliser le code. Sans cela, un seul cœur exécute le code séquentiellement, ligne par ligne, comme expliqué dans la section 2.2.1. Le nombre de cœurs varie d'un micro-contrôleur à l'autre. Il y en a 4 dans le Raspberry Pi.

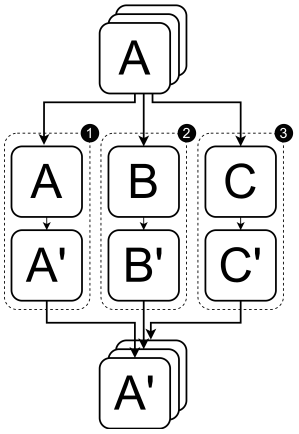

FIGURE 5.9 – Parallélisme : la tâche $i \to i'$ est exécutée sur A, B et C simultanément.

Il s'agit cette fois-ci d'une manière plus intuitive de parallélisation : ce sont les tâches répétitives et invariantes qui sont réalisées simultanément (FIG. 5.9). Celles-ci doivent être insensibles à l'ordre dans lequel leur contenu est traité (itérations indépendantes). La plupart des étapes respectent cette condition : la FFT, l'interpolation, la détermination des angles, CFAR (ou la corrélation), l'extraction du point spéculaire et YOLO. Certaines actions ne peuvent pas être parallélisées, car l'exécution sur un élément dépend des précédents. C'est le cas du filtre EMA, où l'arrière-plan est adapté à chaque itération pour les suivantes.

5.3.3 Temps d'exécution

Le traitement peut être réalisé a posteriori dans la plupart des applications, ce qui a pour effet de relaxer la contrainte de temps. Puisque le projet vise à s'étendre à des applications en temps-réel à long terme, voici un aperçu des performances du programme.

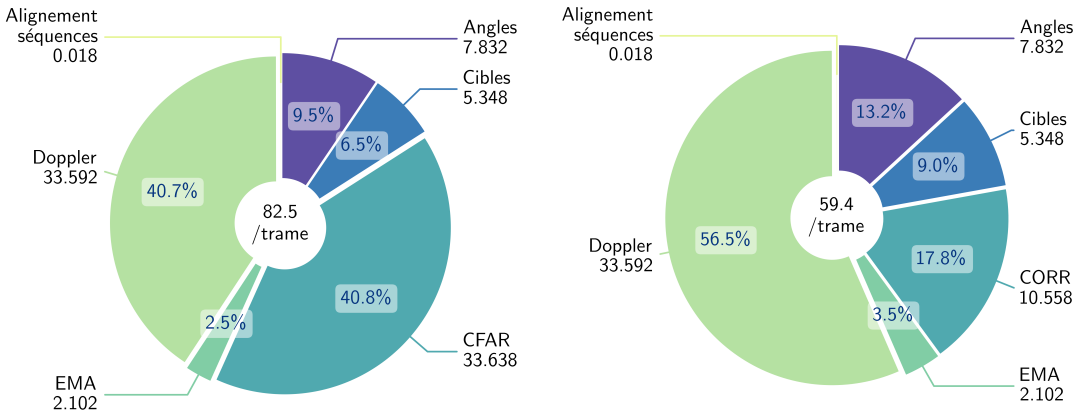

FIGURE 5.10 – Temps de références pour le traitement d'une trame avec CFAR (gauche) ou la méthode de la corrélation (droite). Temps exprimés en ms.

La FIG. 5.10 présente la répartition du temps d'exécution parmi les différentes étapes, où «Doppler» réunit l'extraction de l'enveloppe complexe, l'application de la FFT et le calcul de la norme pour chacune des trois antennes, ce qui explique son incidence. L'étape «cibles» correspond à l'algorithme présenté en fin de section 5.2.3. Avec l'utilisation de CFAR, une trame est traitée toutes les 82.5 ms, soit une vitesse de 12.12 fps. Celle-ci atteint même 16.8 fps lorsque l'utilisateur opte pour la corrélation (à la place de CFAR), à raison de 59.4 ms par trame. Ces résultats sont très encourageants et dépassent le taux d'acquisition de 12 fps.

La FIG. 5.11 reprend les temps de référence du traitement des images. À l'exception de l'alignement des séquences, qui détermine les images à utiliser pour l'interpolation et a lieu tout début du processus, toute la transformation des images est indépendante de celle des trames et peut être exécutée en parallèle. Avec la mise en correspondance, on obtient 7.1 fps. L'interpolation linéaire et de Lagrange nécessiteraient quant à elles 104.0 et 214.7 ms. Il n'a pas été possible d'apprécier le temps d'exécution de la compensation de mouvements (pas d'accès au cœur de `FFmpeg`) ; on peut cependant l'estimer à 6 fps, soit 3.3 fps pour la chaîne complète. Le traitement des images, et plus particulièrement la compensation de mouvements, sont significativement plus longues que celui des trames. Ceci s'explique par les calculs exhaustifs opérés pour la compensation de mouvements, et l'emploi d'un réseau de neurones au sein de YOLO.

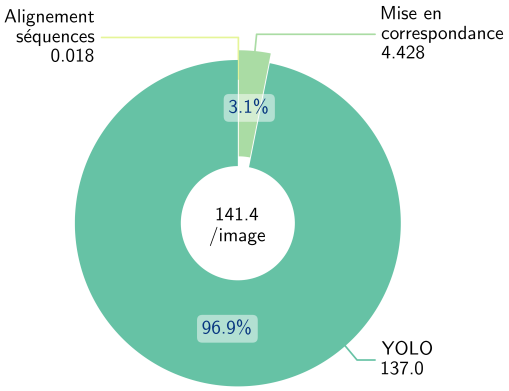

FIGURE 5.11 – Temps de référence pour le traitement d'une image. Temps exprimés en ms.

Pour conclure brièvement, voici deux méthodes prometteuses qui devraient permettre d'atteindre 12 fps (objectif déjà accompli en ce qui concerne les trames) pour une exécution temps-réel :

- ▷ implémentation d'une compensation de mouvements dédiée et optimisée pour cette application ou emploi de celle fournie par la compression H.264 (évite les calculs exhaustifs), comme proposé en 4.2.3.2 ;
- ▷ envoi de données à un ordinateur ou serveur distant : avec un nombre de cœurs plus important, il pourra exécuter toutes les tâches reprises dans les FIG. 5.10 et 5.11 simultanément. Puisque le débit est limité par l'étape la plus lente, il sera judicieux de diviser les plus longues – on pense à «Doppler» et CFAR en premier lieu –.

5.3.4 Résultat

La FIG. 5.12 résume la plupart des aspects évoqués dans les chapitres 3 à 5 et est un exemple de ce que peut produire le programme élaboré pour ce mémoire [6]. Toutes les informations visibles sur l'image et les graphiques sont enregistrés dans la base de données `.csv` ou sous forme d'objets Python (choix de l'utilisateur).

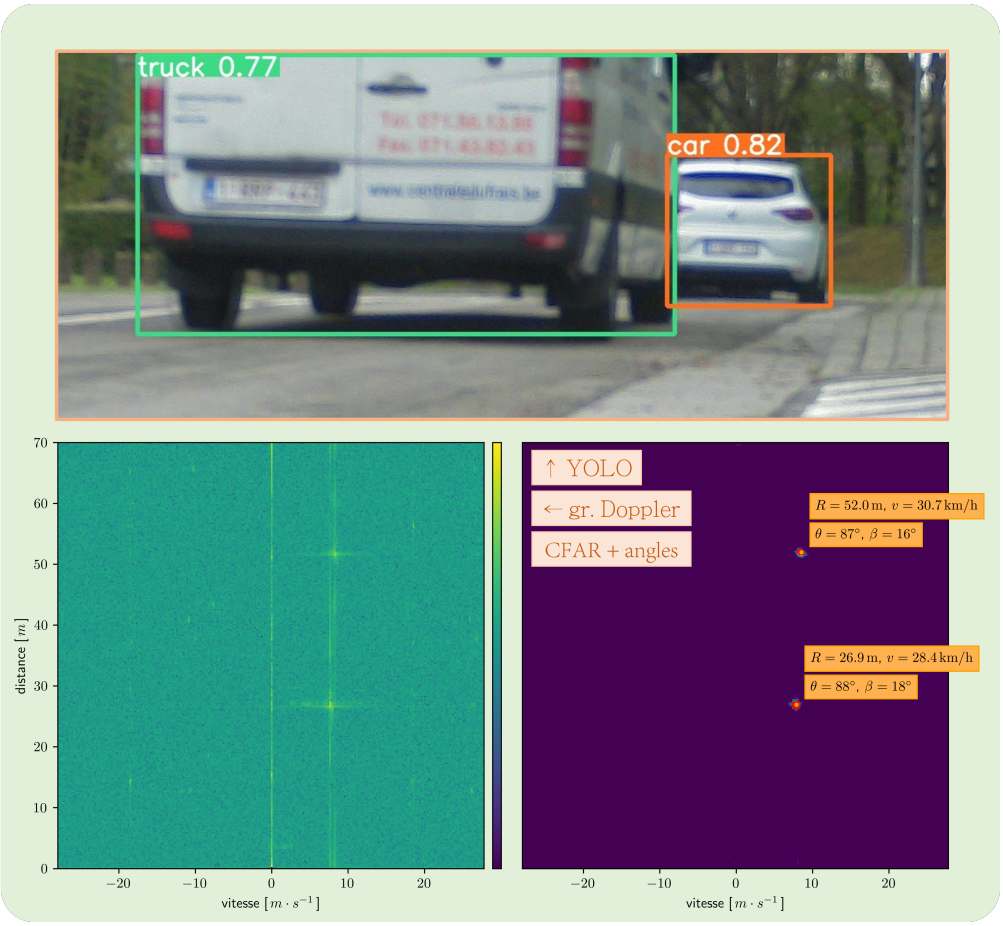

FIGURE 5.12 – Illustration du résultat final. Elle résume la plupart des aspects évoqués dans les chapitres 3 à 5 et repose bien sûr sur l'acquisition du chapitre 2.

Les essais réalisés sur le terrain ont montré que la détection des angles souffrait du bruit. Elle y est en effet particulièrement sensible au vu des expressions exposées dans la section 3.2.2.3. Le temps manqua à la quantification de ces imprécisions, qui sont de l'ordre de 1 à 2 °.

6. L'image et les graphes sont issus du programme. Seuls le fond pâle et les labels «YOLO», «gr. Doppler» et «CFAR + angles» ont été manuellement insérés.

Contribution

Ce travail a permis l'élaboration d'un outil qui peut soutenir la recherche et participer à son développement. Comme souligné dans la revue de littérature rapportée dans l'introduction, l'utilisation de capteurs bimodaux est en effet peu répandue : seuls quelques articles relatent l'association conjointe [7] d'un radar et d'une caméra. Les points susceptibles de contribuer au projet et stimuler l'état de l'art sont succinctement énoncés ci-dessous.

Capteur bimodal en tant qu'outil

Le capteur a été développé dans le but de répondre au besoin d'une observation performante et fiable des routes. Il peut bien sûr servir à diverses applications.

▷ Taux d'acquisition

Le programme est capable de produire 12 paires image–graphe par seconde en cas d'utilisation en temps-réel. À raison de 11.7 fps (Raspberry Pi, enregistrement activé), il peut amasser 42 120 paires en 1h. Un véhicule roulant à 50 km/h est ainsi photographié à 61 reprises sur 70 m. À titre comparatif : les algorithmes entraînés l'année dernière, dont la base de données comprenait 10 000 à 15 000 captures, ne disposaient que de 5 à 6 apparitions par véhicule.

▷ Équilibre entre les images et les graphes

L'entraînement des algorithmes de Gauthier Rotsart de Hertaing et Kevin de Sousa souffrait d'un déséquilibre entre les sources : la corruption des trames réduisait leur nombre. En évitant les pertes et en formant des paires, le capteur s'assure que l'entraînement repose sur autant de graphes que d'images.

7. Sans subordination de l'un à l'autre ; capteurs exploités sur un pied d'égalité.

▷ Suivi efficace des véhicules

Des tests réalisés sur 25 véhicules révèlent que près de 90% des apparitions sont détectées sur les graphes du radar, montrant par la même occasion qu'un véhicule circulant à 30 km/h peut être détecté sur plus de 80 échantillons. Cela constitue une base solide sur laquelle peuvent être appliqués des algorithmes de suivi de trajectoires. Très peu de faux positifs perturberont le traçage (FP : CFAR $\approx 0\%$, CORR $\approx 2.5\%$).

▷ Code clair et modulable

Le programme a été conçu pour être transmis à de potentiels successeurs sur ce projet, ou adapté pour d'autres applications. Il a de ce fait été implémenté de façon claire et structurée, facilitant l'insertion de blocs supplémentaires ou la modification de ceux existants.

Emploi du capteur pour l'étude des réseaux neuronaux

L'outil va permettre d'entraîner des réseaux de neurones dès cet été. Des séances de capture sont d'ores et déjà prévues durant les mois de juillet et août. Fournies à Gauthier Rotsart de Hertaing, les bases de données permettront d'étudier le comportement des algorithmes (MLP et architecture Transformer principalement) et observer l'incidence de plusieurs paramètres.

▷ Taille du jeu d'entraînement $\leftrightarrow$ Performances

Une base de données plus importante peut fortement contribuer à l'accroissement des performances des algorithmes (en se méfiant toutefois du sur–apprentissage). Grâce au dispositif, beaucoup de données peuvent être amassées en un temps raisonnable, ce qui pourrait servir à mettre en lumière l'impact de la taille du jeu d'entraînement sur la précision.

▷ Contexte : environnement, lieu, angle $\leftrightarrow$ Importance respective des graphes (radar) et des images (caméra)

Bien que complexe, l'étude des coefficients internes peut montrer de quelle manière et dans quelle proportion les réseaux utilisent chaque type de données pour l'analyse et la prédiction des trajectoires. Cette question peut être élargie dans des conditions environnementales différentes (moment de la journée, conditions météorologiques), dans plusieurs situations (carrefour, route rectiligne, parking) et sous des angles variés (dispositif proche du sol, en hauteur, embarqué à bord d'une voiture).

Un article est planifié à ce propos avec Gauthier Rotsart de Hertaing pour octobre 2022, sous la supervision des Prof. Craeye, Christophe et Macq, Benoît.

Pistes d'amélioration

Cette section regroupe les pistes d'amélioration soulevées dans les chapitres 1 à 5. Elles ont pour but de poursuivre le travail accompli ou proposer des alternatives intéressantes, qui pourraient, à terme, rendre le programme plus efficace et flexible. L'aspect temps-réel est abordé en premier lieu, suivi d'une méthode d'ajustement au niveau software. Celle-ci fut plusieurs fois discutée mais malheureusement mise de côté, faute de relation entre les points des graphes et les pixels des images.

Aspect temps-réel

▷ Implémentation complète en `C`

Jusqu'à présent, seul le programme d'acquisition est élaboré en `C`. Ce langage bas niveau offre des performances près de 100 fois supérieures à celles de Python. Retranscrire `traitement.py` en `C` peut être judicieux. Attention tout de même au temps de développement, `C` requiert une certaine expérience.

▷ Compensation de mouvements optimisée

Développée en 4.2.3, la compensation de mouvements fonctionne bien. Il serait toutefois plus efficace d'en produire une version dédiée et optimisée à cette application. Celle-ci tirerait idéalement profit de la compression H.264 automatiquement effectuée par la caméra, cf. 4.2.3.2.

▷ Envoi de données à un serveur distant

Avec un nombre de cœurs plus important, un serveur pourra exécuter toutes les étapes du traitement simultanément, tant pour les trames que pour les images. Puisque le débit est limité par l'étape la plus lente, il sera pertinent de diviser les plus longues, comme celles baptisées «Doppler» et CFAR sur les FIG. 5.10 et 5.11.

Ajustement software

Il n'est pas aisé d'évaluer expérimentalement la latence (délai entre la capture et l'insertion de l'estampille). Celle-ci ne dure en effet que quelques centaines de micro-secondes. Il faudrait pour cela développer un appareil capable d'induire un changement significatif – clairement visible sur les images et graphes Doppler – et d'enregistrer le moment exact de ce dernier avec une précision du même ordre de grandeur. Il est en revanche envisageable d'évaluer et corriger cette latence a posteriori. Trois étapes sont proposées.

▷ Correspondance entre les points

La première étape consiste à définir la bijection liant les points des graphes Doppler aux pixels des images. Cette étape, essentielle, n'est pas triviale : elle nécessite une étude approfondie du modèle de la caméra. Un modèle a déjà été réalisé par l'équipe «Antennes».

▷ Métrique : Information Mutuelle (IM)

L'information mutuelle fut introduite en tant que métrique par Maes, Frederik et al. en 1996 [60]. Elle s'applique à des supports de nature très différente, justifiant son emploi pour les images et graphes Doppler. L'information mutuelle entre les sources A et B s'établit comme suit :

$$I(A,B) = \sum_{a,b} p_{AB}(a,b) \log \frac{p_{AB}(a,b)}{p_A(a)p_B(b)}$$

Elle s'appuie sur la fonction de densité de probabilité conjointe $p_{AB}(a,b)$, ce qui explique le besoin de connaître la correspondance formulée ci-dessus.

▷ Algorithme : Dynamic Time Warping (DTW)

Présentée par Berndt D. et Clifford J. en 1994 [61], l'algorithme *Dynamic Time Warping* assure l'alignement de séquences semblables mais dont l'échelle de temps aurait été modifiée (par ex. élongation, compression). Il repose sur la minimisation de la distance entre les points et est couramment mis à contribution pour la reconnaissance vocale, afin d'identifier les mots malgré les écarts de ton et de rythme. Associé à l'information mutuelle comme métrique, il indiquerait si un ajustement des flux est à réaliser, et lequel.

Conclusion

Après une courte description du matériel (chapitre 1), l'acquisition marqua le début de l'élaboration du programme (chapitre 2). Celle-ci répond au besoin de données fiables constaté par Kevin de Sousa et Gauthier Rotsart de Hertaing l'année dernière. Elle assure la récolte fluide et stable des mesures, ainsi que l'insertion des estampilles indispensables pour la synchronisation. Bien que sa réalisation fut principalement rythmée par les longues semaines de compréhension des protocoles et de développement, elle n'en demeure pas moins riche pour le lecteur curieux, mettant en avant la notion de parallélisme, si chère aux yeux des informaticiens, et usant de protocoles astucieux. La coordination entre les différents programmes, résumée à la FIG. 2.2, est le fruit de plusieurs mois de travail et constitue, non sans fierté, une embase solide pour les chapitres suivants.

Une fois les données récoltées se manifesta la nécessité de les traiter afin, à terme, de constituer les jeux et d'étudier l'impact de divers paramètres sur les performances des algorithmes. En premier lieu furent dérivés, à partir des mesures brutes du radar, les graphes Doppler (chapitre 3). Ils furent ensuite filtrés, afin de retirer les éléments constants du bruit qu'ils contenaient, et les images intermédiaires générées : plusieurs filtres et méthodes ont été étudiés à cet égard (chapitre 4). Il s'avéra que le filtre EMA présentait l'avantage d'être plus rapide tout en maintenant un effet passe-bas suffisant pour les graphes, et que l'estimation de mouvements produisait un meilleur résultat pour les images

Le chapitre 5 présente finalement l'aboutissement de cette contribution : l'extraction du contenu des graphes et images et la formation de la base de données. YOLO fut l'occasion d'introduire un réseau de neurones au sein de l'ouvrage et assure la détection au sein des images. En ce qui concerne les trames, la comparaison de CFAR et d'une approche originale alliant filtre gaussien et corrélation a révélé de très bons résultats pour chaque méthode. L'emploi de CFAR, plus précis, ou

de la corrélation, plus rapide, est laissée au choix de l'utilisateur. Une fois les pics identifiés, les cibles sont isolées les unes des autres par l'application de dilatations binaires successives. La position de chaque objet est finalement enregistrée, supposée correspondre au point spéculaire.

Ce manuscrit a mis en lumière les aspects fondamentaux que nécessite la création d'un programme d'acquisition et de traitement pour un capteur bimodal. L'outil, qui profita de l'apport de nombreux domaines – programmation haut et bas niveaux, traitement du signal et de l'image, algorithmes variés et introduction à l'intelligence artificielle–, fonctionne bien et a d'ores et déjà permis l'enregistrement de deux prises de vues. D'une durée de quelques minutes chacune, elles avaient comme but principal d'en vérifier le bon fonctionnement. Elles ont permis la détection de quelques erreurs d'inattention aussitôt corrigées et servirent de support pour les essais/validations des chapitres 2 à 5. Des séances supplémentaires plus longues sont prévues dans le courant du mois de juillet, comme convenu avec les Prof. Craeye, Christophe et Macq, Benoît. Les sets ainsi générés entraîneront, entre autres, les réseaux de Gauthier Rotsart de Hertaing, et mèneront à la rédaction d'un article planifié pour octobre 2022.

Il réside pourtant au cœur de l'homme une attirance irrésistible pour ce qu'il ne détient, et il lui appartient de reconnaître avec humilité les limites de sa création. C'est pourquoi furent avancés dans la section précédente quelques points d'amélioration, qui visent à étendre les capacités de l'outil, le rendant plus efficace pour les tâches dont il est déjà capable et le perfectionnant par un ajustement software supplémentaire. Ces pistes, motivantes et prometteuses, furent discutées oralement, mais il fallut y renoncer en raison du temps restreint.

De cette belle entreprise furent de toute évidence les chapitres 2, 3 et 5 les plus bénéfiques et passionnants. Merci à tous ceux qui y ont contribué.

Annexe

A.1 Spécifications techniques de la caméra, du radar et du Raspberry Pi

Caméra ELP 1/3" CMOS AR0330		
Résolution	320 × 180, 320 × 240, 640 × 360 640 × 480, 1280 × 720, 1920 × 1080	-
Format (vidéo)	YUYV, MJPEG, H.264	-
Cadence	5, 10, 20, 25, 30	fps
Tension d'alimentation	5	V
Courant d'alimentation	150	mA
Interface	USB 2.0 HS/FS	-
Protocole de contrôle	UVC (USB Video Class)	-
Angle d'ouverture	30	°
Lentille	12	mm
Distance focale, EMRP	5695.8, 0.78	pixels
Attributs	Microphone intégré, exposition automatique	-
Paramètres ajustables	Luminosité, contraste, couleur, saturation, définition, gamma, balance des blancs	-
Dimensions	38 × 38 × 32	mm^3
Prix	45	€

TABLE A.2 – Spécifications techniques de la caméra. Les paramètres de distance focale et d'erreur moyenne de re-projection (EMRP) ont été calculés par S. Cardinael et A. Malcourant sur base d'une calibration géométrique [62]. Adapté de [30][62].

Radar K-MD2-RFB-01F-01		
Résolution	256×256	$N_\mathrm{s} \times N_\mathrm{r}$
Type de données	Données brutes (échant. du signal démodulé), graphique distance-vitesse (Doppler), cibles détectées non-filtrées, cibles traquées	-
Cadence	20	fps
Tension d'alimentation	10−16 (nominal : 12)	V
Courant d'alimentation	600	mA
Interfaces	Connexion Serial (3.3 V, 115.2 kbps), Ethernet (10−1000 Mbps)	-
Protocole de contrôle	Envoi de séquences spécifiques (cf. fiche technique)	-
Nombre d'antennes émettrices, réceptrices	1, 3	-
Polarisation	Verticale	-
Modulation	FMCW	-
Gamme de fréquence	23.800−24.800	GHz
Ouverture angulaire (-3dB) élévation, azimut	±10, ±20	deg.
Résolution angulaire	0.1	deg.
Dimensions	$120 \times 72 \times 15$	mm^3
Prix	1336	€

TABLE A.3 – Spécifications techniques du radar. N_s et N_r représentent le nombre d'échantillons par rampe et le nombre de rampes par trame, respectivement. *bps* signifie *bits par seconde*. Des informations plus détaillées sont disponibles dans la fiche technique [35].

RASPBERRY PI 4B (4 GB)		
Processeur	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz	-
Mémoire vive (RAM)	4GB LPDDR4 with on-die ECC	-
Connectivité	2.4/5.0 GHz IEEE 802.11b/g/n/ac wireless, LAN, Bluetooth 5.0, BLE, Gigabit Ethernet, 2 × USB 3.0, 2 × USB 2.0, 40-pin GPIO header	-
Vidéo et son	2 × micro HDMI ports (up to 4Kp60 supported), 2-lane MIPI DSI display port, 2-lane MIPI CSI camera port, 4-pole stereo audio and composite video port	-
Multimedia	H.265 (4Kp60 decode), H.264 (1080p60 decode, 1080p30 encode), OpenGL ES, 3.0 graphics	-
SD card support	Oui	-
Alimentation	5V DC via USB-C connector (minimum 3A), 5V DC via GPIO header (minimum 3A), Power over Ethernet (PoE)–enabled (requires separate PoE HAT)	-
Dimensions	32 × 68 × 98	mm^3
Prix	78.50 (alimentation, étui, ventilateur, câbles, dissipateurs thermiques compris)	€

TABLE A.4 – Spécifications techniques du Raspberry Pi 4B, avec une mémoire vive de 4 GB (1, 2 et 8 GB aussi commercialisés). Le prix est indicatif, car il varie d'un distributeur à l'autre et dépend de la capacité de la mémoire. Adapté de [39].

A.2 Spectre d'un peigne de Dirac

Un peigne de Dirac $p(t)$ est composé d'une succession de deltas, espacés de T. Ce signal est bien sûr périodique et peut donc s'écrire sous la forme d'une série de Fourier de coefficients appelés arbitrairement a_m :

$$p(t) = \sum_{n=-\infty}^{+\infty} \delta\left(t - nT\right) = \sum_{m=-\infty}^{+\infty} a_m \, e^{j2\pi m \frac{t}{T}} \tag{6.1}$$

Pour déterminer les coefficients a_m, il est utile de remarquer que si les deux expressions de (6.1) sont égales, les opérations desquelles elles sont sujettes doivent l'être également. En particulier, pour tout l de $\mathbb{Z}$, les expressions

$$\int_{-T/2}^{T/2} \sum_{n=-\infty}^{+\infty} \delta\left(t - nT\right) \, e^{-j2\pi l \frac{t}{T}} \mathrm{d}t = \int_{-T/2}^{T/2} \delta(t) \mathrm{d}t = 1$$

et

$$\begin{aligned} \int_{-T/2}^{T/2} \sum_{m=-\infty}^{+\infty} a_m \, e^{j2\pi m \frac{t}{T}} \, e^{-j2\pi l \frac{t}{T}} \mathrm{d}t &= \sum_{m=-\infty}^{+\infty} a_m \int_{-T/2}^{T/2} e^{-j2\pi (m-l) \frac{t}{T}} \\ &= \sum_{m=-\infty}^{+\infty} a_m \, T \delta\left(m - l\right) = a_l \, T \end{aligned}$$

doivent être égales. L'orthogonalité des exponentielles complexes a été exploitée. On conclut que $a_l = 1/T \ \forall \, l \in \mathbb{Z}$ et on peut ré-écrire $p(t)$ sous la forme

$$p(t) = \frac{1}{T} \sum_{m=-\infty}^{+\infty} e^{j2\pi m \frac{t}{T}}$$

La dérivation de son spectre est à présent triviale, à savoir :

$$P(\omega) = \mathcal{F}\left(\frac{1}{T} \sum_{m=-\infty}^{+\infty} e^{j2\pi m \frac{t}{T}}\right) = \frac{1}{T} \sum_{m=-\infty}^{+\infty} \mathcal{F}\left(e^{j2\pi m \frac{t}{T}}\right) = \frac{2\pi}{T} \sum_{m=-\infty}^{+\infty} \delta\left(\omega - \frac{2\pi m}{T}\right)$$

Le spectre d'un peigne de Dirac de période T en temporel est en réalité lui-même un peigne de Dirac de période $2\pi/T$.

A.3 Déphasage entre les antennes

La position des antennes réceptrices est schématiquement représentée à la FIG. A.13, avec A à l'emplacement de l'antenne 1 prise comme référence. On sait que $\Delta x = 0.022$ m et $\Delta y = 0.040$ m. Le vecteur $\vec{v}$ indique la direction de la cible. Puisque la cible est lointaine ($R \gg \Delta x, \Delta y$), ce vecteur est identique pour toutes les antennes. On définit les longueurs d, d' comme indiqué à la FIG. A.14 avec le point I tel que $AI \perp IB$.

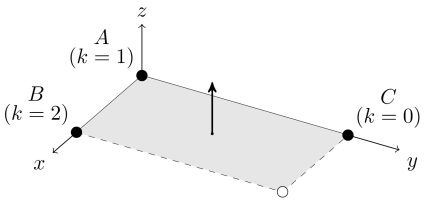

FIGURE A.13 – Placement des antennes réceptrices dans le plan xy. Le radar est orienté dans la direction z. L'angle vide du rectangle accueille l'antenne émettrice.

On commence par écrire $d' = d\sin(\theta)$ et

$$I = \begin{pmatrix} d'\cos(\eta) \\ d'\sin(\eta) \\ d\cos(\theta) \end{pmatrix} = \begin{pmatrix} d\sin(\theta)\cos(\eta) \\ d\sin(\theta)\sin(\eta) \\ d\cos(\theta) \end{pmatrix}$$

On peut ensuite déduire la distance d de la perpendicularité de AI et IB, à savoir que leur produit scalaire doit être nul :

$$\begin{aligned} AI \cdot IB &= d^2\sin^2(\theta)\cos^2(\eta) + d^2\sin^2(\theta)\sin^2(\eta) \\ &\quad - d\sin(\theta)\sin(\eta)\Delta x + d^2\cos^2(\theta) \\ &= d^2\sin^2(\theta) - d\sin(\theta)\sin(\eta)\Delta x \\ &\quad + d^2\cos^2(\theta) \\ &= d^2 - d\sin(\theta)\sin(\eta)\Delta x \\ &= d\left(d - \sin(\theta)\sin(\eta)\Delta x\right) \end{aligned}$$

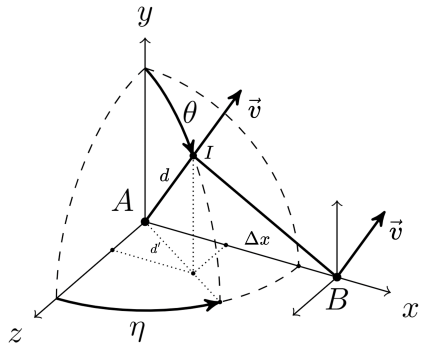

FIGURE A.14 – Géométrie étudiée. Les axes x, y, z correspondent à ceux de la FIG. A.13. Ce nouvel agencement facilite la visualisation.

dont la résolution est évidente : $d = 0$ (solution ignorée) ou $d = \sin(\theta)\sin(\eta)\Delta x$. De la même manière, on peut déduire pour l'antenne C que $d = d\cos(\theta)\Delta y$. En introduisant les variables $\gamma = \sin(\theta)\sin(\eta)$, $\xi = \cos(\theta)$ et en tenant compte de l'indice k associé à chaque antenne ($A \leftrightarrow 1$, $B \leftrightarrow 2$, $C \leftrightarrow 0$), on peut écrire $d_B = d_2 = \gamma\Delta x$ et $d_C = d_0 = \xi\Delta y$.

Enfin, il est utile de remarquer que la distance parcourue par l'onde pour atteindre A est supérieure que celle pour parvenir à B ou C. Il est pour cette raison nécessaire de considérer $\delta_0 = -d_0 = -\xi\Delta y$ et $\delta_2 = -d_2 = -\gamma\Delta x$ pour être en adéquation avec la convention prise en 3.2.2.3.

A.4 Comparaison des versions de YOLOv5

Model	size (pixels)	mAPval 0.5 :0.95	mAPval 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6	1280	55.0	72.7	3136	26.2	19.4	140.7	209.8
+ TTA	1536	55.8	72.7	-	-	-	-	-

TABLE A.5 – Comparaison des différentes versions de `YOLOv5` développé par `Ultralytics`. Extrait de [57]. Les conditions d'évaluation y sont détaillées.

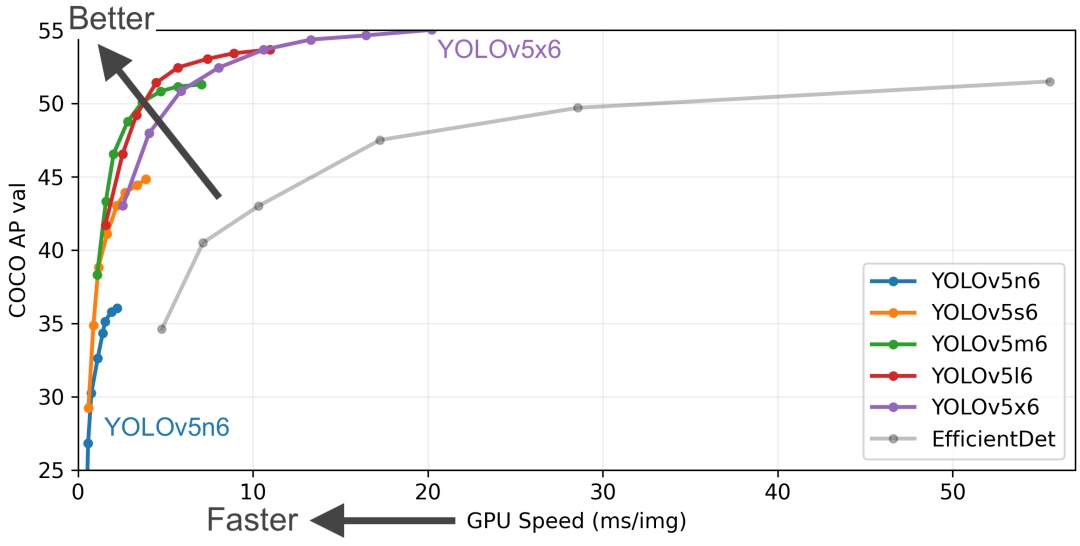

FIGURE A.15 – Comparaison des différentes versions de `YOLOv5` développé par `Ultralytics`. Extrait de [57]. Les conditions d'évaluation y sont détaillées.

A.5 Détection des véhicules (séquences)

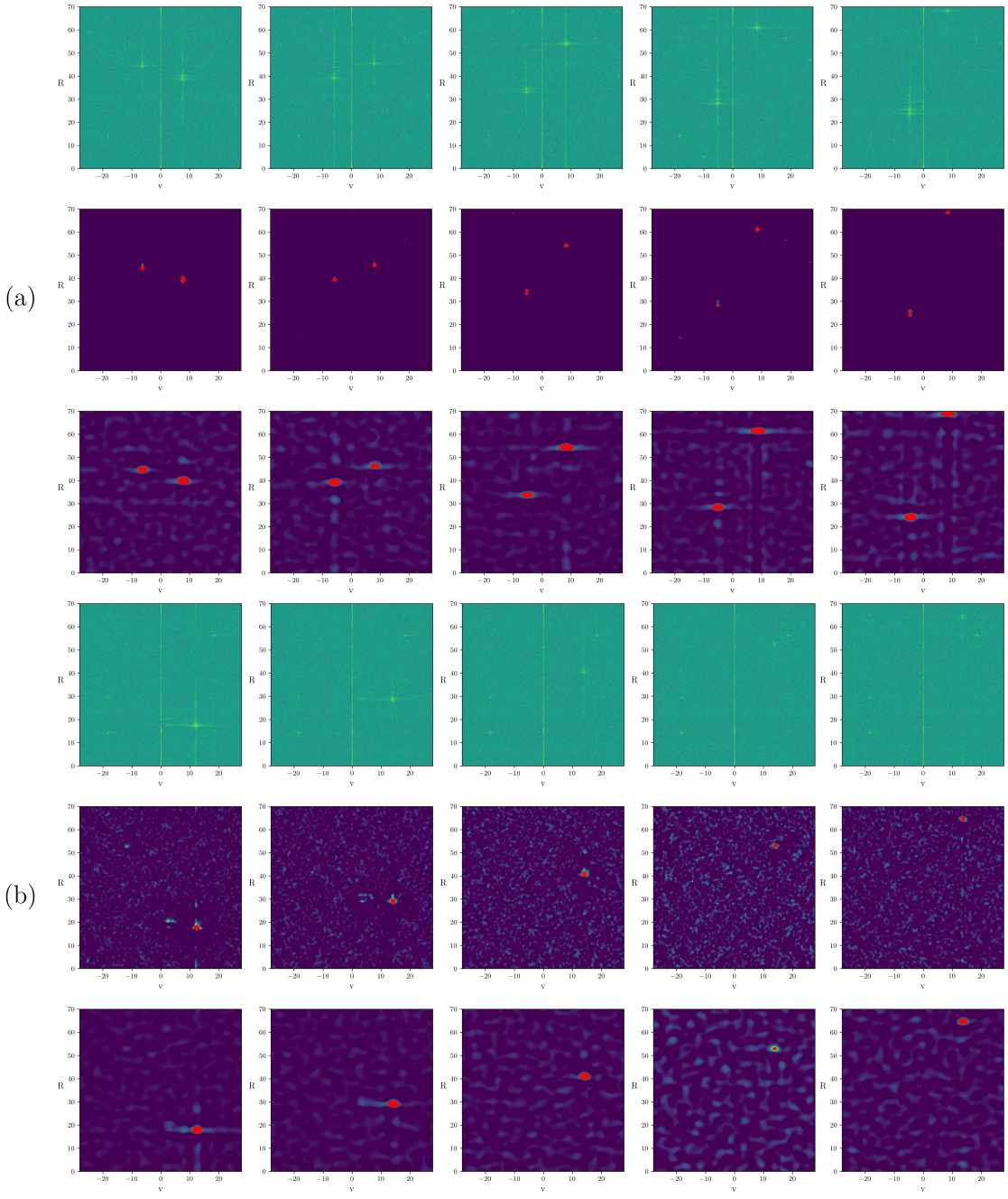

FIGURE A.16 – 2 séquences issues de prises de vue différentes. **Gauche :** graphe Doppler (brut). **Milieu :** méth. CFAR. **Droite :** méth. de la corrélation. Les cibles détectées sont marquées par les points rouges. Les séquences contiennent en réalité 50 images (1 image affichée toutes les 10 images).

A.6 Comparaison CFAR et corrélation

Résultat de la série d'évaluation effectuée dans le but de comparer l'algorithme CFAR à la méthode employant la corrélation. Les séquences étudiées comptabilisent 1418 apparitions de véhicules réparties parmi pas moins de 1368 graphes. Le pourcentage de faux positifs (FP dans la TABLE A.6) est déterminé par rapport au nombre total de détections (par ex. $16/(16+224) = 6.67\%$ pour le cas CORR du Test A). La synthèse des résultats est affichée et commentée à la section 5.2.3.

	Test A									**Test B**					
NVEH	1	2	3	4	5	**TOT.**		FP		1	2	**TOT.**		FP	
APPR	18	68	52	58	51	247	100%			96	35	131	100%		
CFAR	16	60	51	55	49	231	93.52%	0	0%	74	35	109	83.21%	0	0%
CORR	15	63	51	44	51	224	90.69%	16	6.67%	70	35	105	80.15%	0	0%

	Test C								**Test D**							
NVEH	1	2	3	4	**TOT.**		FP		1	2	3	4	**TOT.**		FP	
APPR	70	83	89	8	250	100%			57	91	68	17	233	100%		
CFAR	68	79	89	7	243	97.20%	0	0%	54	90	61	13	218	93.56%	0	0%
CORR	63	83	87	8	241	96.40%	6	2.42%	54	91	60	13	218	93.56%	3	1.36%

	Test E													
NVEH	1	2	3	4	5	6	7	8	9	10	**TOT.**		FP	
APPR	20	73	96	57	21	70	9	65	73	73	557	100%		
CFAR	13	72	96	47	18	59	8	60	62	63	498	89.41%	0	0%
CORR	10	72	96	42	15	60	8	56	57	60	476	85.46%	8	1.65%

TABLE A.6 – Évaluations effectuées dans le but de comparer l'algorithme CFAR à la méthode employant la corrélation. **NVEH**=num. du véh., **APPR**=nbr. d'apparitions (nbr. de graphes sur lequel le véh. est visible), **CFAR**=nbr. d'appar. détectées par l'alg. CFAR, **CORR**=nbr. d'appar. détectées par la méth. de la corrélation, **FP**=faux positifs (détections erronées).

Références

[1] StatBel. *Accidents de la circulation*. 2021. URL : https://statbel.fgov.be/fr/themes/mobilite/circulation/accidents-de-la-circulation#news (visité le 15/02/2022).

[2] C.-K. Sung. "Extraktion von typischen und komplexen Vorgängen aus einer langen Bildfolge einer Verkehrsszene". In : *Mustererkennung 1988*. Sous la dir. d'Horst Bunke, Olaf Kübler et Peter Stucki. Berlin, Heidelberg : Springer Berlin Heidelberg, 1988, p. 90-96. ISBN : 978-3-662-08895-1.

[3] Dieter Koller et al. "Model-based object tracking in traffic scenes". In : t. 588. Jan. 2006, p. 437-452. ISBN : 978-3-540-55426-4. DOI : 10.1007/3-540-55426-2_49.

[4] Maria Jokela, Matti Kutila et Long Le. "Road condition monitoring system based on a stereo camera". In : *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*. 2009, p. 423-428. DOI : 10.1109/ICCP.2009.5284724.

[5] Yoichiro Iwasaki, Masato Misumi et Toshiyuki Nakamiya. "Robust Vehicle Detection under Various Environmental Conditions Using an Infrared Thermal Camera and Its Application to Road Traffic Flow Monitoring". In : *Sensors (Basel, Switzerland)* 13 (juin 2013), p. 7756-73. DOI : 10.3390/s130607756.

[6] Hugh Griffiths, Peter Knott et Wolfgang Koch. "Christian Hülsmeyer : Invention and Demonstration of Radar, 1904". In : *IEEE Aerospace and Electronic Systems Magazine* 34.9 (2019), p. 56-60. DOI : 10.1109/MAES.2019.2934814.

[7] Additi Singh, Soumyasree Bera et Rabindranath Bera. "Review on Vehicular Radar for Road Safety". In : juin 2019, p. 41-47. ISBN : 978-981-10-8910-7. DOI : 10.1007/978-981-10-8911-4_5.

[8] Lee C. Potter et al. “Sparsity and Compressed Sensing in Radar Imaging”. In : *Proceedings of the IEEE* 98.6 (2010), p. 1006-1020. doi : 10.1109/JPROC.2009.2037526.

[9] Muhammad Hadi et al. “Compressive sensing applied to radar systems : an overview”. In : *Signal Image and Video Processing* (déc. 2015). doi : 10.1007/s11760-015-0824-y.

[10] A. Aubry et al. “A new radar waveform design algorithm with improved feasibility for spectral coexistence”. In : *IEEE Transactions on Aerospace and Electronic Systems* 51.2 (2015), p. 1029-1038. doi : 10.1109/TAES.2014.140093.

[11] Le Zheng et al. “Radar and Communication Coexistence : An Overview : A Review of Recent Methods”. In : *IEEE Signal Processing Magazine* 36.5 (2019), p. 85-99. doi : 10.1109/MSP.2019.2907329.

[12] J. Andrew Zhang et al. “Enabling Joint Communication and Radar Sensing in Mobile Networks -A Survey”. In : *IEEE Communications Surveys and Tutorials* PP (oct. 2021), p. 1-1. doi : 10.1109/COMST.2021.3122519.

[13] Derek Magee. “Tracking multiple vehicles using foreground, background and motion models”. In : *Image Vision Comput.* 22 (fév. 2004), p. 143-155. doi : 10.1016/S0262-8856(03)00145-8.

[14] N.K. Kanhere, S.J. Pundlik et S.T. Birchfield. “Vehicle segmentation and tracking from a low-angle off-axis camera”. In : *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05).* T. 2. 2005, 1152-1157 vol. 2. doi : 10.1109/CVPR.2005.365.

[15] M.B. van Leeuwen et F.C.A. Groen. “Vehicle detection with a mobile camera : spotting midrange, distant, and passing cars”. In : *IEEE Robotics Automation Magazine* 12.1 (2005), p. 37-43. doi : 10.1109/MRA.2005.1411417.

[16] Margrit Betke, Esin Haritaoglu et Larry Davis. “Real-time multiple vehicle detection and tracking from a moving vehicle”. In : *Mach. Vis. Appl.* 12 (août 2000), p. 69-83. doi : 10.1007/s001380050126.

[17] Shin-Ping Lin, Yuan-Hsin Chen et Bing-Fei Wu. “A Real-Time Multiple-Vehicle Detection and Tracking System with Prior Occlusion Detection and Resolution, and Prior Queue Detection and Resolution”. In : *18th International Conference on Pattern Recognition (ICPR'06).* T. 1. 2006, p. 828-831. doi : 10.1109/ICPR.2006.159.

[18] Frank Dellaert et Charles E. Thorpe. “Robust car tracking using Kalman filtering and Bayesian templates”. In : *Other Conferences.* 1998.

[19] J. J. Wu et al. “Recovery of the 3-D location and motion of a rigid object through camera image (An Extended Kalman Filter Approach)”. In : *International Journal of Computer Vision* 2 (2004), p. 373-394.

[20] Vincenzo Lippiello, Bruno Siciliano et Villani Luigi. “Adaptive extended Kalman filtering for visual motion estimation of 3D objects”. In : *Control Engineering Practice - CONTROL ENG PRACTICE* 15 (jan. 2007), p. 123-134. doi : 10.1016/j.conengprac.2006.05.006.

[21] Poonsak Sirichai et al. “Smart Car with Security Camera for Road Accidence Monitoring”. In : *Procedia Engineering* 8 (déc. 2011), p. 308-312. doi : 10.1016/j.proeng.2011.03.057.

[22] X. Clady et al. “Cars detection and tracking with a vision sensor”. In : *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683)*. 2003, p. 593-598. doi : 10.1109/IVS.2003.1212979.

[23] Fulvio Gini. “Grand Challenges in Radar Signal Processing”. In : *Frontiers in Signal Processing* 1 (2021). issn : 2673-8198. doi : 10.3389/frsip.2021.664232. url : https://www.frontiersin.org/article/10.3389/frsip.2021.664232.

[24] Ping Lang et al. “A Comprehensive Survey of Machine Learning Applied to Radar Signal Processing”. In : *ArXiv* abs/2009.13702 (2020).

[25] Felix Nobis et al. “A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection”. In : oct. 2019, p. 1-7. doi : 10.1109/SDF.2019.8916629.

[26] Ramin Nabati et Hairong Qi. “CenterFusion : Center-based Radar and Camera Fusion for 3D Object Detection”. In : *CoRR* abs/2011.04841 (2020). arXiv : 2011.04841. url : https://arxiv.org/abs/2011.04841.

[27] Vladimir Lekic et Zdenka Babic. “Automotive radar and camera fusion using Generative Adversarial Networks”. In : *Computer Vision and Image Understanding* 184 (2019), p. 1-8. issn : 1077-3142. doi : https://doi.org/10.1016/j.cviu.2019.04.002. url : https://www.sciencedirect.com/science/article/pii/S1077314219300530.

[28] Xiangmo Zhao et al. “Fusion of 3D LIDAR and Camera Data for Object Detection in Autonomous Vehicle Applications”. In : *IEEE Sensors Journal* 20.9 (2020), p. 4901-4913. doi : 10.1109/JSEN.2020.2966034.

[29] Jean Léger. “Radar target classification based on micro-Doppler signature analysis”. Prom. : Craeye, Christophe ; Vandendorpe, Luc. Mém. de mast. École polytechnique de Louvain, Université catholique de Louvain, 2016.

[30] Alexis Duflot. “Conception d’un senseur intégré multimodal pour l’observation des routes”. Prom. : Craeye, Christophe ; Macq, Benoît. Mém. de mast. École polytechnique de Louvain, Université catholique de Louvain, 2020.

[31] Raffaele Di Martino. “Modelling and simulation of the dynamic behaviour of the automobile”. Theses. Université de Haute Alsace - Mulhouse, jan. 2005. URL : https://tel.archives-ouvertes.fr/tel-00736040.

[32] Nerijus Kudarauskas. “Analysis of emergency braking of a vehicle”. In : *Transport* 22 (juill. 2007). DOI : 10.1080/16484142.2007.9638118.

[33] AliExpress. *ELP-1080P H264 Aptina AR0330 CMOS, USB CCTV hd 2.8mm usb*. URL : https://ko.aliexpress.com/item/32423025870.html (visité le 03/03/2022).

[34] RFbeam Microwave GmbH. *K-MD2 Engineering Sample*. URL : https://www.rfbeam.ch/product?id=21 (visité le 03/03/2022).

[35] *K-MD2 Engineering Sample*. Fiche technique. RFbeam Microwave GmbH. 2018. URL : https://www.rfbeam.ch/product?id=21.

[36] Wikipedia, L’encyclopédie libre. *Raspberry Pi*. URL : https://fr.wikipedia.org/wiki/Raspberry_Pi (visité le 08/03/2022).

[37] Raspberry Pi Foundation. *About us*. URL : https://www.raspberrypi.org/about/ (visité le 08/03/2022).

[38] Raspberry Pi Foundation. *Raspberry Pi 4. Your tiny, dual-display, desktop computer... and robot brains, smart home hub, media centre, networked AI core, factory controller, and much more*. URL : https://www.raspberrypi.com/products/raspberry-pi-4-model-b/?variant=raspberry-pi-4-model-b-4gb&resellerType=industry (visité le 08/03/2022).

[39] *Raspberry Pi 4 Computer Model B*. Product Brief. Raspberry Pi Trading Ltd. 2021. URL : https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf.

[40] *ALF1501D*. Caractéristiques techniques. ELC. URL : https://elc.fr/media/alf1501d.pdf.

[41] Kevin Desousa et Gauthier Rotsart de Hertaing. “Conception d’un senseur intégré multimodal pour l’observation des routes”. Prom. : Craeye, Christophe ; Macq, Benoît. Mém. de mast. École polytechnique de Louvain, Université catholique de Louvain, 2021.

[42] Eloi Guerrero-Menéndez. “Frequency-modulated continuous-wave radar in automotive applications”. In : 2018.

[43] Tú Võ Tien. “Diversité spatiale, temporelle et fréquentielle pour la mesure précise de distance et d’angle d’arrivée en ultra large bande”. Thèse de doct. Juin 2019.

[44] Bong-seok Kim et al. “A Low-Complexity FMCW Surveillance Radar Algorithm Using Two Random Beat Signals”. In : *Sensors* 19 (jan. 2019), p. 608. doi : 10.3390/s19030608.

[45] Thomas Feuillen et al. “Quantity over Quality : Dithered Quantization for Compressive Radar Systems”. In : *2019 IEEE Radar Conference (RadarConf)*. 2019, p. 1-6. doi : 10.1109/RADAR.2019.8835568.

[46] Bong-seok Kim, Sangdong Kim et Jonghun Lee. “A Novel DFT-Based DOA Estimation by a Virtual Array Extension Using Simple Multiplications for FMCW Radar”. In : *Sensors* 18 (mai 2018), p. 1560. doi : 10.3390/s18051560.

[47] Youn-Sik Son, Hyuk-Kee Sung et Seo Heo. “Automotive Frequency Modulated Continuous Wave Radar Interference Reduction Using Per-Vehicle Chirp Sequences”. In : *Sensors* 18 (août 2018), p. 2831. doi : 10.3390/s18092831.

[48] Graham Brooker. “Understanding millimetre wave FMCW radars”. In : *1st International Conference on Sensing Technology* (jan. 2005).

[49] Sergio Saponara et Bruno Neri. “Radar Sensor Signal Acquisition and Multidimensional FFT Processing for Surveillance Applications in Transport Systems”. In : *IEEE Transactions on Instrumentation and Measurement* 66.4 (2017), p. 604-615. doi : 10.1109/TIM.2016.2640518.

[50] Matthias Kronauge, Christoph T. Schroeder et Hermann Rohling. “Radar target detection and Doppler ambiguity resolution”. In : *11-th INTERNATIONAL RADAR SYMPOSIUM* (2010), p. 1-4.

[51] M.T. Orchard et G.J. Sullivan. “Overlapped block motion compensation : an estimation-theoretic approach”. In : *IEEE Transactions on Image Processing* 3.5 (1994), p. 693-699. doi : 10.1109/83.334974.

[52] Bruce Lucas et Takeo Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI)”. In : t. 81. Avr. 1981.

[53] Krishna Rao Vijayanagar. *I, P, and B-frames – Differences and Use Cases Made Easy*. OTT Verse. url : https://ottverse.com/i-p-b-frames-idr-keyframes-differences-usecases/.

[54] Humberto Dominguez et al. “The H.264 Video Coding Standard”. In : *Potentials, IEEE* 33 (mars 2014), p. 32-38. doi : 10.1109/MPOT.2013.2284525.

[55] Simone Bianco et al. “Benchmark Analysis of Representative Deep Neural Network Architectures”. In : *IEEE Access* 6 (2018), p. 64270-64277. doi : 10.1109/ACCESS.2018.2877890.

[56] Joseph Redmon et al. “You Only Look Once : Unified, Real-Time Object Detection”. In : *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, p. 779-788. doi : 10.1109/CVPR.2016.91.

[57] Krishna Rao Vijayanagar. *Ultralytics : YOLOv5*. GitHub. url : https://github.com/ultralytics/yolov5 (visité le 05/12/2022).

[58] Hermann Rohling. “Radar CFAR Thresholding in Clutter and Multiple Target Situations”. In : *IEEE Transactions on Aerospace and Electronic Systems* AES-19.4 (1983), p. 608-621. doi : 10.1109/TAES.1983.309350.

[59] Gui Gao et al. “An Adaptive and Fast CFAR Algorithm Based on Automatic Censoring for Target Detection in High-Resolution SAR Images”. In : *IEEE Transactions on Geoscience and Remote Sensing* 47.6 (2009), p. 1685-1697. doi : 10.1109/TGRS.2008.2006504.

[60] Frederik Maes et al. “Multi-modality image registration by maximization of mutualinformation”. In : t. 16. Juill. 1996, p. 14-22. isbn : 0-8186-7368-0. doi : 10.1109/MMBIA.1996.534053.

[61] Donald J. Berndt et James Clifford. “Using Dynamic Time Warping to Find Patterns in Time Series”. In : *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*. AAAIWS’94. Seattle, WA : AAAI Press, 1994, 359–370.

[62] Sébastien Cardinael et Arthur Malcourant. “Conception d’un senseur intégré multimodal pour l’observation des routes”. Prom. : Craeye, Christophe ; Macq, Benoît. Mém. de mast. École polytechnique de Louvain, Université catholique de Louvain, 2018.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**