UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

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

Caméra & radar

Mémoire présenté par
Sébastien CARDINAEL , Arthur MALCOURANT

en vue de l'obtention du grade de Master
Ingénieur civil électromécanicien, ingénieur civil mécanicien

Promoteur(s)
Christophe CRAEYE, Benoît MACQ

Lecteur(s)
Christophe PARISOT, André VAN DEN BOGAERT

Année académique 2017-2018

Préface

Ce sujet de mémoire nous a passionné. Nous nous y sommes investis en ne ménageant ni nos efforts de compréhension ni notre temps. Cependant, ceci n'aurait pas donné autant de résultats si nous n'avions pas bénéficié de l'aide de nos promoteurs Benoît Macq et Christophe Craeye. Nous tenons à les remercier chaleureusement à la fois pour leurs conseils et leurs compétences qu'ils ont tenu à partager avec nous. De plus, ils nous ont permis de rencontrer deux industriels André Van den Bogaert et Christophe Parisot qui nous ont apporté une vision commerciale et extérieure sur le projet. Nous tenons à les remercier pour leurs conseils avisés. Nous remercions aussi Thomas Feuillen pour ses nombreuses explications quant au radar utilisé, David Spôte et François Hubin pour la sélection de la batterie ainsi que Nicolas Leblanc pour ses conseils concernant le choix de la caméra. Nous sommes également reconnaissants envers Christophe Deleval et Maxime Lahy pour nous avoir fourni leur TFE ainsi que leurs connaissances dans ce domaine.

Notre avenir professionnel dépend de beaucoup de facteurs qui nous sont encore inconnus mais sur le plan de notre formation nous avons appris à prendre un problème en main avec ses inévitables retours en arrière, appréhension des limites tant techniques qu'humaines mais aussi les joies d'avoir progressé. La persévérance, le retour indispensable du moral après des essais ratés, la compréhension et la synthèse d'articles scientifiques parfois difficiles d'approche, l'utilisation de notions vues mais qu'il a fallu redécouvrir sont incontestablement des acquis que ce mémoire nous a inculqué pour la vie. Merci à tout l'EPL pour l'ambiance du travail mais aussi l'amitié qui s'est développée entre nous. Merci aux professeurs chercheurs qui ont participé avec dévouement à notre formation. Merci aussi à nos parents qui dans le contexte familial nous ont souvent encouragé à ne pas laisser tomber les bras. Merci à notre grand-père commun, professeur ordinaire émérite, Georges Cardinael pour la relecture de ce travail et ses conseils de rédaction.

"Tout ce qui doit durer est lent à croître"

Il est temps maintenant de commencer à partager nos connaissances pour le bien des autres.

Arthur Malcourant et Sébastien Cardinael, Louvain-la-Neuve, le 10 juin 2018.

Table des matières

Abstract

Le suivi de cibles multiples joue un rôle important dans le monde de l'ingénierie. Dans ce mémoire, un senseur multimodal composé d'une caméra et d'un radar est utilisé pour suivre des piétons et des voitures respectivement. La méthode implémentée se découpe en trois grandes étapes : la détection, le suivi et la fusion de données. La solution finale utilise le détecteur de piétons FPDW ("The Fastest Pedestrian Detector in the West"), les véhicules sont quant à eux observés à l'aide du radar KMD2. Ces cibles potentielles sont suivies par le biais de filtres de Kalman permettant d'introduire une approche stochastique. La dissociation entre plusieurs piétons ou voitures est réalisée grâce à une variante de l'algorithme Hongrois de James Munkres. La fusion des informations radar-caméra après plusieurs traitements de données est réalisée dans le but d'étudier les caractéristiques de fréquentation d'une route. La précision du senseur multimodal a été vérifiée à travers un cas d'étude spécifique d'un carrefour fort fréquenté. Au vu des résultats obtenus par les méthodes mises en place, il est apparu que cette association caméra-radar est pertinente. Des applications sont multiples et certaines recommandations d'amélioration permettront de faire encore évoluer ce nouveau dispositif.

Mots-clés : Suivi de cibles multiples, Caméra, Radar, Filtre de Kalman, Détection de piétons, Algorithme Hongrois, Senseur multimodal, Analyse routière

Multiple object tracking plays an important role in the world of engineering. In this thesis a camera and a radar are used to track pedestrians and cars respectively. The implemented method is divided into three main steps : detection, tracking and data fusion. As a pedestrian detector for the final solution, the FPDW ("The Fastest Pedestrian Detector in the West") is used. Cars are detected thanks to the radar KMD2. These potential targets are followed by means of Kalman filters which introduce a stochastic approach. To allow the dissociation between several pedestrians or cars, a James Munkres's variant of the Hungarian assignment algorithm is operated. The data fusion is then realised after quite a few processing on them. The purpose is to analyse the attendance of a road. The accuracy of the multimodal sensor was checked through tests on a specific case study. This one is a crowded crossroad. From the results achieved by the implemented algorithm, it appears that this combination between a camera and a radar is relevant. Multiple applications and some recommendations for improvements will push this new device forward.

Keywords : Multiple object tracking, Camera, Radar, Kalman filter, Pedestrian detection, Hungarian assignment algorithm, Multimodal sensor, Traffic analysis

Table des figures

Liste des tableaux

Chapitre 1

Introduction

A ce jour, de nombreuses applications d'analyse de voitures et de piétons sont développées dans le secteur de l'ingénierie. Parmi celles-ci figure le comptage de voitures sur autoroute, la détection de piétons sur voie ferrée, la prérimétrie pour la surveillance,... Dans bien des cas, les systèmes implémentés se concentrent sur un certain type de cible pour des lieux définis préalablement.

Dans ce mémoire, une combinaison de ces applications a été imaginée dans le but d'effectuer une analyse routière axée sur les piétons et les voitures. L'idée est d'utiliser les capteurs les plus adéquats à un certain type de cible pour former un dispositif multimodal polyvalent. Pour ce faire, l'analyse des piétons se fera à l'aide d'une caméra tandis que celle des voitures sera réalisée avec un radar et ceci simultanément.

L'objectif de ce travail est donc d'imaginer un dispositif innovant permettant d'analyser les caractéristiques de fréquentation d'une route. On peut définir celles-ci comme étant le nombre de piétons et de voitures, leurs trajectoires respectives ainsi que la vitesse moyenne des véhicules sur un intervalle de temps. Ces caractéristiques obtenues et fusionnées pour les deux senseurs permettent de définir une interaction piétons-voitures. Celle-ci nous informe sur le nombre de voitures se trouvant sur la route lors de la traversée d'un piéton, quelle distance minimale les sépare et à quelle vitesse roule la voiture à cet instant. Cette analyse peut alors donner une notion du danger de la route étudiée.

Pour atteindre cet objectif, une méthode de suivi de cibles multiples doit être implémentée. Elle permet de dissocier les différentes cibles de chaque type entre elles et de stocker les informations nécessaires à l'analyse (positions, vitesses) les concernant. La complexité réside dans un premier temps dans la détection des cibles. Il faut localiser uniquement les piétons sur l'image et associer les nombreuses détections radar aux voitures. Cette étape doit être réalisée astucieusement. En effet, sans de bonnes détections, aucun suivi ne peut s'effectuer correctement. Ensuite, la seconde complexité se trouve dans la différenciation des cibles de même type entre elles. Il faut leur associer les détections radar-caméra correspondantes. A nouveau, cette étape est indispensable pour obtenir les informations de déplacements temporels conformes à chaque cible. Finalement, la troisième complexité demeure dans les détections bruitées, parasites ou manquantes qui peuvent détériorer la qualité du suivi.

Une fois les objectifs fixés et leurs complexités dévoilées, le développement des solutions s'est fait en plusieurs étapes. Afin de vérifier cette implémentation de suivi, un cas d'étude a été choisi. Celui-ci est un carrefour fréquenté par de nombreux piétons et voitures. Il représente donc le lieu idéal pour tester ce senseur multimodal. Les solutions apportées et leurs résultats encourageants sont présentés tout au long de ce mémoire.

Afin d'exposer nos démarches, nous avons tout d'abord présenté les dispositifs utilisés pour ce travail, une justification concernant leur choix est développée. Le cas d'étude et ses challenges sont également décrits au chapitre 2. Ensuite, les recherches théoriques quant aux méthodes existantes de détection et de suivi de cibles sont rassemblées dans le chapitre 3. Suite à ces recherches, une première implémentation est effectuée. Celle-ci rend compte d'un bon fonctionnement algorithmique vis-à-vis de cibles multiples dévoilant toutefois des défauts à corriger (Chapitre 4). Sur base de ce premier essai, des traitements de données sont ajoutés et des optimisations de paramètres sont effectués. Ceux-ci permettent d'obtenir une solution concrète et de remplir les objectifs fixés (Chapitre 5). Dès lors, nous avons confronté cette recherche à la réalité grâce au cas d'étude. Une analyse complète de celui-ci est donc présentée. Celle-ci donne une idée du danger de ce lieu à l'aide du dispositif (Chapitre 6). Finalement, au vu des résultats obtenus, des améliorations permettant de faire encore évoluer le fonctionnement du senseur ainsi que des idées d'applications futures sont proposées (Chapitre 7).

Chapitre 2

Description du dispositif

Avant de présenter les deux senseurs, il est important de rappeler le but principal de ce travail. Il consiste en la combinaison d'une caméra et d'un radar pour l'analyse des routes. Pour réaliser cet objectif, les senseurs doivent pouvoir être placés sur le lieu d'étude. Les deux capteurs ont donc été assemblés à l'intérieur d'une boîte permettant de transporter les différents câbles ainsi que la batterie d'alimentation du radar (voir figure 2.1).

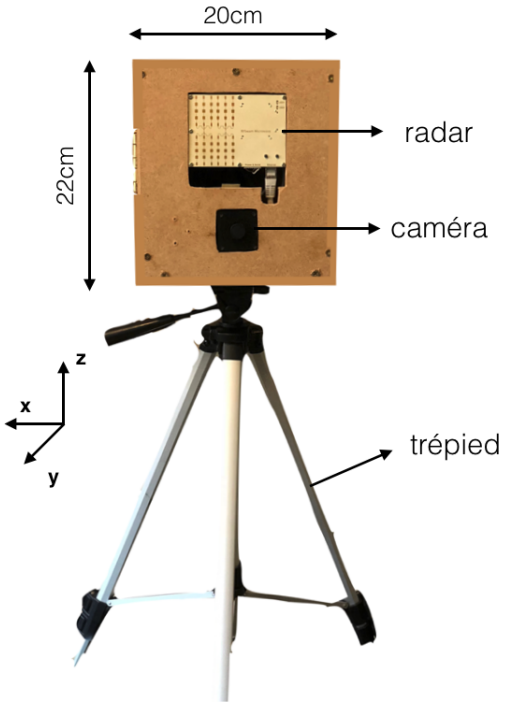


FIGURE 2.1 – Montage du dispositif utilisé

Dans cette partie, les deux dispositifs utilisés pour le projet ainsi que leur couplage seront présentés. Nous justifierons, à chaque fois, le choix de ces deux capteurs. Finalement, le cas d'étude (un carrefour de Louvain-la-Neuve) sur lequel est basé ce travail sera décrit.

2.1 Caméra

La caméra utilisée pour ce projet est une caméra USB de chez ELP. Certains critères devaient être pris en compte pour permettre un choix optimal. Tout d'abord, par souci de simplicité au niveau de l'utilisation informatique et de la programmation, son port devait être de type USB. Ensuite, la qualité de la caméra devait être suffisante que pour permettre un bon fonctionnement de détection à de grandes distances. Ci-dessous, on reprend ses caractéristiques utiles pour notre travail (table 2.1).

Senseur	1/3" CMOS AR0330
Autofocus	Oui
Résolution	Max. 1920x1080
Protocole USB	USB 2.0 HS/FS
Prix	45€
Lentille	12mm
Paramètres ajustables	Luminosité/Contraste/Couleur saturation / Definition/Gamma/WB
Angle d'ouverture	30°
Cadence de prise d'images	30 *fps*
Source de courant	DC 5V (l'alimentation se fait par le câble USB connecté à l'ordinateur)

Table 2.1 – Caractéristiques de la caméra

Cette caméra répondait à nos exigences en ce qui concerne la qualité et la facilité d'emploi. De plus, son prix est raisonnable (45 euros). Lors de son utilisation, on a pu constater l'avantage de son zoom. Cette caractéristique constitue un atout pour l'analyse étant donné que la caméra renvoie une image de bonne qualité malgré qu'elle se situe au même endroit que le radar (section 2.3) se trouvant à plus de 40 mètres du carrefour étudié (section 2.4).

2.1.1 Calibration de la caméra

Dans le but d'appliquer correctement les différents modèles liés à la caméra dans la suite de ce travail, nous avons effectué une calibration géométrique de celle-ci. Le but étant donc d'estimer les paramètres de la lentille et du capteur d'image pour permettre par exemple, de mesurer la taille d'un objet dans le monde réel ou encore de corriger la distorsion. Pour estimer ces paramètres, il faut obtenir la correspondance entre des points dans l'espace en trois dimensions et les points image en deux dimensions. Ces relations sont obtenues grâce à des modèles d'étalonnage (dans notre cas, nous utilisons un damier).

La calibration a été réalisée à l'aide du programme Matlab. L'algorithme existant permet d'obtenir ces paramètres et de calculer une erreur de reprojection (distances en pixels entre les points détectés et reprojectés [1]) [30]. Pour ce faire, de multiples photos d'un damier sont prises avec la caméra placée à différentes positions (figure (2.2)-(2.3)). Connaissant les dimensions de chaque carré du damier ainsi que son origine, l'algorithme utilise un modèle *pinhole* (section 4.1.1) comme modèle de transformation du trois dimensions vers du deux dimensions. Cela permet d'obtenir, après la résolution de l'équation de correspondance, les paramètres intrinsèques et extrinsèques de la caméra.

1. Les points reprojectés sont les points connus du damier dans l'espace 3D qui sont projetés sur l'image en pixels

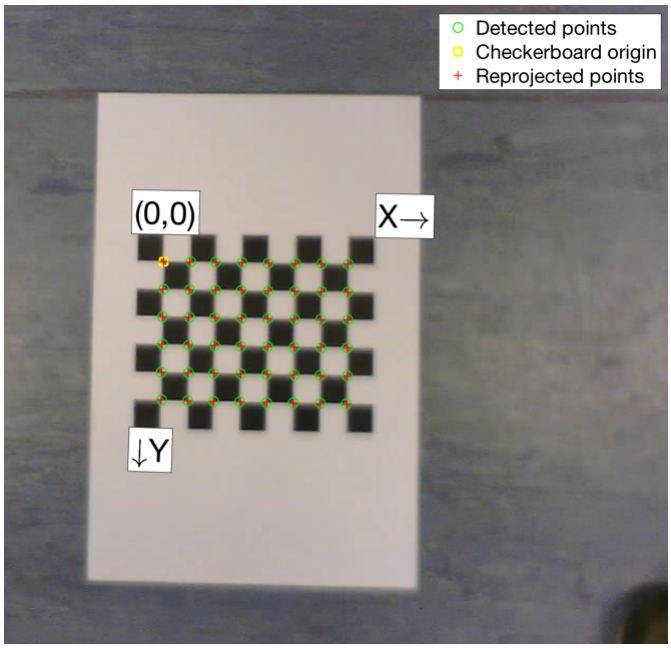


FIGURE 2.2 – Damier avec les points reprojectés et détectés pour la calibration

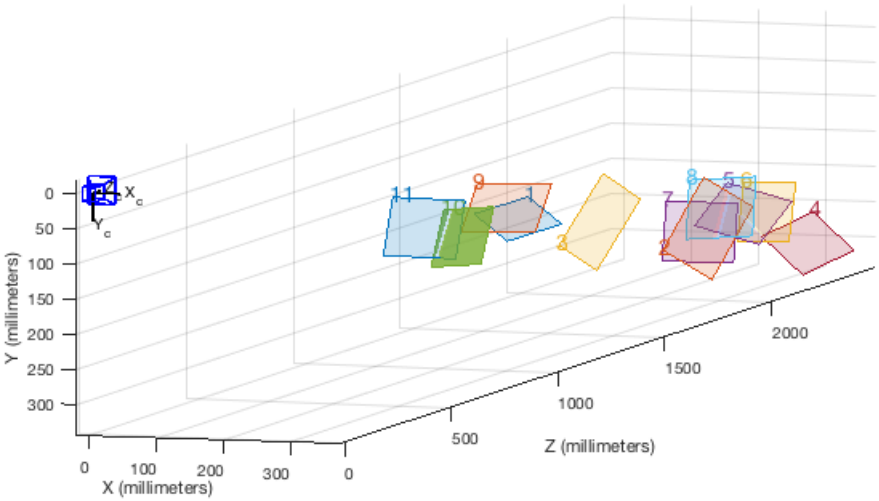


FIGURE 2.3 – Onze prises de vue différentes du damier avec la caméra

Les paramètres renvoyés par le programme et utiles pour les modèles implémentés dans ce travail sont la distance focale en pixels ainsi que l'erreur moyenne de reprojection (figure 2.4).

Distance focale = 5695.8 pixels

Erreur moyenne de reprojection = 0.78 pixels

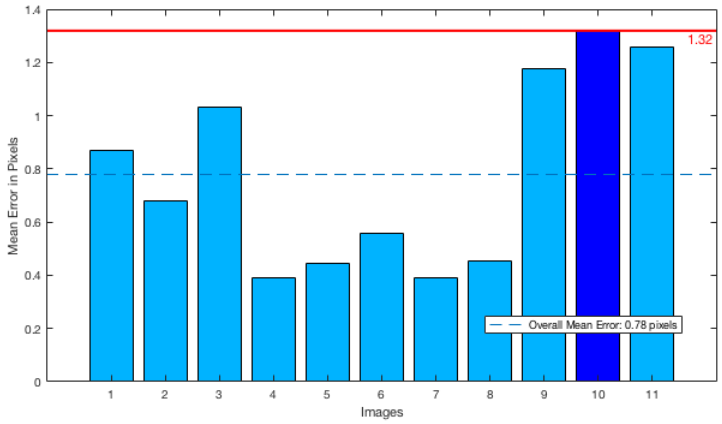


FIGURE 2.4 – Erreur moyenne de reprojection pour les onze prises de vue différentes

Finalement, connaissant les propriétés de cette caméra, on peut conclure qu'elle est un outil adéquat pour ce travail.

2.2 Radar

Pour remplir les objectifs du travail, le radar KMD2 de chez RFbeam a été sélectionné. Ce choix a été réalisé pour des raisons pratiques, mais également pour la performance du dispositif. Tout d'abord, ce radar avait déjà été utilisé dans le TFE de Deleval Christophe et Lahy Maxime [15] et nous a été prêté par l'UCL. De plus, ce radar réalise le traitement de signaux en interne et le code, permettant de le faire fonctionner, est écrit sous Matlab (environnement dans lequel nous travaillons). Finalement, le radar permet la détection de la position et la vitesse de cibles multiples sur un même *frame* et présente une bonne résolution. Ses caractéristiques sont présentées dans le tableau (2.2) ci-dessous.

Gamme de fréquence	24000-24250GHz
Puissance de sortie	20dBm
Nombre d'antennes émetrices	1
Nombre d'antennes réceptrices	3
Sortie	Ethernet
Distance de détection maximale	100m (piétons) , 300m (voitures). Résolution : 1m
Angle de vue	+- 15° (azimuth) et +- 10° (élévation). Résolution : 0.1°
Vitesse maximale de la cible	+- 128km/h. Résolution : 1km/h
Prix	1100€
Alimentation	12V / 0.6 Amps

TABLE 2.2 – Caractéristiques du radar

Ce radar permet de renvoyer la liste contenant la vitesse, la distance, l'angle azimutal et l'élévation d'une cible dans le champ de vision et ce pour un maximum de 256 cibles par *frame*. Il peut analyser jusqu'à 20 *frames* par seconde dépendant de la complexité du traitement. Lors de nos tests, celui-ci fonctionnait toute fois à une cadence de 15 *frames* par seconde (*fps*). Un dernier point important à mentionner est son alimentation. Il fonctionne sous une tension de 12 volts avec un courant de 0.6 ampère. Une batterie de petite taille (13x7x7cm) suffit à l'alimenter. Elle est facilement transportable avec le dispositif. Le choix de ce radar est donc pertinent.

2.3 Couplage des dispositifs

Une fois ces deux capteurs sélectionnés, ils ont dû être groupés pour permettre la conception de ce senseur intégré multimodal. L'assemblage s'est donc fait sous différentes contraintes. Tout d'abord, ces deux capteurs doivent être alignés horizontalement et verticalement pour que les données renvoyées par chaque capteur puissent être utilisées dans le même plan x-y. Ensuite, le dispositif doit être placé à hauteur d'homme (hauteur : 1.5 mètres) et idéalement avec une visée parallèle au sol [2]. Les capteurs ont donc été fixés à l'intérieur d'une boîte se trouvant sur un trépied (le but est de pouvoir s'adapter au terrain et toujours pouvoir le disposer à bonne hauteur et parallèle au sol)(figure 2.1). Un plan a été réalisé (voir annexe A1) pour permettre la meilleure précision concernant l'alignement. Cette boîte permet également le transport facile de la batterie ainsi que l'ensemble des câbles des capteurs.

2.4 Cas d'étude

L'objectif de ce travail étant l'association de deux capteurs pour permettre l'analyse de la fréquentation et du danger des routes, il est intéressant de choisir un cas d'étude. Celui-ci permet de réaliser différents tests et de mettre en relief le travail de recherche réalisé. Le cas d'étude est un carrefour très fréquenté en voitures et en piétons situé Avenue Georges Lemaître à Louvain-la-Neuve. En effet, comme on peut le voir sur la figure (2.5), des emplacements de parking sont situés de part et d'autre de la route principale, celle-ci est donc régulièrement traversée par les piétons. De plus, des bâtiments fort fréquentés se trouvent de chaque côté (crèche, bâtiments de l'université,...). Aucun passage pour piétons n'existe à cet endroit, les voitures n'ont donc pas tendance à ralentir et roulent relativement vite (50 km/h). Ce cas d'étude est également complexe à analyser, la route tourne au niveau du carrefour et est en pente. Le dispositif est placé après le carrefour à 40 mètres en amont du tournant dans le but de couvrir le champ le plus large possible (figure 2.5).

Ce lieu d'étude est donc un endroit très bien adapté aux objectifs de ce travail. C'est un carrefour considéré dangereux au vu de la fréquentation et de la vitesse des voitures couplées à la traversée régulière de piétons. Grâce au dispositif installé, une analyse de la fréquentation pourra être effectuée (section 6) dans le but de mettre en avant le danger de ce type de carrefour.

2. Ces contraintes sont dues aux méthodes de détection. Le détecteur de piétons choisi pour ce travail est plus performant à hauteur d'homme. Le radar, lui, a un angle d'élévation très faible (faisceau pincé verticalement) et donc il ne couvrirait pas une large zone de la route s'il se trouvait en hauteur.

FIGURE 2.5 – Cas d'étude vu du haut

Chapitre 3

Détection et suivi de cibles multiples

Dans ce chapitre, les différentes recherches concernant les outils algorithmiques existants en rapport avec l'objectif du travail sont présentés. Pour rappel, l'objectif principal de ce travail est de réaliser le suivi simultané de cibles multiples aussi bien pour les piétons que pour les voitures. C'est en réalisant un bon *tracking* qu'une analyse de l'interaction piétons-voitures sera possible.

Nous avons effectué des recherches dans le but de rassembler les méthodes existantes permettant d'aboutir à notre objectif. Suite à ces recherches, des choix ont été faits parmi les algorithmes qui seront utilisés et justifiés dans les chapitres (4) et (5). Dans ce chapitre, les techniques existantes permettant de réaliser un suivi de cibles multiples à partir de détections seront présentées. Seules les plus adaptées pour ce travail ont été sélectionnées. Ensuite, nous développerons les méthodes permettant de détecter les cibles avec les capteurs utilisés.

3.1 Méthodes de suivi de cibles multiples

Le suivi de cibles multiples joue un rôle important dans le monde de l'ingénierie. En effet, dans de plus en plus de domaines tels que la surveillance, le traitement de signal, la robotique,... des techniques de suivi stochastiques sont développées afin d'améliorer les performances du système. Les capteurs utilisés (caméra, radar, GPS, lidar,...) dans les différentes applications sont toujours sujets à des bruits extérieurs dus à l'environnement ainsi que des bruits internes causés par le capteur lui-même faussant le suivi des cibles. Il est alors intéressant d'inclure aux détections un modèle probabiliste pour corriger les erreurs possibles des senseurs et définir une zone dans laquelle se trouve assurément la cible. La plupart de ces modèles sont basés sur le filtrage récursif de Bayésien [36] qui comporte deux étapes principales :

— Action : estimation de la position $\hat{x}_k$ avec un modèle d'état linéaire ou non linéaire à partir de l'ancien état x_{k-1}
— Perception : observation de la position z_k à l'aide d'un capteur et correction de celle-ci avec l'estimation pour obtenir la nouvelle position x_k

Seules les données de l'état antérieur et la nouvelle mesure sont nécessaires. La distribution de probabilité de l'état en k est alors mise à jour lors de ces deux étapes.

De nombreuses méthodes différentes utilisent ce principe de base. Dans la section suivante, nous allons développer l'analyse des deux types les plus répandus dans le domaine du suivi de cibles.

3.1.1 Filtre de Kalman

Le filtre de Kalman est le modèle classique des filtres Bayésiens. C'est un outil qui sert à estimer récursivement l'état d'un système à partir de l'état précédent et des mesures bruitées ou manquantes. Cette estimation se base sur un modèle dynamique linéaire défini par l'utilisateur. L'ensemble du modèle (le capteur, l'environnement, les déplacements de la cible,...) est caractérisé par une matrice de covariance qui décrit le degré de corrélation entre les différentes variables d'état (position, vitesse,...). Ce modèle suit une distribution de probabilité continue et gaussienne. Étant donné que ce type de distribution n'est décrit que par sa moyenne et sa covariance, seules ces deux valeurs doivent être mises à jour à l'aide d'équations simples pour chaque nouvelle détection rendant le temps de calcul très faible.

Son fonctionnement est similaire à celui des filtres Bayésiens : une estimation de position est réalisée ($\hat{x}_k$) à partir de l'ancien état, la nouvelle position (x_k) est alors déterminée en réalisant le meilleur compromis entre la détection du capteur (z_k) et la prédiction ($\hat{x}_k$).

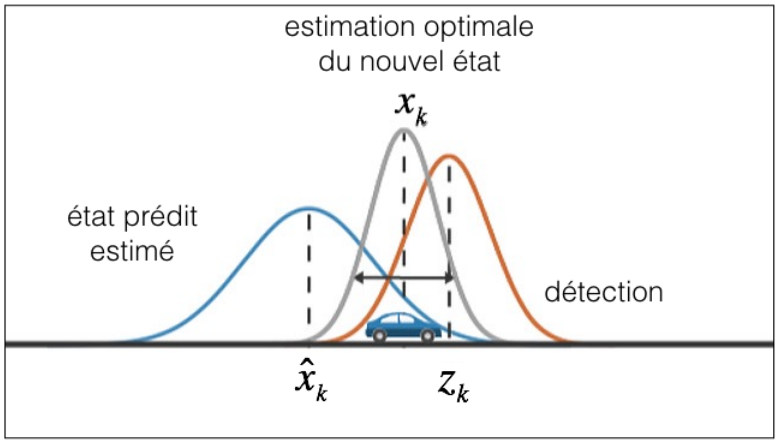


FIGURE 3.1 – Représentation des étapes du filtre de Kalman [27] et des courbes de distribution de probabilité

Les filtres de Kalman sont idéaux pour les systèmes qui changent continuellement [1]. Ils ont l'avantage d'être légers en mémoire et très rapides en terme de calculs. L'utilisation de tels filtres nécessite toutefois une bonne connaissance de la situation étudiée. En effet, le modèle dynamique linéaire utilisé doit correspondre au mieux aux déplacements des cibles pour avoir un état de prédiction cohérent. De plus, la modélisation de l'incertitude du capteur et du bruit environnemental doit avoir une distribution gaussienne ou du moins s'en rapprocher au mieux. Cette modélisation nécessite également la connaissance ou la détermination des variances et des moyennes de la matrice d'incertitude.

3.1.2 Filtre à particules

Le filtre à particules permet à nouveau d'estimer récursivement l'état d'un système à partir de l'état précédent et des mesures (pouvant être bruitées ou manquantes). Dans ce cas, l'état est décrit par un ensemble de particules pondérées qui représentent chacune une position possible de la cible [2]. Afin de prédire la position future, un modèle dynamique, pas nécessairement linéaire, est appliqué à chaque particule.

Au départ, un ensemble de particules est créé autour de la première détection. Celles-ci vont alors bouger selon le modèle dynamique posé (prédiction). Un poids (une importance) est

1. Ceci est dû au fait qu'un modèle dynamique est appliqué à chaque itération
2. Plus le poids est élevé, plus la cible a de probabilité de se trouver à cet endroit.

attribué à chacune d'entre elles et est défini en accord avec la fonction de densité de probabilité construite autour de la mesure du capteur (innovation). Cette fonction prend en compte le modèle d'incertitude du senseur préalablement calculé. Les particules sont alors réarrangées. Celles ayant un poids important sont remplacées par davantage de particules tandis que celles ayant un poids plus faible sont remplacées par peu de particules (échantillonage). Ces trois étapes sont alors répétées à chaque nouvelle détection (figure 3.2).

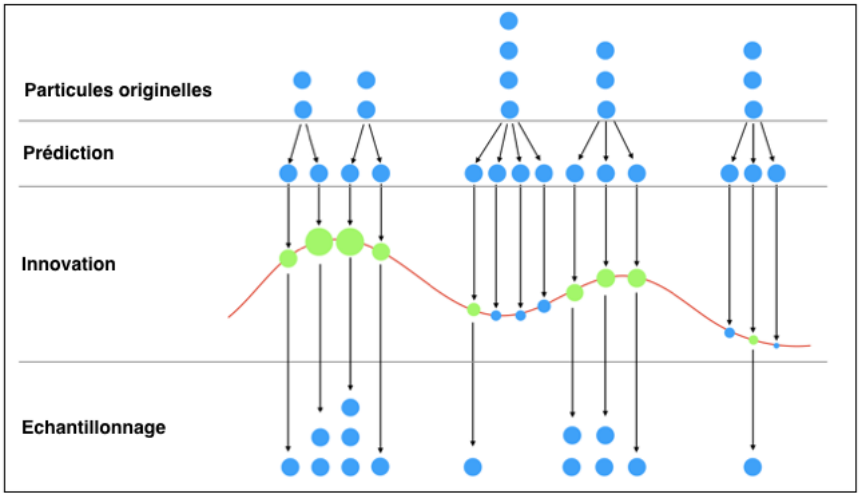


FIGURE 3.2 – Représentation 1D des trois étapes du filtre à particules [15]

Ce type de filtrage permet une grande adaptabilité du mouvement de la cible étant donné que la position est exprimée à l'aide d'un ensemble de positions possibles (particules). Le modèle d'incertitude peut quant à lui être déterminé avec une distribution quelconque bien que la plus part du temps, celui-ci est exprimé par une somme de distributions normales. Bien évidemment, les performances sont liées au nombre de particules qui elles mêmes sont liées au temps d'exécution. Le coût de calcul pour cette méthode reste très élevé même pour un faible nombre de particules. En effet, il faut, pour chaque étape, calculer : le déplacement, l'incertitude et le ré-échantillonage de chacune d'elles.

Finalement, au vu des points positifs et négatifs de ces deux méthodes, nous avons décidé d'utiliser le filtre de Kalman dans notre travail. La justification de ce choix et l'implémentation mathématique se trouvent à la section (4.2.2).

3.2 Méthodes de détections de piétons et de voitures

Les méthodes de suivi présentées ci-dessus ont naturellement besoin de détections pour fonctionner. Cette étape est primordiale au bon déroulement du suivi. Comme déjà mentionné précédemment il a été décidé de séparer les capteurs pour la détection de cibles différentes. En effet, la caméra sert à la détection et au suivi de piétons tandis que le radar sert à la détection et au suivi de voitures. Au vu de ce choix, les recherches ont été effectuées séparément. Dans la section suivante, les algorithmes de détections de piétons existants seront présentés. Ensuite, le fonctionnement particulier du radar utilisé sera décrit.

3.2.1 Caméra - détection de piétons

La détection correcte du piéton est une étape importante à l'algorithme de suivi. Il est donc intéressant de connaître les différentes méthodes existantes dans le but de faire un choix parmi celles-ci. Ces techniques de détection sont nombreuses et sont constituées de modèles statistiques établis par apprentissage supervisé [3] à partir de diverses caractéristiques visuelles (*features*) d'une zone de l'image. Les *features* d'une image sont des transformations mathématiques calculées sur les pixels de celle-ci. Ces caractéristiques peuvent être calculées localement ou globalement. Elles vont permettre de mieux rendre compte des propriétés visuelles de l'image (par exemple : couleur, forme, ...). Pour la détection de cibles, elles vont servir de "critères" pour dissocier une zone avec un piéton d'une zone sans. Ces caractéristiques peuvent être invariantes ou non du changement de taille de la zone prise sur l'image. Celles-ci seront plus détaillées dans la suite de ce développement. Les modèles statistiques permettent donc avec ces *features* de décider si une zone de l'image contient un piéton ou non. Ces zones sont généralement des boites rectangulaires qui encadrent le piéton, des *bounding boxes*.

L'algorithme de détection avec une caméra se présente généralement en trois étapes principales :

A. Génération de candidats avec localisation sur l'image

B. Opération de classification

C. Etape de suppression

Il existe plusieurs techniques pour réaliser chaque étape. Dans la suite, celles-ci sont présentées[7].

A. Génération de candidats

Lors de cette étape, l'image est scannée pour permettre de trouver les candidats potentiels. Ceux-ci passent en même temps à l'étape de classification pour vérifier s'ils contiennent un piéton. Pour ce faire, soit toutes les zones de l'image sont explorées et donc toutes ces zones passent à la classification, soit une pré-sélection des candidats est établie puis ceux-ci, seulement, passent à la classification.

A.1. Exploration complète de l'image La technique la plus connue et la plus simple permettant cette exploration complète est appelée la méthode de la fenêtre glissante. Pour ce faire, une fenêtre (*bounding box*) va passer sur toute l'image pour détecter si il y a un candidat. On peut également appeler cette boite une fenêtre de classification car elle "contient" le détecteur permettant de décider si un piéton se trouve à l'intérieur. Trois manières de réaliser ce scan existent :

3. Création de règles grâce à des bases de données d'exemples. En l'occurrence, dans ce cas, ce sont des photos d'humains.

A.1.1. *Dense image pyramid* Le but est de créer une pyramide d'images à de nombreuses échelles différentes . L'image va dès lors être sous-échantillonnée pour détecter des cibles plus larges et sur-échantillonnée pour détecter de plus petites cibles. Les caractéristiques visuelles de l'image sont recalculées pour chaque échelle. Ensuite, la fenêtre de classification ayant une taille fixe, va scanner chaque image (voir (a) figure 3.3). Le calcul des *features* pour chaque échelle d'image augmente la complexité du détecteur mais cela le rend plus efficace car tous les types de *features* peuvent être utilisés pour celui-ci.

A.1.2. *Classifier pyramid* Dans ce cas, la fenêtre a une taille variable et l'image ne change pas d'échelle. L'image est donc scannée par des fenêtres de classification de taille différente (voir (b) figure 3.3). Cependant, uniquement les *features* invariantes de la transformation d'échelle peuvent être utilisées pour le détecteur [4], ce qui restreint la qualité de celui-ci car la plupart des caractéristiques visuels ne sont pas invariantes de la transformation. Par contre, cela permet une détection plus rapide car l'échelle de l'image n'est pas changée à de multiples reprises et les *features* ne sont pas recalculées.

A.1.3. Méthode hybride Généralement, le choix entre les deux méthodes présentées ci-dessus est souvent un compromis entre rapidité de détection et performance. C'est pourquoi une méthode hybride a été inventée [16]. Celle-ci consiste à passer des fenêtres de classification de taille variable pour des échelles de l'image différentes (moins nombreuses que dans le cas du *dense image pyramid*). Dans ce cas, les *features* qui ne sont pas invariantes de la transformation sont approximées [5] pour chaque échelle de la fenêtre de classification. Cette méthode permet d'approcher la rapidité du *classifier pyramid* avec la précision de la *dense image pyramid*.

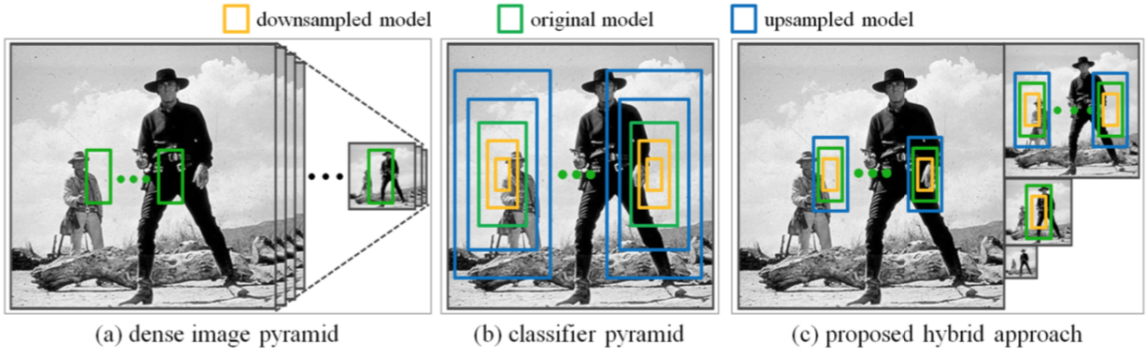


FIGURE 3.3 – Différentes méthodes de la fenêtre glissante

A.2. Recherche sélective Au vu du coût des calculs requis pour explorer une image complète, d'autres méthodes existent permettant d'augmenter la rapidité du détecteur. Celles-ci consistent en la pré-sélection de candidats sans passer par toute l'image [4]. Généralement l'image est segmentée [6] suivant différents critères pour permettre l'obtention des zones particulières à analyser. Il est intéressant de mentionner dans le cas d'un *tracking* (détection sur des *frames* d'une vidéo), la pré-sélection effectuée à l'aide du mouvement. Pour ce faire une méthode de *background substraction* est utilisée. Celle-ci permet de visualiser des zones de l'image qui ont bougé par rapport à un modèle initiale [7]. Cette méthode va réduire la complexité en présentant

4. Etant donné que la taille de la fenêtre varie, il faut que ce qui se trouve à l'intérieur soit indépendant de l'échelle pour permettre une comparaison correcte lors de la classification

5. Ces méthodes d'approximation de caractéristiques qui ne sont pas invariantes de la transformation d'échelle sortent du cadre de ce TFE et ne sont donc pas expliquées ici [16].

6. Partition de l'ensemble des pixels de l'image en différents groupes

7. Cette méthode est plus amplement expliquée à la section (7.1) car celle-ci a été implémentée en guise d'amélioration de notre algorithme.

un nombre restreint de candidats au classificateur.

Pour la génération de candidats, le détecteur choisi réalise une exploration complète de l'image à l'aide de la méthode hybride. Ce choix est détaillé à la section (4.1.1).

B. Opération de classification

Pour réaliser cette étape de classification, la plupart des techniques de détection utilise des modèles statistiques qui sont entraînés avec des vecteurs de caractéristiques visuelles calculées sur une base de données d'images contenant des piétons. La fonction permettant de décider si il y a présence d'un piéton ou non est déterminée par ce modèle. Deux grands types de modèle existent, les modèles discriminants ainsi que les modèles génératifs. Nous ne présenterons pas les modèles génératifs qui sont peu répandus.

B.1. Modèles discriminants Ces modèles approximent une décision Bayésienne en se basant sur les paramètres d'une fonction qui établit une distinction entre une classe de piétons et de non-piétons (cette fonction étant entraînée à l'aide d'exemples d'images existantes). Cette fonction discriminante se base sur les *features*. Comme déjà mentionné, celles-ci sont des sources hétérogènes d'information concernant des zones de l'image. Ces zones peuvent correspondre à des points, des contours ou des régions d'intérêts. A chaque *feature* correspond un vecteur ou un scalaire. Les *features* les plus connues et les plus utilisées dans la détection de piétons sont :

— *Haar wavelet features* : des fenêtres délimitant des zones rectangulaires adjacentes sont définies. La somme de l'intensité des pixels est calculée pour chaque fenêtre et ensuite la différence entre ces sommes constituent la caractéristique d'ondelettes de Haar.

— Histogramme de gradients orientés (HOG) : des histogrammes locaux de l'orientation du gradient (variation locale du niveau de gris) sont calculés sur des zones régulièrement réparties sur l'image.

Des variantes ainsi que de nombreuses autres *features* existent. Chaque *feature* a ses caractéristiques concernant la performance du détecteur (un détecteur est performant si il renvoie des détections exactes avec pas ou peu de faux positifs [8]) et la rapidité de calculs. C'est pourquoi les meilleurs opérations de classification se font en combinant différents types de *features*.

Avec ces critères de distinction établis, il est important d'avoir une architecture de classification efficace c'est à dire de définir la technique de décision. A nouveau, différentes fonctions existent et les principales seront présentées ci-dessous :

B.1.1. Machines à vecteurs de support (SVM) Cette méthode est un outil puissant pour résoudre des problèmes de classification. Cette fonction de classification est entraînée par une large base de données. Grâce à celle-ci, la fonction définit un hyperplan (frontière) entre deux classes (piétons et non-piétons) pour une *feature* sélectionnée. Cette frontière de séparation a pour but de maximiser la distance (marge) entre elle et les échantillons (+ O) les plus proches (figure 3.4). Le but étant d'appliquer une séparation maximale entre les classes d'objets, on a donc un classificateur binaire et linéaire non probabiliste. Cette méthode peut également être appliquée si les données ne sont pas linéairement séparables auquel cas l'espace des données d'entrée est transformé en un espace de plus grande dimension pour lequel on peut réaliser une séparation linéaire (figure 3.5). Cela augmente, cependant, le coût de calcul.

8. Détecte un piéton alors qu'il n'y en a pas à l'endroit détecté

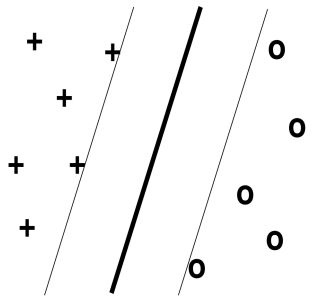


FIGURE 3.4 – Exemple d'un hyperplan (ligne en gras) sur des données 2D linéairement séparables. Celui-ci maximise la distance entre les données "croix" et "rond" les plus proches. [10]

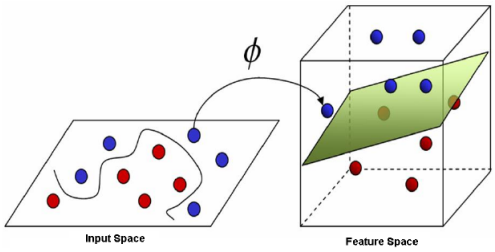


FIGURE 3.5 – Exemple de données qui ne sont pas linéairement séparables. La dimension de l'espace des données d'entrée est augmentée pour permettre une séparation linéaire. [33]

B.1.2. *Adaptive boosting* L'*adaptive boosting* permet de construire un classificateur fort qui est en réalité composé de plusieurs classificateurs faibles [9]. Ce classificateur fort est une combinaison linéaire pondérée des classificateurs faibles sélectionnés. Lors de la phase d'entraînement (c'est à dire lorsque l'algorithme est utilisé sur la base de données d'images), la pondération de l'importance des classificateurs faibles est déterminée de façon itérative. En effet, un classificateur faible est d'abord créé et testé sur la base de données, ensuite un prochain classificateur faible est entraîné et ajusté en fonction des erreurs du classificateur précédent et ainsi de suite. La complexité ainsi que la pondération des nouveaux classificateurs faibles augmentent au fur et à mesure de la phase d'entraînement.

B.1.3. Réseau neuronal convolutif Contrairement aux deux méthodes présentées ci-dessus dans lesquels les *features* sont sélectionnées par l'auteur de l'algorithme, le but de cette méthode est de sélectionner automatiquement un modèle complexe permettant d'extraire des *features* en fonction du contenu au niveau des pixels lors de la phase d'entraînement. Cette technique doit être entraînée sur de grandes bases de données de piétons pour avoir une bonne précision [17].

Le détecteur choisi pour ce travail utilise un modèle discriminant qui est l'*adaptive boosting*. Ce choix est détaillé à la section (4.1.1).

C. Étape de suppression

A la fin des deux étapes permettant la détection et la localisation des piétons, plusieurs fenêtres de classification ont détecté la même cible. Cette étape permet de garder la *bounding box* la plus adéquate entourant celui-ci (voir figure 3.6). Nous utilisons l'algorithme de *non-maximum suppression* (NMS) [13]. La procédure la plus commune est celle de *Greedy*. Elle commence par choisir une boite supposée contenir un piéton possédant le meilleur score de certitude [10], les boites trop proches de celle-ci (sous un certain seuil de distance) sont supprimées. Ensuite la boite contenant le second meilleur score est choisie et ses voisines proches sont supprimées et ainsi de suite, jusqu'à n'avoir plus de boites à sélectionner.

9. Un classificateur faible a un seuil de décision pour une seule *feature*. C'est par exemple un SVM présenté ci-dessus

10. Les algorithmes de détection renvoient, généralement, un score en pourcent concernant la certitude de contenance d'un piéton.

FIGURE 3.6 – Étape de suppression appliquée sur un piéton (NMS)

3.2.2 Radar - détection de voitures

Le radar va être utile dans l'algorithme de suivi pour localiser les voitures et retourner leur vitesse. Afin d'obtenir ces données, le radar va émettre des ondes modulées qui seront réfléchies contre les cibles. Celles-ci vont revenir au dispositif en apportant les informations nécessaires. Un traitement de signal est alors réalisé. Dans un premier temps, le signal reçu va être démodulé et placé ainsi en bande fréquentielle de base. Une transformée de Fourier en deux dimensions (vitesse et distance) sera alors appliquée pour obtenir les données finales. Dans cette section, une explication physique du fonctionnement du radar est réalisée.

Le radar KMD2 utilisé est un dispositif à onde continue modulée en fréquences [23] [20]. Il effectue une variation permanente de la fréquence du signal d'émission à partir d'une fréquence initiale f_0. La modulation, dans le cas du KMD2, est linéaire (modulation en dent de scie). Cette méthode d'émission permet de calculer à la fois la distance à laquelle se trouve une cible ainsi que sa vitesse radiale (figure 3.7).

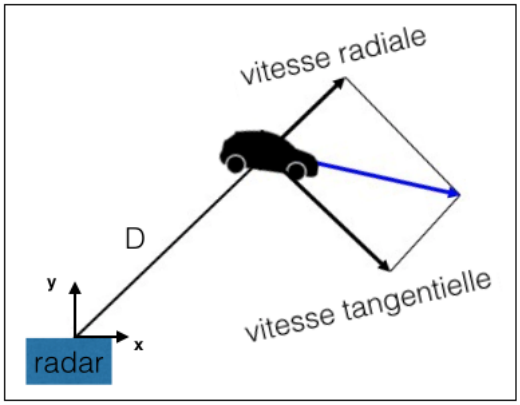


FIGURE 3.7 – Distance et vitesse radiale d'une cible mesurées par le radar

Calcul de la distance et de la vitesse

La distance (D) entre une cible fixe et un émetteur est facilement obtenue en calculant le temps entre l'émission et la réception du signal à l'aide de la formule suivante :

$$D = \frac{c_0 \cdot |\Delta t|}{2} \tag{3.1}$$

avec c_0 la vitesse de la lumière. L'évaluation de la distance nécessite une impulsion pour envoyer un signal et ensuite attendre l'écho retourné par la cible. Dans notre cas, le radar émet constamment et il serait donc impossible de dissocier l'écho de l'onde émise afin de calculer l'intervalle de temps les séparant. C'est pourquoi le radar émet en modulant, par un taux linéaire, sa fréquence. Il est alors possible de trouver une alternative au Δt (figure 3.8) :

$$\Delta t = \frac{\Delta f}{df/dt}$$

avec Δf la différence de fréquence entre l'onde émise et celle reçue et df/dt le taux linéaire de modulation.

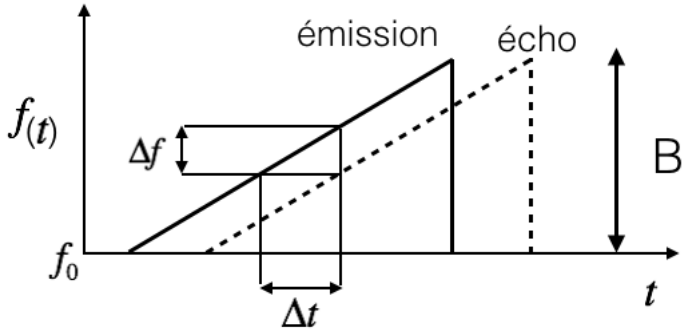


FIGURE 3.8 – Représentation du signal envoyé et reçu

Si la cible visée est en mouvement, le Δf va être modifié par l'effet Doppler et le résultat sera alors faussé. Cet effet résulte d'une variation de fréquence de l'onde après être réfléchie par la cible. La différence de fréquence totale devient alors :

$$\Delta f' = \Delta f + f_D \tag{3.2}$$

avec f_D le décalage Doppler qui dépend de la vitesse radiale.

Afin de calculer la vitesse radiale [11] et la distance d'une cible il faut alors étudier le signal envoyé et reçu pour finalement mettre en évidence ces deux données et réaliser une transformée de Fourier en deux dimensions. Les pics renvoyés par cette transformation nous fourniront leurs valeurs. Soit, le signal envoyé sous la forme suivante :

$$s = exp\left[j2\pi \int f(t)\mathrm{dt}\right]$$

avec $f(t) = f_0 + \frac{B}{T}t = f_0 + \Delta f_d$ la modulation en dent de scie, B la bande passante et T la période du signal.

11.
$$f_D = 2v_r \frac{f_0}{c_0 - v_r} \rightarrow v_r \approx \frac{c_0 f_D}{2f_0}$$
avec f_0 la fréquence d'émission, f_D la différence de fréquence entre l'onde émise et reçue, v_r la vitesse radial de la cible et en faisant l'approximation que $c_0 - v_r \approx c_0$.

Une fois l'onde envoyée, celle-ci revient après un certain délai qui dépend de la distance séparant le radar de la cible ainsi que du décalage Doppler.

$$r = s(t - \tau(t))$$
$$\text{avec} \quad \tau(t) = \frac{2D}{c} - \frac{2v_r t}{c}$$

Pour ressortir les informations voulues, il faut d'abord démoduler le signal et le placer en bande de base. Cette opération est réalisée à l'aide d'une démodulation cohérente [38] qui, pour un signal représenté sous forme complexe, revient à multiplier le conjugué du signal reçu par celui envoyé :

$$r_{BB} = r^* \cdot s \tag{3.3}$$
$$= exp[-j2\pi \int_0^t f(t)\ \mathrm{dt}] \cdot exp[j2\pi \int_0^{t-\tau(t)} f(t)\ \mathrm{dt}] \tag{3.4}$$
$$= exp[j2\pi \int_t^{t-\tau(t)} f(t)\ \mathrm{dt}] \tag{3.5}$$
$$= exp[j2\pi(-f_0\tau(t) + \frac{B}{2T}(\tau^2(t) - 2t\tau(t)))] \tag{3.6}$$

De l'expression obtenue après intégration, nous pouvons faire l'hypothèse que le terme en τ^2 peut être considéré comme une constante dans l'exponentiel de droite. En effet, celle-ci sera utilisée pour la détermination de la distance qui n'est quasi pas affectée par le Doppler. On obtient donc :

$$r_{BB} = C \cdot exp[j2\pi(-f_0(\frac{2D}{c} - \frac{2v_r t}{c}))] \cdot exp[-j2\pi\Delta f_d(t)(\frac{2D}{c} - \frac{2v_r t}{c})] \tag{3.7}$$

A nouveau, nous pouvons faire l'hypothèse que le terme $2v_r t/c$ de l'exponentielle de droite est négligeable. Nous remarquons également que le terme $2D/c$ de l'exponentielle de gauche est constant. Finalement, deux expressions, possédant en argument la distance et la vitesse recherchée, sont obtenues. Le radar travaillant en temps discret, il faut alors discrétiser notre signal pour le rendre traitable. La période d'échantillonnage sur une dent de scie est représentée par T_s, soit M échantillons pour la période T d'un signal (voir figure 3.9). On peut alors récrire l'équation de l'onde comme suit :

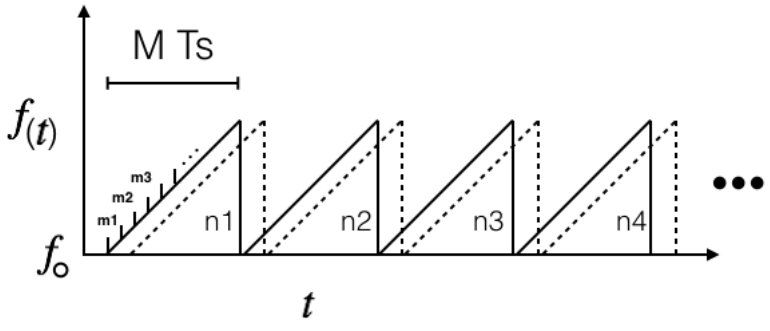


FIGURE 3.9 – Représentation du signal envoyé et reçu

$$r_{BB} = C \cdot exp[j2\pi f_0 \frac{2v_r}{c} nMT_s] \cdot exp[-j2\pi\Delta f_d(mT_s)\frac{2D}{c}] \tag{3.8}$$
$$= C \cdot exp[j2\pi f_0 \frac{2v_r}{c} nT] \cdot exp[-j2\pi \frac{B}{M} m \frac{2D}{c}] \tag{3.9}$$

Nous obtenons deux expressions dépendantes de la vitesse et de la distance, il suffit alors d'effectuer une transformée de Fourier discrète sur v_r et D pour les déterminer.

Dans le cas de détection de cibles multiples, le radar va, pour les dissocier, envoyer un total de 512 *chirps* (signaux) par *frame*. La transformée de Fourier en deux dimensions renverra alors plusieurs pics qui correspondront à des cibles différentes. L'équation de l'onde obtenue se réécrit :

$$r_{BB} = \sum_{k=1}^{512} C \cdot exp[j2\pi f_0 \frac{2v_{r_k}}{c} nT] \cdot exp[-j2\pi \frac{B}{M} m \frac{2D_k}{c}]$$

On peut observer le résultat de la transformée de Fourier sur la figure (3.10) [37]. Les trois pics observables sur ce graphe correspondent à trois cibles. On peut alors retrouver leur distance et leur vitesse sur les axes x et y respectivement.

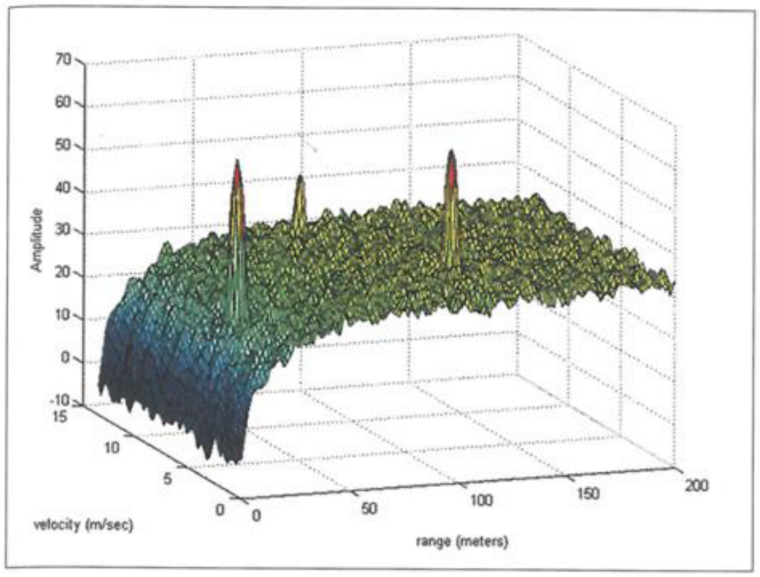


FIGURE 3.10 – Représentation de la transformée de Fourier 2D en vitesse (axe y) et distance (axe x), les pics représentent des cibles [37]

Calcul de l'angle azimutal

La distance et la vitesse obtenues à l'aide des formules précédentes sont exprimées en repère absolu. Il est intéressant dans notre application de trouver les composantes x et y de ces valeurs. Pour se faire, l'angle azimutal doit être calculé. Le radar KMD2 dispose de trois antennes réceptrices situées l'une à coté de l'autre, celles-ci vont être utilisées pour calculer le déphasage ϕ de l'onde reçue entre les antennes et ainsi calculer l'angle θ entre le dispositif et la cible. En faisant l'hypothèse de champ lointain (la distance cible-radar doit être bien plus petite que celle séparant les antennes), le signal réceptionné peut être approximé comme étant des ondes parallèles (figure 3.11).

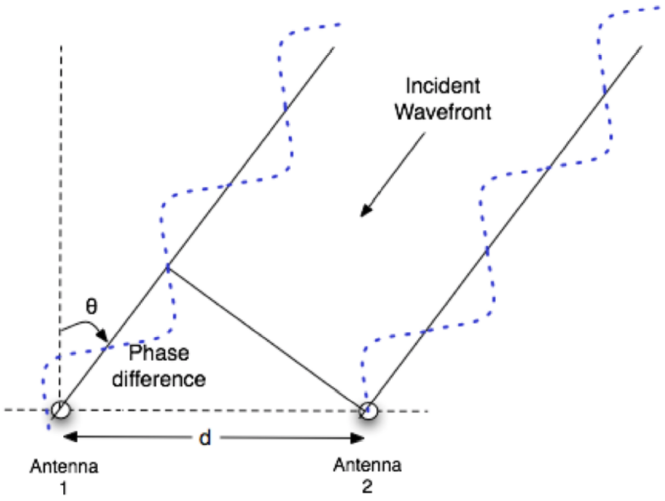


FIGURE 3.11 – Réception d'un signal sous l'hypothèse de champ lointain [37]

Finalement, l'angle est calculé comme suit :

$$d_\phi = \frac{\phi}{2\pi}\lambda$$

$$\theta = arcsin(\frac{d_\phi}{d})$$

avec $\lambda = c/f$, la longueur d'onde.

On peut donc conclure que le radar est capable de détecter plusieurs cibles par *frame* et de les localiser dans le plan x-y. Il permet également d'obtenir leur vitesse respective à tout instant. Ses compétences remplissent donc les exigences requises.

Chapitre 4

Choix préliminaire de l'implémentation du suivi de cibles multiples

Après avoir présenté les différentes étapes existantes permettant la détection et le suivi de cibles multiples, il faut maintenant choisir la méthode la plus appropriée à ce travail. Des décisions ont du être prises parmi les algorithmes présentés pour permettre de commencer au mieux cette étude. Celle-ci consiste donc à suivre simultanément des piétons et des voitures à l'aide d'une caméra et d'un radar. Nous avons décidé de séparer l'analyse des piétons de celle des voitures. Le *tracking* a été effectué indépendamment pour chaque capteur et ensuite la fusion de leurs données a été effectuée. Cette méthode de suivi décrite dans la suite de ce chapitre reste, de façon logique, fort semblable pour les piétons et les voitures.

Les méthodes choisies permettant de détecter les piétons et les voitures seront tout d'abord présentées et argumentées. Ensuite, la méthode de *tracking* utilisée sera expliquée pour les deux cas de suivi. Après, la fusion des données pour les piétons et les voitures sera définie. Finalement, des résultats de cette première méthode pour les deux capteurs utilisés indépendemment l'un de l'autre seront présentés suivi de premiers résultats de fusion sur la route étudiée.

4.1 Détections

La première étape de cette étude consiste à détecter la présence d'une cible qui devra être suivie. Dans cette section, les méthodes de détections implémentées sont développées avec une étude de leurs performances. Chaque capteur ayant une méthode de détection différente, leur analyse sera réalisée séparément. Tout d'abord, l'algorithme de détection de piétons sera présenté suivi de l'algorithme de détection de voitures.

4.1.1 Détections de piétons

Etant donné qu'il existe une large gamme de détecteurs de piétons utilisant les différentes méthodes présentées précédemment (section 3.2.1), un choix a du être réalisé. Basé sur l'analyse convaincante effectuée par Deleval Christophe et Lahy Maxime [15], le choix du détecteur de piétons a été dicté par la solution, écrite par Dollàr, Bellongie et Perona, appelée "The Fastest Pedestrian Detector in the West" [16].

Cet algorithme a donc été choisi au vu de ses bonnes performances théoriques mais également de son aspect pratique. Tout d'abord, compte tenu de l'étude effectuée sur le détecteur [3], celui-ci

obtient 60% de taux de détection par image sur la base de données Caltech-USA [1] comparé à 50% pour les autres méthodes concurrentes. Ces performances seront testées de façon plus approfondie dans la suite de cette section. Ensuite, en ce qui concerne son aspect pratique, le code est écrit en matlab, qui est l'environnement dans lequel nous travaillons. De plus, ce code avait déjà été implémenté par Deleval Christophe et Lahy Maxime dans leur TFE [15] et nous a été transmis directement. Celui-ci est également sous une licence BSD qui impose des restrictions minimales sur l'utilisation et la redistribution du code.

Dans cette section, l'algorithme et ses performances seront décrits. La méthode, couplée au détecteur, permettant d'obtenir la position des détections en mètres sera expliquée. Pour finir, nous analyserons le modèle d'erreur sur la position renvoyée par le détecteur sera présentée.

Description de l'algorithme

Cet algorithme est un algorithme dont la classification est concentrée sur les piétons. Celui-ci reprend certaines des différentes techniques déjà présentées à la section (3.2.1). Comme expliqué précédemment, la structure du détecteur se fait en trois étapes :

A. Génération de candidats avec localisation sur l'image (*bounding box*)

B. Opération de classification

C. Etape de suppression

A. Génération de candidats La méthode de génération de candidats utilise la technique de la fenêtre glissante hybride. Cette technique reprend donc à la fois l'approche *classifier pyramid* et l'approche *dense image pyramid*. Le principe de cette méthode consiste à scanner (comme expliqué précédemment à la section 3.2.1) l'image à différentes échelles avec des boites de tailles variables. Les données images (*features*) de la *bounding box* sont approximées mais sont très proches de celles obtenues si on avait utilisé uniquement la *dense image pyramid*. Pour rappel, cette méthode permet d'allier la précision de l'approche *dense image pyramid* avec la vitesse de l'approche *classifier pyramid*. Chaque boite est passée à la classification pour savoir si elle contient un piéton.

B. Opération de classification Le classificateur utilisé ici est un classificateur créé par *adaptive boosting*. Un certain nombre de classificateurs faibles [2] pondérés suivant leur importance sont combinés pour en obtenir un robuste. Ces classificateurs faibles sont entraînés à l'aide d'une base de données d'images de piétons. Ce classificateur robuste consiste donc en une cascade de classificateurs dont la complexité augmente au fur et à mesure. Des classificateurs non-complexes sont donc utilisés en premier lieu lors de la détection pour rejeter rapidement des fenêtres de classification ne contenant pas de piétons.

Lors de l'utilisation du détecteur, les *features* (telle que la somme locale, les histogrammes ou encore les ondelettes de haar) des fenêtres de classification sont approximées et calculées à l'aide de la méthode d'image intégrale [11]. Cette méthode est utilisée pour obtenir efficacement la somme d'une fonction $f(x, y)$ (somme de l'intensité lumineuse des pixels,...) sur une zone rectangulaire d'une image. Le calcul s'effectue au début sur toute l'image et ensuite, les valeurs pour n'importe quelle taille de rectangle peuvent être trouvées rapidement. Le principe est basé sur le fait que chaque point sur une image est représenté par la somme des éléments (*features*)

1. Cette base de données reprend des images prises en environnement urbain depuis une voiture. Elle contient plus de 2300 piétons différents en présence de voitures, vélos,... Les performances de la plupart des détecteurs sont évaluées sur cette base de donnée.

2. Pour rappel, chaque classificateur faible a comme critère de décision un type de *feature* qui est différente pour chaque classificateur.

de la matrice comprise entre le point supérieur gauche de l'image et le point considéré. Cette méthode est illustrée à la figure (4.1). Sur le premier schéma, on retrouve les simples valeurs d'entrée de l'image. Sur le second, la redéfinition de chaque point en sommant les éléments de la matrice correspondante est réalisée. Finalement, sur le troisième, l'intégrale image permet de calculer la somme sur un rectangle D de coin gauche supérieur (x_1, y_1) et de coin droit inférieur (x_2, y_2). Mathématiquement cela donne :

$$\sum_{x=x_1}^{x_2} \sum_{y=y_1}^{y_2} f(x,y) = I(x_2,y_2) - I(x_2,y_1-1) - I(x_1-1,y_2) + I(x_1-1,y_1-1)$$

Avec : $I(x,y)$ la somme de tous les $f(x,y)$ au dessus et à gauche du pixel (x,y).

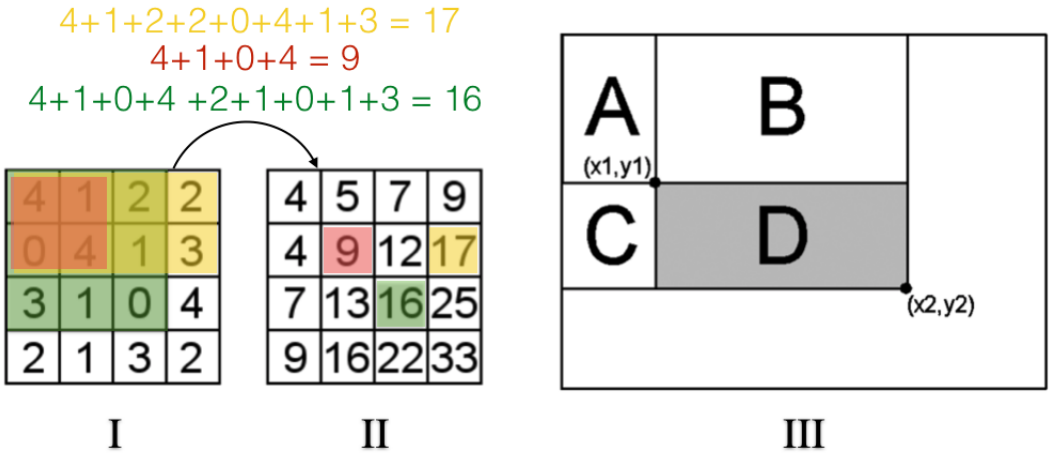


FIGURE 4.1 – Exemple de la méthode d'image intégrale[11]

A la fin, les *bounding boxes* contenant des piétons sont gardées et chacune possède un seuil de certitude. Celui-ci représente le pourcentage de probabilité que cette boite possède bien un piéton.

C. Suppression Comme expliqué précédemment, il arrive que plusieurs *bounding boxes* enveloppent le même piéton. Les boites les moins adaptées à celui-ci sont donc supprimées à l'aide de la méthode de *non-maximum suppression*.

Performances du détecteur

Une fois cet algorithme choisi, des tests ont été réalisés pour évaluer sa performance dans l'algorithme de suivi. Pour permettre cette analyse, trois critères entrent en compte :

A Qualité de la détection

B Qualité de la position de la détection

C Vitesse de l'algorithme

A. Qualité de la détection Comme mentionné précédemment, ce détecteur a de bonnes performances théoriques lors de tests sur des bases de données connues. Cependant, nous avons décidé de tester le détecteur sur la route du cas d'étude pour avoir une idée concrète de l'efficacité de celui-ci. Ces tests ont été réalisé en variant le paramètre seuil de certitude de présence d'une cible à l'intérieur de la *bounding box*. Cette étude, appelée étude ROC, est réalisée dans la section (5.1).

La distance de détection est un paramètre important à prendre en compte. La taille d'une boite minimale utilisée lors de la génération de candidats est de cent pixels (si le piéton est plus petit que cent pixels sur l'image alors celui-ci ne sera pas encadré correctement). A l'aide du modèle *pinhole* (présenté ci-après) et au vu de la caméra utilisée, les boites encadrant les piétons devraient encadrer correctement des piétons situés à 100 mètres maximum. Cette distance est amplement suffisante au vu de l'application d'analyse effectuée.

B. Qualité de la position de la détection L'algorithme utilisé permet l'obtention de la position en pixels du coin supérieur gauche de la boite dans laquelle un piéton se trouve ainsi que les dimensions de cette boite. Ces données concernant le piéton seront utilisées pour obtenir sa position en mètres dans le plan x-y (présenté ci-après). De manière générale, les *bounding boxes* encadrent bien le piéton (figure 4.2). Cependant, il y a des cas où la boîte n'encadre pas correctement le piéton (figure 4.3), cela engendrera des erreurs sur la position [3]. Nous verrons toutefois que ce problème sera atténué avec le traitement de données réalisé à la section (5.2).

FIGURE 4.2 – Cas d'une *bounding box* qui encadre bien le piéton

FIGURE 4.3 – Cas d'une *bounding box* qui n'encadre pas bien le piéton

C. Vitesse de l'algorithme La vitesse pour scanner une image avec ce détecteur est de 0.2 seconde. Pour réaliser le suivi, l'algorithme analyse la vidéo *frame* par *frame* et celle-ci est filmée à 30 *fps*. A titre d'exemple, une vidéo de trente secondes (900 *frames*) prendra trois minutes à être analysée. Ce qui représente six fois le temps réel. Dans ce cas d'étude, ce n'est pas contraignant car l'analyse n'est pas faite directement en temps réel mais c'est une contrainte à prendre en compte dans le cas d'une implémentation quelconque future du dispositif en temps réel.

3. Lors de l'utilisation du modèle *pinhole*, cela engendre une erreur de position en x-y (voir section 4.1.2)

4.1.2 Calcul de la position d'un piéton à l'aide du modèle *pinhole*

Toutes les positions exprimées en pixels doivent être transformées en mètres pour permettre l'analyse dans le plan x-y. Cette transformation s'effectue à l'aide du modèle de *pinhole*. L'utilisation de cette technique couplée à la méthode de détection aboutit à des erreurs de position dans le plan x-y, celles-ci sont présentées ci-après.

Description de la méthode

Ce modèle réalise une approximation du premier ordre de la transposition de la scène en trois dimensions à l'image en deux dimensions. Deux hypothèses sont à prendre en compte :

1. La caméra ne réalise pas de distorsion d'image
2. Les cibles filmées sont nettes, c'est à dire que la caméra réalise un bon *focus*

Dans notre cas, le dispositif est placé à hauteur d'homme et est perpendiculaire au sol, on peut alors schématiser le procédé (figure 4.4).

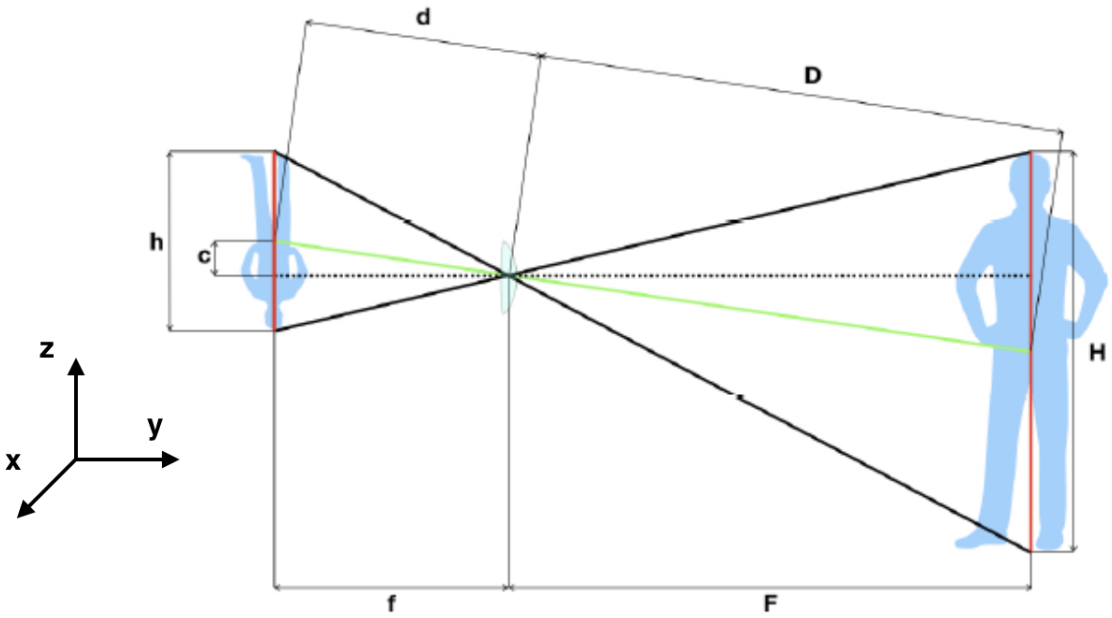


FIGURE 4.4 – Représentation schématique du modèle de *pinhole* [15]

Le but du modèle est de trouver la position du piéton dans le plan x-y. Pour obtenir ces distances, nous verrons qu'une inconnue manque aux équations. Il faut alors faire l'hypothèse forte que la taille d'un piéton analysé (H sur la figure 4.1.2) est approximativement de 1,80 mètres. Néanmoins, cette hypothèse contraignante pourrait être supprimée avec l'utilisation de deux caméras en stéréo (cette amélioration est proposée section (7.2)).

En connaissant f (la distance focale déterminée section 2.1.1), H (la hauteur d'un piéton fixée par hypothèse), c (la distance entre le centre de la boite et l'horizontal en pixels) et h (la hauteur de la *bounding box* renvoyée en pixels). On trouve alors la distance d à l'aide du théorème de Pythagore.

$$d = \sqrt{f^2 + c^2}$$

En appliquant Thalès sur les deux triangles on trouve la distance D.

$$D = \frac{H \cdot d}{h}$$

Il faut également obtenir l'angle azimutal à l'aide de la position horizontale en pixels (p_x) et la distance focale.

$$\theta_x = arctan(\frac{p_x}{f})$$

On obtient donc la position du piéton dans le plan x-y (en mètres) :

$$x = D \cdot sin(\theta_x)$$

$$y = D \cdot cos(\theta_x)$$

Erreur sur la position

Une fois l'algorithme de détection et le modèle du *pinhole* appliqués, la position du piéton en mètres est obtenue dans le plan x-y. Cependant, un modèle d'erreur globale doit être réalisé afin de prendre en compte les différentes imprécisions. Celles-ci sont dues à :

1. L'imprécision du détecteur : la boite encadrant le piéton n'est pas toujours précise (voir figure 4.3). Cela est dû à l'étape de suppression (section 4.1.1.C) qui n'est pas constante et peut parfois retourner des *bounding boxes* moins adéquates. Cette imprécision ne dépend pas de la distance à laquelle se trouve la cible, elle est aléatoire.
2. L'hypothèse de la taille d'un piéton pour le modèle *pinhole*. La hauteur d'un piéton a été fixée à 1.8 mètres, il arrive donc que la boite surestime ou sous-estime la taille des piétons faussant ainsi la position calculée.

Nous avons donc réalisé des tests pour obtenir cette erreur globale sur la position. Lors de ceux-ci, un piéton a été placé à différents endroits sur une étendue quadrillée [4] allant de 15 à 45 mètres en profondeur et sur une largeur de 16 mètres. Ce piéton s'est positionné à 24 endroits différents (voir figure 4.5) et trois photos ont été prises pour chaque position (trois *frames* par position [5]).

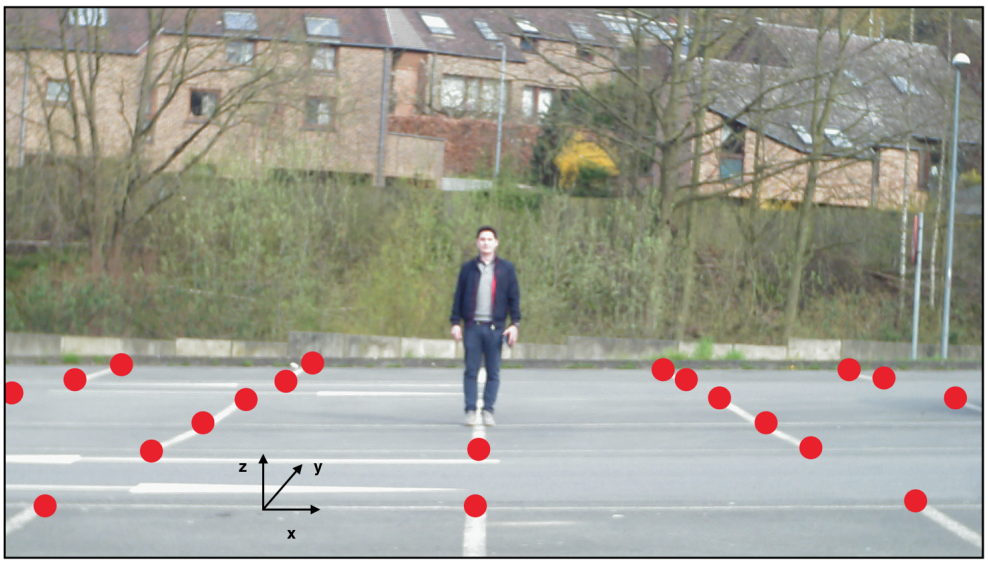


FIGURE 4.5 – Test pour le modèle d'erreur de la caméra : 24 positions différentes du piéton (trois photos ont été prises par position)

4. Parking
5. Cela permet de mieux rendre compte de la réalité car la caméra filme à du 30 *fps*

Ensuite, le détecteur et le modèle *pinhole* ont été appliqués sur chacune de ces 72 images. A chaque fois, pour un emplacement, trois positions ont été renvoyées par le détecteur appliqué sur les trois images de cet endroit. Nous avons donc calculé la moyenne de ces trois positions pour chaque emplacement. Connaissant la position réelle du piéton à tout endroit, on peut, maintenant, comparer la position moyenne renvoyée par le détecteur avec la position réelle pour chaque emplacement (figure 4.6). Sur cette illustration, le cas d'une boite encadrant parfaitement le piéton a été mis en évidence (photo de droite), ainsi que le cas d'une boite qui n'est pas correctement adaptée au piéton (photo de gauche).

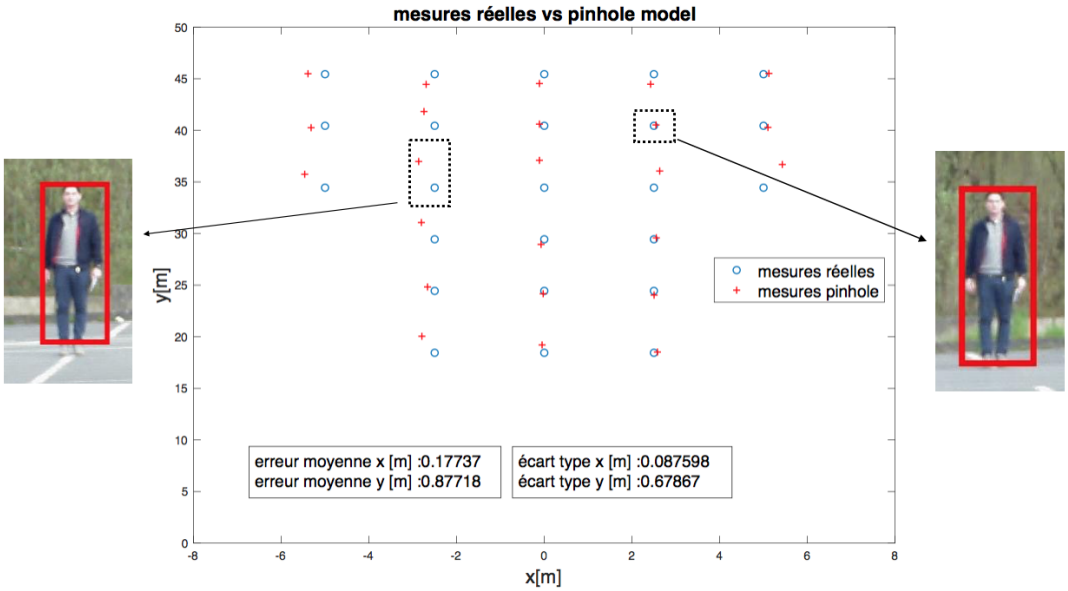


FIGURE 4.6 – Test pour le modèle d'erreur de la caméra : comparaison de la position réelle avec la position détectée pour les 24 emplacements

Dans un premier temps, l'erreur en x et en y pour chaque position détectée par rapport à la position réelle a été calculée. Ceci a donc permis d'obtenir la moyenne de ces erreurs (figure 4.6). En x, l'erreur moyenne est de 18 centimètres, ce qui est faible. Tandis qu'en y, une erreur de plus ou moins 90 centimètres est calculée. Ceci démontre une certaine imprécision du détecteur pour la profondeur.

Dans le but de représenter au mieux l'erreur de position, celle-ci a été modélisée à l'aide d'une fonction de densité de probabilité multivariable. Par conséquent, la déviation standard en x et en y a dû être obtenue. Pour ce faire, la déviation standard pour chaque ligne de positions [6] en x et en y a été évaluée. La moyenne de ces déviations standards pour toutes les lignes de positions en x peut être calculée (déviation standard en x) ainsi que celle pour toutes les lignes de positions en y (déviation standard en y). Grâce à ces déviations standards (proches de l'erreur moyenne), la distribution de l'erreur peut être représentée. Pour cela, on représente, dans le plan cartésien, la fonction de probabilité de densité de l'erreur autour d'une moyenne fixée en (0,0). Le but étant d'avoir une idée de cette erreur pour chaque position de détection (figures 4.7 et 4.8).

6. Nous définissons une ligne de positions comme étant une ligne reprenant différents emplacements. Par exemple, une ligne de positions en x reprend tous les emplacements se trouvant à un x constant et un y variable.

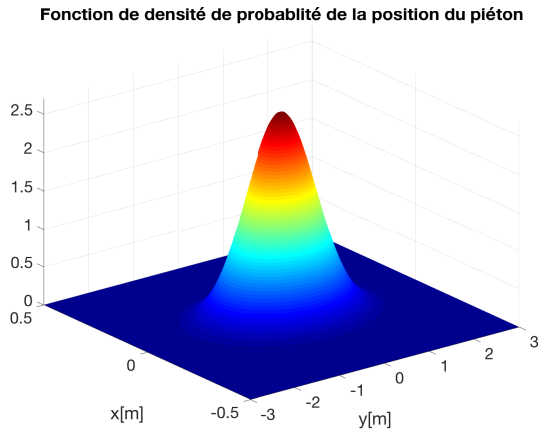


FIGURE 4.7 – Fonction de densité de probabilité de la position du piéton

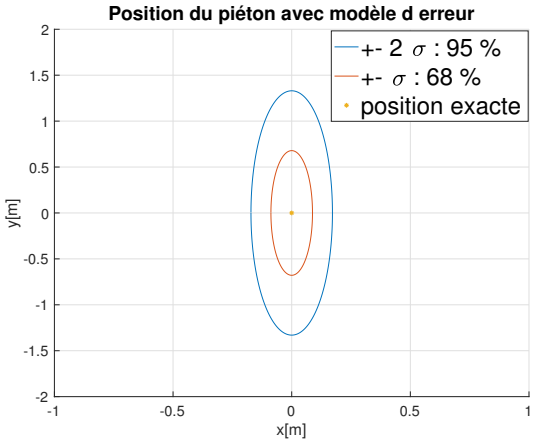


FIGURE 4.8 – Distribution de l'erreur autour de la position exacte

Comme représenté sur les graphes ci-dessus, on voit que l'erreur de position en x se trouvera dans 95% des cas ($[-2\sigma, 2\sigma]$) à 10 centimètres de part et d'autre de la cible. Tandis que l'erreur de position en y peut atteindre 1.2 mètres. Il est important de mentionner le fait que l'erreur sur les positions en x ne dépend pas de la position en y et inversement. On en conclut donc que les positions renvoyées par le détecteur en x sont très précises et ne doivent pas être corrigées. Cependant, au vu de la précision en y, nous devons effectuer des corrections pour permettre une bonne analyse de la trajectoire. Les améliorations présentées à la section (5.2) permettront de corriger cette erreur.

4.1.3 Détections de voitures

La détection de voitures se fait grâce au radar. Ses caractéristiques ont été présentées précédemment (section 2.2). Il est tout de même utile de rappeler, pour cette section, sa capacité à obtenir la position et la vitesse de multiples cibles sur un même *frame* (setion 3.2.2).

Paramètres utilisés

Le radar KMD2 permet une modification de quelques paramètres internes pour s'adapter au mieux aux gammes de distances et vitesses souhaitées. Dans la situation étudiée, nous avons besoin d'obtenir des résultats sur une distance de maximum 100 mètres et des vitesses allant jusqu'à 100 km/h. Pour ce faire, trois paramètres ont été modifiés :

— la bande passante : B
— la fréquence initiale : f_0
— le délai initial : t_{init}

La bande passante a été fixée à 384 MHz pour obtenir un range maximum adéquat. Cette valeur a été déterminée à l'aide de la formule suivante :

$$D_{max} = \frac{c\Delta t_{max}}{2} = \frac{Mc}{2B} = \frac{256 \cdot 3 \cdot 10^8}{2 \cdot 384 \cdot 10^6} = 100\ [m]$$

La fréquence initiale a été fixée à 23 933 MHz [7]. Cette valeur permet aux ondes envoyées de garder une bonne puissance d'émission dans la bande de fréquence utilisée. Le délai initial (figure 4.9) d'envoi entre chaque signaux (*chirps*) va permettre de changer le range de fréquence Doppler

7. Valeur proposée par RFbeam pour une distance de 100 mètres

pour obtenir une vitesse maximum de $[-100, 100]$ km/h soit $[-27.7, 27.7]$ m/s. La période d'une onde devient alors : $T = M\ T_s + t_{init}$. Finalement, le délai initial est fixé à $44.33\mu s$ [8] à l'aide de la formule suivante :

$$v_{rmax} = \pm\frac{\lambda}{4T} = \pm\frac{c}{4f}\frac{1}{M\ T_s + t_{init}} = \pm\frac{3 \cdot 10^8}{4 \cdot 24 \cdot 10^9 \cdot (256 \cdot 67.58 \cdot 10^{-6} + 44.33 \cdot 10^{-6})} = 27.7\,\text{m/s}$$

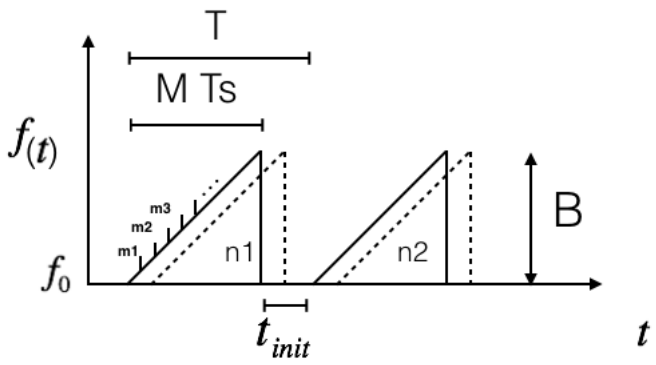


FIGURE 4.9 – Représentation du signal envoyé et reçu

Erreur sur les données radar

Une fois les paramètres adéquats du radar fixés, nous avons décidé d'analyser l'erreur sur la position ainsi que sur la vitesse.

Tout d'abord, en ce qui concerne la position, des tests assez restreints ont été effectués (annexe A2). En effet, ceux-ci sont complexes à réaliser de façon précise et ne sont donc pas assez représentatifs. Toutefois, les valeurs d'erreur obtenues sont plus petites que celles données par la fiche technique du radar (azimut : 0.1 degré , distance : 1 mètre (section 2.2)). Par conséquent, ce sont les résolutions données par la fiche technique qui doivent être prises en compte pour la position (prise de l'erreur la plus importante).

Ensuite, concernant la vitesse renvoyée par le radar, des tests ont également été effectués. Ceux-ci, contrairement à la position, ont pu facilement être réalisés et ce de façon précise. Dans ce cas, nous nous fierons donc directement aux résultats expérimentaux. Des mesures ont été prises pour une voiture venant vers le radar avec une vitesse fixe différente pour chaque test [9] : [40, 50 , 60, 70 et 80] km/h. Sur la figure (4.10), la vitesse de la voiture est affichée en fonction de sa distance par rapport au radar.

8. La résolution du radar étant de 22ns/step le t_{init} est en fait fixé à 2015 ($2015 \cdot 22\ 10^{-9} = 44.33\mu s$.)
9. Régulateur de vitesse (*cruise control* de la voiture)

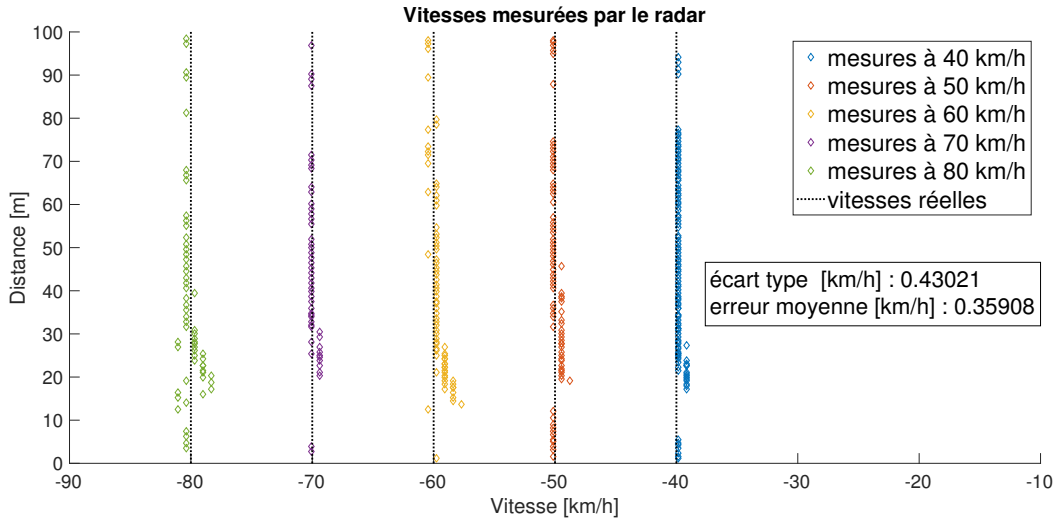


FIGURE 4.10 – Test du radar pour le modèle d'erreur de la vitesse

Sur ce graphe, il apparaît que les erreurs de vitesse sont plus élevées lorsque la voiture est proche du radar (de 10 à 40 mètres) et ce pour chaque cas. De plus, les erreurs sont également plus élevées pour des vitesses plus hautes. Malgré le fait que ces erreurs peuvent varier suivant la vitesse et la distance, nous avons décidé de réaliser une erreur globale. Ce choix est justifié par le fait que, sur la route étudiée, les voitures ont des vitesses très variables (allant de 5 à 70 km/h) et celles-ci sont mesurées quand elles sont proches et éloignées du radar.

L'erreur globale de vitesse est obtenue en moyennant l'erreur moyenne de chaque cas. Celle-ci vaut 0.36 km/h et se trouve en dessous de la résolution annoncée (selon la fiche technique du radar, la résolution est de ±1 km/h). Ensuite, la déviation standard a été calculée pour chaque test et la moyenne des ces déviations a pu être obtenue. Ci-dessous, la distribution de l'erreur globale de vitesse est représentée à l'aide d'une fonction de densité de probabilité (figure 4.11).

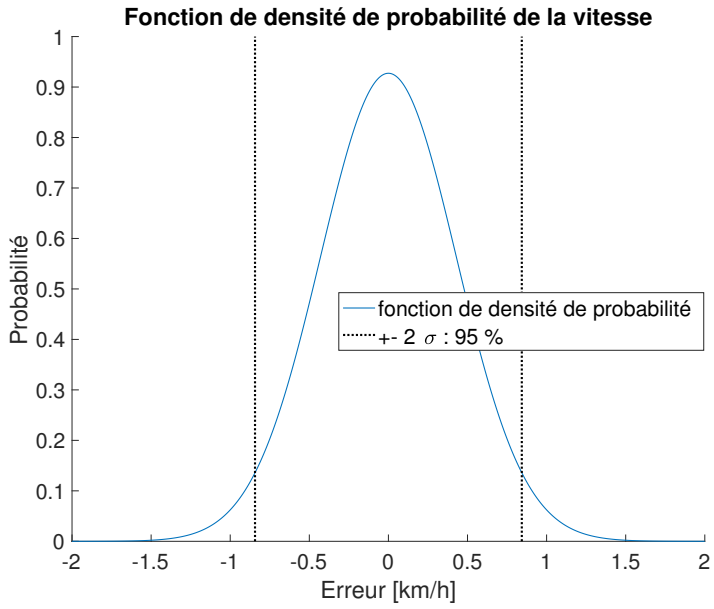


FIGURE 4.11 – Fonction de densité de probabilité de l'erreur de vitesse

Au vu de la gaussienne représentée ci-dessus, on observe donc que 95% des erreurs se situent entre [-0.86 ;0.86] km/h.

Finalement, on constate que le radar est performant en terme de précision aussi bien pour la vitesse que pour la position. Aucune modification ne devra donc être effectuée pour les corriger.

4.2 Algorithme de suivi

Une fois qu'une cible est détectée, il faut suivre celle-ci tout au long de sa présence dans le champ de vision du capteur. L'algorithme se répète sur chaque *frame* l'un à la suite de l'autre. Le but est de collecter des données (cibles potentielles) obtenues par les détecteurs et de rassembler ces données en un ensemble d'observations produites par la même cible [1][28]. L'algorithme se déroule comme suit :

Détection

Pour commencer, le *frame*[10] est analysé pour détecter la présence de piétons ou de voitures ainsi que leur position (section 4.1).

Création d'une piste

Il est important, à cette étape, de définir la notion de piste qui va être utilisée dans la suite du développement.

Piste : On définit une piste comme étant un historique temporel d'une cible depuis son entrée dans le champ de vision du capteur jusqu'à sa sortie. Cet historique est représenté comme une structure apparaissant au tableau (4.12) et contient[11] :

piste piéton	piste voiture
ID : un chiffre représentant l'ordre d'apparition de la piste (le compte est indépendant pour les piétons et les voitures)	
Bbox : historique des dimensions du cadre entourant le piéton et positions du centre du cadre pour tous les *frames* où la piste est présente (unité : pixels)	**Centroïdes** : historique des positions de la voiture pour tous les frames où la piste est présente (unité : mètres)
Filtre de Kalman (voir section 4.2.2) adapté au déplacement du piéton. Chaque piéton possède son propre filtre.	**Filtre de Kalman** (voir section 4.2.2) adapté au déplacement de la voiture. Chaque voiture possède son propre filtre.
Age : le nombre de *frames* de présence de la piste	
Temps : historique du temps auquel chaque position de la piste a été enregistrée (unité : secondes)	
/	**Vitesses** : historique des vitesses pour tous les *frames* où la piste est présente (unité : km/h)
totalVisibleCount : nombre de détections assignées à la piste lors de sa présence	
consecutiveInvisibleCount : le nombre de *frames* consécutifs où la piste n'a reçu aucune assignation de détections (ce nombre est remis à zéro dès que la piste reçoit une détection)	

FIGURE 4.12 – Structure représentant une piste

10. Pour la caméra, un *frame* correspond à une capture d'image à un instant t tandis que pour le radar il correspond à une prise de données pour un instant t.
11. Cet historique diffère légèrement entre un piéton et une voiture

Lors de cette étape, il y a donc création de pistes pour chaque détection indépendante. Ces détections sont des détections qui n'ont pas été assignées à une piste (l'assignation d'une détection à une piste est un raisonnement expliqué dans la suite de ce développement). En début d'algorithme il n'existe pas de pistes donc toutes les détections sont des détections indépendantes et deviennent donc des pistes à suivre.

Prédiction et assignation

Il s'agit maintenant de prédire la future position de chaque cible (piste). Pour cela, un filtre de Kalman est utilisé. Les différentes étapes de ce filtre sont expliquées en détail dans la section (4.2.2). Chaque piste a son propre filtre de Kalman adapté et celui-ci est ajusté au cours de l'évolution de la piste.

Après que la position de la piste ait été prédite, les données de positions des détections obtenues lors de l'étape de détection sont utilisées. Il faut assigner ces détections aux pistes c'est à dire qu'il faut décider si la détection d'une cible correspond ou pas à la même cible, détectée dans le frame précédent, qui s'est déplacée. Une piste n'a droit qu'à une seule ou aucune détection et une détection n'a droit qu'à une seule ou aucune cible (dans ce cas elle deviendra une piste à son tour lors de l'étape de création). L'algorithme permettant de faire ce choix est expliqué dans la section (4.2.1).

Actualisation des pistes

A partir d'ici, on constate deux types de pistes existantes : les pistes assignées et les pistes non-assignées.

1. Pistes assignées : ces pistes ont reçu une détection appropriée et donc leur position qui avait été prédite par le filtre est ajustée grâce à la détection en utilisant la correction du filtre. L'historique stocke donc la nouvelle position, le temps de celle-ci ainsi que sa vitesse (pour les voitures). Son nombre de visibilité (totalVisibleCount) et son âge sont incrémentés.
2. Pistes non-assignées : ces pistes n'ont pas reçu de détections. Par conséquent, leur position est la prédiction obtenue avec le filtre sans ajustement avec une détection. Leur âge ainsi que leur nombre consécutif de non-visibilité (consecutiveInvisibleCount) sont incrémentés.

Suppression des pistes

A la fin de l'algorithme, il est important de supprimer les pistes traquant un piéton qui est sorti du champ de vision des capteurs. Pour cela, on regarde au taux de non-visibilité consécutif de chaque piste (consecutiveInvisibleCount). Une piste qui n'a pas reçu d'assignation de détections pendant un certain nombre de *frames* consécutifs augmente sa probabilité d'être une piste suivant une cible qui a disparu et qui, par conséquent, doit être éliminée. On fixe alors un paramètre (invisibleForTooLong) qui va servir de seuil à ce nombre de non visibilité consécutive. Si le seuil est dépassé, la piste est supprimée de l'algorithme (c'est à dire qu'elle ne sera plus traquée) et son historique complet est enregistré. Il faut rester prudent quant à la détermination de ce paramètre car il arrive que les détecteurs ratent de vraies détections. Par conséquent, si ce paramètre est trop petit, des pistes toujours présentes dans le champ de vision vont être supprimées. Cependant, comme il est peu fréquent que les détecteurs ratent plusieurs fois d'affilé des détections lorsqu'une cible est présente dans le champ de vision, nous fixons, dans un premier temps, ce seuil à 10 *frames*.

Finalement, l'algorithme recommence à l'étape de création de pistes pour les détections qui n'ont pas été assignées à des pistes (ce sont de nouvelles cibles à suivre). Il est donc important que les détecteurs ne renvoient quasi pas de faux positifs car, dans ce cas, une fausse piste est

créée. Il y a donc un suivi de cible inexistante. L'algorithme recommence alors en analysant le *frame* suivant.

4.2.1 Algorithme Hongrois

Pour permettre l'assignation des détections aux pistes, on fait appel à la variante de l'algorithme Hongrois de James Munkres [34]. Le but de ce raisonnement est d'optimiser l'affectation, grâce à des valeurs numériques appelées coûts, d'une détection à une piste. D'un point de vue mathématique, on a r_{ij} le coût qui détermine la sensibilité d'affectation d'une détection d_i à une piste p_j. Cette sensibilité est définie par une distance. La fonction permettant d'obtenir la distance est expliquée à la section (4.2.1). Chaque détection d_i a donc une distance r_{ij} par rapport à chaque piste p_j. Le but est de trouver un compromis pour lier une détection à une piste avec une distance entre les deux la plus petite possible et ce pour toutes les pistes et détections existantes. Pour réaliser cela, l'algorithme Hongrois consiste donc à trouver dans la matrice coût (matrice avec les distances séparant chaque détection à chaque piste) un ensemble d'éléments indépendants [12] dont la somme est minimale par rapport à tous les autres ensembles de valeurs indépendantes existants dans cette matrice. L'exemple ci-dessous illustre cette explication.

Soit 4 pistes (p_1,p_2,p_3,p_4) et 4 détections (d_1,d_2,d_3,d_4) et la matrice r reprenant les distances détections-pistes.

	d_1	d_2	d_3	d_4
p_1	r_{11}	r_{12}	r_{13}	r_{14}
p_2	r_{21}	r_{22}	r_{23}	r_{24}
p_3	r_{31}	r_{32}	r_{33}	r_{34}
p_4	r_{41}	r_{42}	r_{43}	r_{44}

L'algorithme de James Munkres permet de trouver un ensemble de valeurs indépendantes (voir valeurs en jaune) dont la somme est minimum par rapport à tous les autres ensembles existants. On a donc un compromis pour l'assignation des détections aux pistes :

- La détection 1 est assignée à la piste 2 (d_2 - p_1)
- La détection 2 est assignée à la piste 1 (d_1 - p_2)
- La détection 3 est assignée à la piste 3 (d_3 - p_3)
- La détection 4 est assignée à la piste 4 (d_4 - p_4)

Grâce à cet algorithme, l'affectation des détections aux pistes se fait de manière optimale. Cependant, toutes les détections ne doivent pas toujours être assignées à une piste. C'est le cas de l'apparition d'une nouvelle cible dans le champ de vision et donc l'obligation de créer une nouvelle piste pour suivre cette cible. Pour gérer ce cas, il est indispensable de fixer un seuil de non-assignation [25] (costOfNonAssignement). Après avoir trouvé l'ensemble des valeurs indépendantes de la matrice coût pour permettre la liaison détection-piste, chaque valeur de la distance entre la détection et sa probable future piste est comparée à ce seuil. Si la valeur de la distance est plus petite que ce seuil alors la détection est assignée à la piste. Par contre, si la distance dépasse le seuil la détection ne sera pas assignée. On aura donc une piste qui n'a pas reçu de détection (prédiction de sa future position sans détection) et une détection non-assignée qui créera alors une nouvelle piste.

12. Des éléments d'une matrice sont dits indépendants si aucun de ceux-ci ne se trouve à la même ligne ou à la même colonne.

Il est important, ici, de mentionner le risque d'échange de filtres lié à cette distance seuil. On définit un échange de filtres de la manière suivante : lorsque deux cibles se trouvent proches l'une de l'autre (se croisent, se suivent,...), celles-ci sont traquées à l'aide de deux pistes. Il arrive que ces deux pistes échangent de cibles. Ceci est dû aux deux détections qui ne sont pas assignées à la bonne piste car celles-ci sont trop proches. C'est donc pour cela que fixer ce paramètre de distance seuil n'est pas simple. Il ne peut être trop faible car cela mènerait à la création de nouvelles pistes à chaque détection. Il ne peut pas non plus être trop élevé sinon ce cas d'échange de filtres serait fréquent. C'est pour cela qu'un compromis a du être réalisé et ce expérimentalement dans un premier temps.

Suite à cette analyse, il faut noter l'importance du choix de la matrice coût. En effet, celle-ci est l'élément central pour l'affectation correcte d'une détection à une piste. Cette matrice coût est donc définie par une distance particulière.

Choix de la matrice coût : distance

Le but, ici, est de déterminer la matrice coût la plus appropriée à notre cas de suivi de cibles multiples. Etant donné l'utilisation d'un filtre de Kalman pour la prédiction (section 4.2.2), il est intéressant de lier ce coût aux paramètres du filtre. La fonction distance utilisée pour permettre ce lien est la suivante [21] [26] :

$$d(z) = (z - \mathrm{H}x)^{\mathrm{T}} \sum^{-1} (z - \mathrm{H}x) + ln|\sum|$$

Dans cette équation apparaît le terme $(z - \mathrm{H}x)$ représenté par :

— z : le vecteur de mesures prises par le senseur (détections)
— x : le vecteur des prédictions réalisées par le filtre de Kalman de la piste
— H : la matrice de transformation d'échelle

et le terme $\sum = \mathrm{HPH}^{\mathrm{T}} + \mathrm{R}$ avec :

— P : la matrice de covariance d'état
— R : la matrice de covariance du bruit interne au capteur

Cette fonction de distance permet d'obtenir une méthode optimale pour mesurer la sensibilité d'affectation d'une détection à une piste. Elle prend en compte la covariance de l'état prédit et le bruit interne au capteur.

4.2.2 Filtre de Kalman

Afin de réaliser le suivi de piétons et de voitures il est indispensable d'utiliser des filtres qui permettent une représentation plus proche de la réalité. En effet, tout capteur possède des imprécisions qui donnent lieu à des erreurs. Ces filtres de suivi permettent alors de prendre en compte les irrégularités et d'exprimer de manière stochastique la position d'une cible.

Dans le cadre de notre étude, nous avons décidé d'utiliser le filtre de Kalman. En effet, comme précisé dans la section (3.1.1), ce dernier possède les avantages suivants :

— Implémentation simple
— Seuls l'état précédent ainsi que la mesure actuelle sont nécessaires à l'estimation d'une nouvelle position
— Temps d'exécution rapide
— La connaissance de la moyenne et de la variance suffisent à décrire le modèle d'incertitude

Après avoir testé le filtre de Kalman et le filtre à particules, on a pu constater que le filtre de Kalman possédait les capacités suffisantes pour l'application de suivi multiple de piétons et de voitures pour le cas d'étude analysé. La connaissance relative du modèle dynamique des piétons (traversent de gauche à droite ou inversement) et des voitures (suivent la route) nous a poussé à l'utilisation de ce filtre. Le filtre à particules possède une plus grande adaptabilité (voir section 3.1.2) mais nécessite un temps d'exécution bien supérieur pour au final procurer une précision de même niveau que son homologue, le filtre de Kalman. Le choix de ce type de filtre a également été encouragé par la possibilité de décrire l'incertitude du capteur et du bruit environnemental par une distribution gaussienne. Celle-ci permet de bien décrire le modèle, il n'est donc pas nécessaire d'utiliser d'autres distributions plus complexes comme le permet le filtre à particules.

Dans la section suivante, l'application de ce filtre sera expliquée de manière plus mathématique. Ensuite, les choix des paramètres utilisés dans notre algorithme seront détaillés.

Implémentation mathématique

Le filtre de Kalman est basé sur un modèle dynamique adaptatif appliqué à un vecteur d'état. Afin de suivre au mieux la cible, l'algorithme de Kalman se décompose en trois étapes :

1. Prédiction
2. Détection
3. Mise à jour

1. Lors de la première étape, il faut prédire le nouvel état du vecteur $\hat{\mathbf{x}}_k$ qui, dans le cas du suivi de piétons et de voitures, représente la position (x,y) et la vitesse (u,v) :

$$\mathbf{x} = \begin{bmatrix} x \\ u \\ y \\ v \end{bmatrix}$$

L'état est également défini à l'aide d'une matrice de covariance P_k qui représente le degré de corrélation entre chaque variable d'état. Cette première étape utilise la matrice de prédiction F_k qui représente la transition entre un état à $t-1$ jusqu'à t (exemple : avancer la position de la cible en fonction de la vitesse reçue et du delta temps entre deux nouvelles détections) afin de prédire le nouveau vecteur d'état et la nouvelle matrice de covariance (figure 4.13) :

$$\hat{\mathbf{x}}_k = F_k \cdot \hat{\mathbf{x}}_{k-1}$$

$$P_k = F_k \cdot P_{k-1} \cdot F_k^T$$

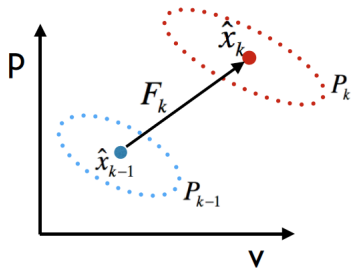


FIGURE 4.13 – Exemple de l'étape de prédiction pour un cas simple (abscisse = vitesse , ordonnée = position)

Après la prédiction réalisée, la matrice Q_k, représentant l'incertitude que peut apporter le milieu extérieur, est ajoutée à la matrice de covariance. Elle peut être vue comme une matrice traduisant le bruit environnemental :

$$P_k = F_k \cdot P_{k-1} \cdot F_k^T + Q_k$$

2. La seconde étape consiste à transposer la prédiction dans l'échelle adéquate au capteur utilisé (caméra-pixels, radar-mètres). Pour ce faire, la matrice H_k est utilisée (figure 4.14) :

$$\widehat{\mathbf{x}'}_k = H_k \cdot \widehat{\mathbf{x}}_k$$

$$P'_k = H_k \cdot P_k \cdot H_k^T$$

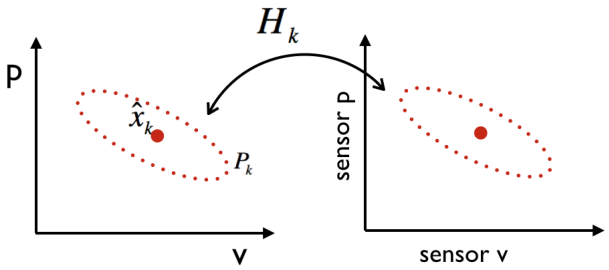


FIGURE 4.14 – Représentation de la transition d'échelle

Nous allons ensuite incorporer la mesure retournée par le capteur (caméra ou radar) z_k et y ajouter l'incertitude R_k qui représente la matrice de covariance du bruit interne au capteur (figure 4.15).

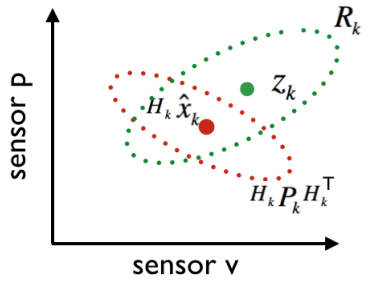


FIGURE 4.15 – Représentation de la détection et de la prédiction

3. Finalement, lors de la troisième étape, on recalcule une nouvelle moyenne du vecteur d'état et de la matrice de covariance en utilisant la multiplication de l'état prédit et détecté. Dès lors, la mise à jour des vecteurs $\widehat{\mathbf{x}'}_k$ et P'_k est obtenue (figure 4.16). On a donc la nouvelle position estimée.

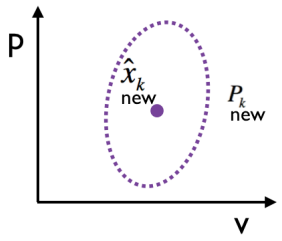


FIGURE 4.16 – Représentation de la mise à jour

Les trois étapes se répètent alors à chaque nouvelle détection.

Détermination des paramètres

Une étape importante dans l'utilisation d'un filtre de Kalman est la détermination des matrices décrivant l'incertitude du système et la transition d'état. En effet, si celles-ci ne rendent pas compte de la réalité, l'erreur des estimations peut ne pas converger assez rapidement. Ici, les paramètres choisis pour la caméra ainsi que pour le radar sont présentés.

Caméra :

Matrice de prédiction F_k et vecteur d'état La matrice traduisant l'évolution des positions a été déterminée de manière logique. En effet, pour décrire la position d'un piéton, un mouvement à vitesse constante est considéré. On obtient l'équation linéaire de transition tel que :

$$\hat{\mathrm{x}}_k = \hat{\mathrm{x}}_{k-1} + \Delta t \cdot \mathrm{u}$$

$$\hat{\mathrm{y}}_k = \hat{\mathrm{y}}_{k-1} + \Delta t \cdot \mathrm{v}$$

Soit une matrice de prédiction :

$$F_k = \begin{pmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Nous avons décidé de travailler directement dans l'échelle du capteur (H_k = matrice identité). Les positions sont donc exprimées en pixels.

Comme précisé dans la section (2.1), la caméra fonctionne à du 30*fps*. Le Δt entre chaque nouvelle détection est alors de 1/30 de seconde. Après un ensemble de tests, il a été décidé de prendre comme vitesse horizontale 1.2 m/s (vitesse moyenne d'un piéton traversant une route). Le signe de cette vitesse sera adapté au sens de traversée ; si le piéton traverse de droite à gauche, la vitesse sera négative et inversement. C'est grâce à l'historique des déplacements d'une piste (section 4.2) que le signe de cette vitesse sera fixé. Effectivement, l'historique de positions sera vérifié chaque seconde pour analyser le sens de traversée du piéton et par conséquent adapter le sens de sa vitesse. Afin de respecter les unités, il faut exprimer la vitesse en pixels par seconde. La résolution de notre caméra étant de 1920x1080 et la largeur de la route filmée étant de 12 mètres, on approxime cette vitesse à :

$$u = \frac{1920}{12} \cdot 1.2 = 192 \quad [\mathrm{px/s}]$$

La vitesse verticale étant proche de 0, celle-ci a été fixée à $v = 10$ px/s.

Matrice de covariance P_k Les relations d'état étant linéaires, la covariance de deux variables peut être obtenue comme étant égale au produit de leurs écarts types. On vérifie cela à l'aide du développement mathématique suivant :

$$\begin{aligned}
&\text{Si : } Y = a \cdot X + b \\
&\text{Alors : } Cov(X,Y) = Cov(X, aX+b) = a \cdot Cov(X,X) + Cov(X,b) \\
&\qquad\qquad = a \cdot Var(X) = (a\sigma_X) \cdot \sigma_X \\
&\qquad\qquad = \sigma_Y \cdot \sigma_X
\end{aligned}$$

Il faut alors estimer l'écart type de nos variables d'état afin de construire la matrice P_k. Par simplicité, la valeur de l'écart type sera interprétée comme étant l'erreur moyenne Δ. Après un ensemble de tests, nous avons pris les paramètres suivants :

$$\Delta\mathbf{x} = \begin{pmatrix} \Delta x \\ \Delta u \\ \Delta y \\ \Delta v \end{pmatrix} = \begin{pmatrix} 5 \\ 16 \\ 2 \\ 1 \end{pmatrix} \rightarrow P_k = \begin{pmatrix} (\Delta x)^2 & \Delta x \Delta u & 0 & 0 \\ \Delta u \Delta x & (\Delta u)^2 & 0 & 0 \\ 0 & 0 & (\Delta y)^2 & \Delta y \Delta v \\ 0 & 0 & \Delta v \Delta y & (\Delta v)^2 \end{pmatrix}$$

Matrice de covariance du bruit extérieur Q_k Le bruit environnemental étant faible avec la caméra fixe, la covariance des variables d'état est considérée comme indépendante. Nous avons fixé, expérimentalement, un bruit de 10 pixels sur les positions et de 7.0711 pixels/s sur les vitesses. Cela donne la matrice de covariance suivante :

$$Q_k = \begin{pmatrix} 100 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 50 \end{pmatrix}$$

Matrice du bruit de la mesure R_k A l'aide des tests sur le détecteur effectués à la section (4.1.1), les écarts-types de la position en pixels ont pu être approximés. La matrice R_k peut alors être construite en considérant une corrélation nulle entre la position horizontale et verticale :

$$\begin{aligned}
\sigma_x &= 0.087593\ [m] \rightarrow \pm 10\ [\text{px}] \\
\sigma_y &= 0.67867\ [m] \rightarrow\ \pm 80\ [\text{px}] \\
R_k &= \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix} = \begin{pmatrix} 100 & 0 \\ 0 & 6400 \end{pmatrix}
\end{aligned}$$

Radar :

Matrice de prédiction F_k et vecteur d'état A nouveau, la matrice F_k a été déterminée pour décrire un mouvement linéaire d'une voiture en considérant, cette fois, une vitesse adaptée à chaque nouvelle détection. En effet, le capteur renvoie la position (x, y) et la vitesse (u, v) que l'on peut utiliser dans le vecteur d'état afin d'améliorer la précision de la prédiction. Le radar fonctionnant à du 15 *fps*, le Δt est égale à 1/15 de seconde. La matrice de prédiction prend la même forme que celle définie pour la caméra.

Matrice de covariance P_k Le radar renvoie directement les données en mètres et mètres par seconde. Les écarts types sont donc exprimés dans les mêmes unités afin d'obtenir, comme pour la caméra, la matrice de transformation H_k égale à la matrice identité. Les écarts types sont à nouveau, par soucis de simplicité, considérés comme l'erreur moyenne. Après un ensemble de tests, les paramètres suivants ont été déterminés :

$$\Delta\mathbf{x} = \begin{pmatrix}\Delta x\\ \Delta u\\ \Delta y\\ \Delta v\end{pmatrix} = \begin{pmatrix}2\\ 1\\ 2\\ 1\end{pmatrix} \rightarrow P_k = \begin{pmatrix}(\Delta x)^2 & \Delta x\Delta u & 0 & 0\\ \Delta u\Delta x & (\Delta u)^2 & 0 & 0\\ 0 & 0 & (\Delta y)^2 & \Delta y\Delta v\\ 0 & 0 & \Delta v\Delta y & (\Delta v)^2\end{pmatrix}$$

Matrice de covariance du bruit extérieur Q_k Lors de la détection de voitures, il est possible que les données renvoyées par le radar correspondent de temps à autre à l'avant de la voiture, à l'arrière ou encore aux cotés du véhicule. Il faut dès lors prendre en compte une position variable de la voiture même si ces détections correspondent à la même cible. Les données radar étant correctes, c'est la disposition de la voiture qui va déterminer quelle facette sera *flashée*. Il s'agit donc bien d'un bruit environnemental qui ne dépend pas du capteur. En réalisant plusieurs tests, il a été déterminé une différence moyenne de 3.16 mètres en position. Le bruit concernant la vitesse est assez faible et sera fixé à 1.41 mètres par seconde. Les différentes variables sont considérées pour le bruit extérieur comme étant indépendantes. On obtient alors la matrice suivante :

$$Q_k = \begin{pmatrix}10 & 0 & 0 & 0\\ 0 & 2 & 0 & 0\\ 0 & 0 & 10 & 0\\ 0 & 0 & 0 & 2\end{pmatrix}$$

Matrice du bruit de la mesure R_k Avec la connaissance de l'erreur de distance et de l'angle azimutal du radar (section 4.1.3), l'erreur de la position en x et en y peut être calculée. Celle-ci varie en fonction de la distance de la voiture par rapport au radar. Elle sera donc recalculée à chaque itération tout comme la matrice R_k. On définit les erreurs Δx et Δy comme suit :

$$\Delta x = max(\mid d \cdot sin(\theta) - (d \pm 1) \cdot sin(\theta \pm 0.1^o) \mid)$$

$$\Delta y = max(\mid d \cdot cos(\theta) - (d \pm 1) \cdot cos(\theta \pm 0.1^o) \mid)$$

avec d qui vaut la distance et θ l'angle azimutal. A nouveau, on approxime les écarts types comme étant l'erreur Δ et on obtient R_k :

$$R_k = \begin{pmatrix}(\Delta x)^2 & \Delta x\Delta y\\ \Delta y\Delta x & (\Delta y)^2\end{pmatrix}$$

La détermination de l'ensemble de ces matrices décrivant l'incertitude de l'environnement et des capteurs ainsi que le modèle dynamique nous permettront un suivi optimal.

4.3 Fusion des données

Pour réaliser une étude de la route, il est indispensable d'avoir le comportement des voitures et des piétons au même moment. Pour cela, le lancement des deux capteurs se fait simultanément lors de la prise de données. Il a fallu également s'assurer de l'alignement parfait de la visée des deux dispositifs (voir plan annexe A1). En effet, cela pourrait avoir un impact considérable sur la position d'un piéton par rapport à une voiture dans le plan x-y. Ensuite, étant donné que le fichier des données du radar renvoie le temps précis auquel les capteurs ont été lancés, c'est ce temps qui servira de temps initial de référence pour la caméra. La vidéo étant filmée à du 30 *fps*, lors de l'analyse de chaque *frame*, le temps stocké est incrémenté de 1/30 de seconde. Tandis que pour chaque *frame* du radar, le temps est connu car il a été enregistré à la prise de données.

Enfin, les algorithmes de suivi expliqués précédemment sont lancés pour chaque ensemble de données (données radar et vidéo caméra) indépendamment. Après avoir effectué le suivi, les données sont assemblées et triées chronologiquement. On représente notre méthode de fusion sur le schéma ci-dessous (figure 4.17).

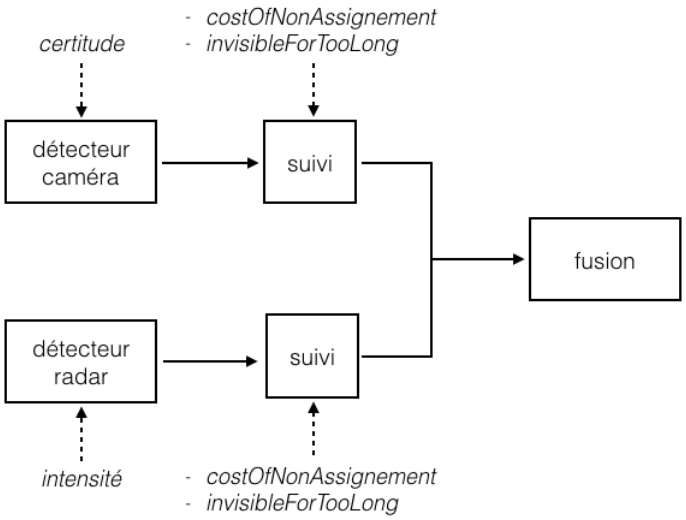


FIGURE 4.17 – Étapes de l'algorithme de fusion (premier essai)

Dès lors, ce tri va nous permettre à l'aide de structures dynamiques sous Matlab, d'afficher graphiquement les déplacements des piétons et des voitures à tout instant comme illustré ci dessous (figure 4.18). Sur ce graphe dynamique, il apparaît, en pointillé rouge, l'angle de détection des capteurs (30°), les deux lignes bleues sont les délimitations de la route principale et les deux lignes parallèles en noir représentent la route transversale par laquelle les piétons traversent. Pour les piétons et voitures, un historique de leurs positions est affiché. Le point sur lequel se trouve l'affichage de son id est la dernière position temporelle de la cible.

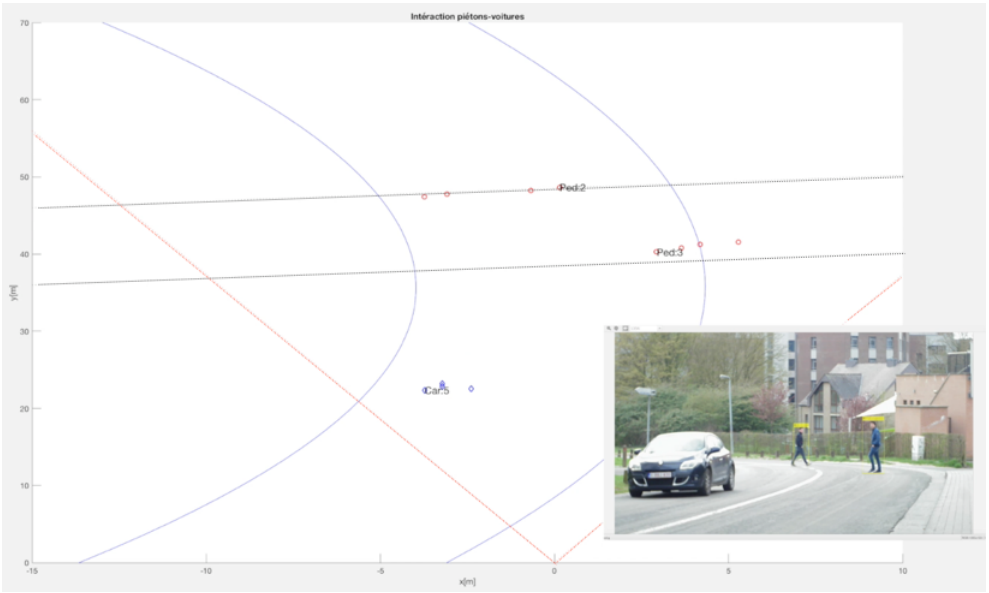


FIGURE 4.18 – Représentation de l'affichage dynamique de l'interaction piétons-voitures (rouge : angle de visée, bleu : route principale, noir : route transversale)

Pour permettre cette fusion de données, trois modifications concernant des paramètres de l'algorithme ont été effectuées.

La première modification à effectuer concerne le seuil de non-visibilité consécutif pour les piétons. Celui-ci avait été fixé à 10 *frames* précédemment (voir section 4.2) pour de simples tests où le manque consécutif de détections était rare. Hors, pour les tests de fusion, il s'est avéré que, souvent, une voiture passait devant la caméra. Par conséquent, un piéton suivi à ce moment disparaissait du champ de vue. Pour garder la piste existante pendant ce court laps de temps, le taux de non-visibilité consécutif doit être augmenté. Cet instant dure maximum deux secondes au vu de la vitesse des voitures, ce qui représente 60 *frames* de caméra. Nous avons donc fixé ce paramètre invisibleForTooLong à 60 *frames* pour les piétons.

La deuxième modification à apporter se situe à nouveau au niveau du seuil consécutif de non-visibilité mais, cette fois-ci, pour les voitures. Lors de tests nous avons remarqué qu'il était fréquent que deux voitures se suivent. Dans ce cas, il y a donc une voiture qui en cache une autre par rapport au radar. Cela signifie que la piste cachée n'obtient pas de détections pendant ce laps de temps. Elle est donc prédite et elle risque ensuite d'être perdue. Étant donné que cet instant ne dépasse jamais une seconde, nous avons fixé le seuil consécutif de non-visibilité à 15 *frames* de radar pour palier à ce problème pour les voitures.

La dernière modification réalisée, et ce dans le but d'avoir une meilleure visualisation du tracé des voitures, est l'affichage des positions prédites par le filtre de Kalman lorsqu'il n'y a pas de détection assignée à la piste. En effet, pour les piétons, il y a uniquement les détections qui sont affichées car afficher les prédictions n'est pas utile pour visualiser leur trajectoire. Tandis que pour les voitures, le cas d'une voiture masquée du radar par une autre, empêche fortement la bonne visualisation de la trajectoire de la voiture cachée. En affichant les positions prédites, cela permet d'avoir le déplacement complet de la voiture masquée.

4.4 Tests et résultats

Dans cette section, des tests réalisés pour cette implémentation préliminaire de l'algorithme sont présentés. Ces tests ont été indispensables pour permettre de multiples améliorations pour l'application finale. Ils nous ont permis de visualiser les défauts principaux et de trouver des solutions pour y remédier.

Dans un premier temps, les tests sur le *tracking* de piétons seront présentés et suivis des tests sur le *tracking* de voitures. Finalement, un premier essai de la fusion de données sur le cas d'étude sera discuté.

4.4.1 Résultats du suivi de piétons

Pour permettre d'analyser les performances de l'algorithme de suivi implémenté pour les piétons, des tests de suivi d'abord pour une cible individuelle puis pour deux cibles en même temps ont été réalisés. Les performances analysées, ici, étaient :

— La précision de la position
— L'exactitude de la trajectoire
— L'exactitude du nombre de piétons présents

Les différents paramètres discutés précédemment ont été fixés comme suit :

costOfNonAssignement	40
invisibleForTooLong	60 *frames*
seuil de certitude du détecteur	60 %

Table 4.1 – Paramètres fixés pour le suivi de piétons

Le premier paramètre a été fixé en réalisant des tests (vérification qu'il n'y ait pas échange de filtres (voir section 4.2.1)). Le second paramètre a été déterminé suivant ce qui a été dit précédemment. Le troisième a également été fixé expérimentalement.

Nous avons donc réalisé quatre types de mesure pour tester les performances de notre algorithme. Ces tests sont représentatifs du cas d'étude analysé. Parmi ceux-ci, il y a :

Cible unique

1. Traversée d'un piéton
2. Carré effectué par un piéton

Cibles multiples

3. Traversée de deux piétons (croisement)
4. Tracé en forme de **L** effectué par chaque piéton

Ces quatre tests ont été effectué sur un vaste espace vide [13]. Les piétons se trouvent entre 35 et 45 mètres de distance de la caméra. Ces différents tests sont présentés ci-dessous :

13. Terrain de foot

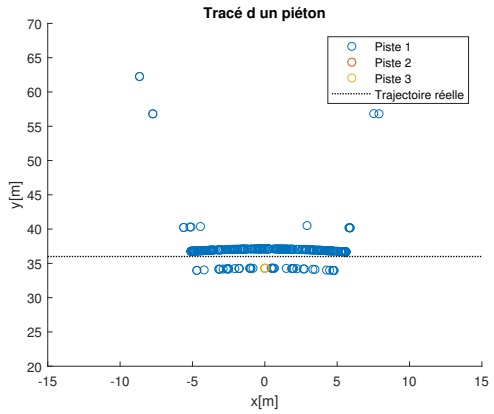


FIGURE 4.19 – Test 1 : Traversée d'un piéton (piste 1), les autres pistes sont des faux positifs

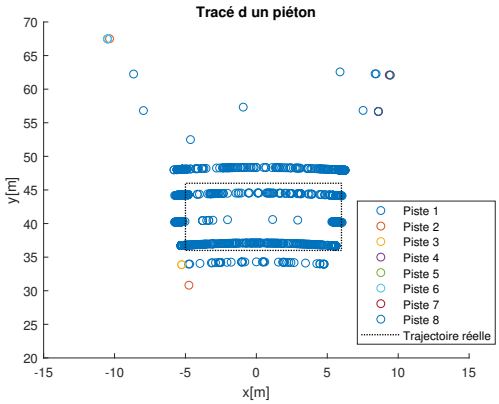


FIGURE 4.20 – Test 2 : Carré effectué par un piéton (piste 1), les autres pistes sont des faux positifs

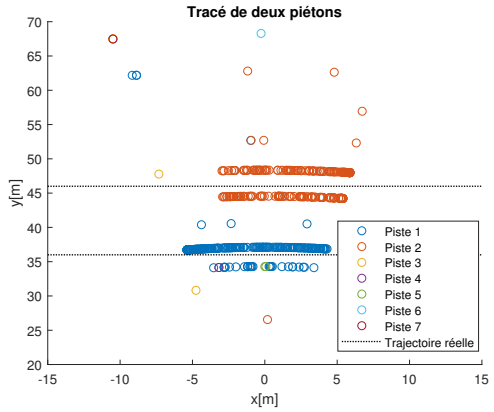


FIGURE 4.21 – Test 3 : Croisement de deux piétons (piste 1 et 2), les autres pistes sont des faux positifs

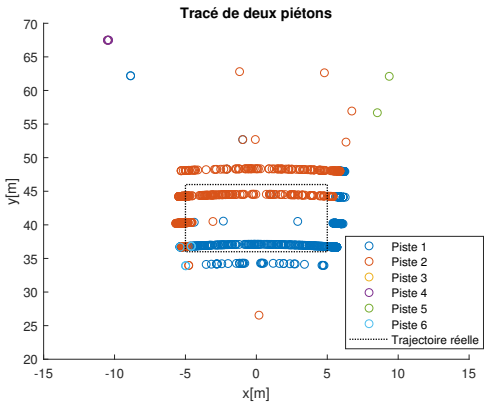


FIGURE 4.22 – Test 4 : forme de **L** effectué par 2 piétons (piste 1 et 2), les autres pistes sont des faux positifs

Discussion des résultats

Tout d'abord, en ce qui concerne la position, celle-ci n'est pas précise. Les erreurs de position obtenues à la section (4.1.1) varient autour de la trajectoire réelle du piéton. Pour rappel, celles-ci sont dues aux boîtes du détecteur qui n'encadrent pas correctement ce dernier. Cette variation de taille de boîtes engendre des "sauts" de un à deux mètres en y.

Ensuite, l'une des priorités de l'algorithme étant d'effectuer le comptage de piétons, un problème majeur apparaît : de fausses détections ont créé de fausses pistes. Celles-ci ne sont donc pas des piétons et sont pourtant prises en compte dans le comptage final.

Ces problèmes seront résolus à l'aide de traitements de données à la section (5.2).

4.4.2 Résultats du suivi de voitures

Les performances à analyser pour l'algorithme de suivi des voitures sont l'exactitude de la trajectoire ainsi que l'exactitude du comptage de voitures analysées. Pour ce qui est de la précision de la position et de la vitesse, il a été démontré à la section (4.1.3) que les mesures prises par le radar sont suffisamment précises (rappel fiche technique du radar section 2.2). Les paramètres fixés précédemment sont les suivants :

costOfNonAssignement	10
invisibleForTooLong	15 *frames*
intensité	20

TABLE 4.2 – Paramètres fixés pour le suivi de voitures

Le premier paramètre a été fixé en réalisant des tests (vérification qu'il n'y ait pas échange de filtres (voir section 4.2.1)). Le second paramètre a été déterminé suivant ce qui a été dit précédemment. Le troisième a également été fixé expérimentalement. A nouveau, nous avons réalisé trois tests de cas fréquents apparaissant sur la route étudiée. Parmi ceux-ci il y a :

1. Aller simple d'une voiture
2. Croisement de deux voitures
3. Suivi de deux voitures

Ces trois tests ont été réalisés sur la route étudiée, ils sont représentés ci-dessous à certains moments lors de l'affichage dynamique :

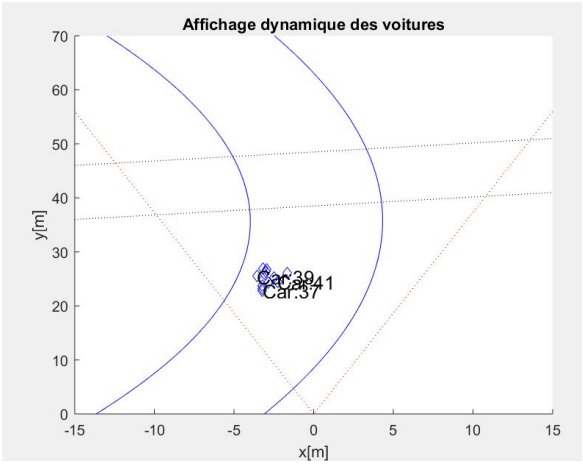


FIGURE 4.23 – Test 1 : Aller simple d'une voiture

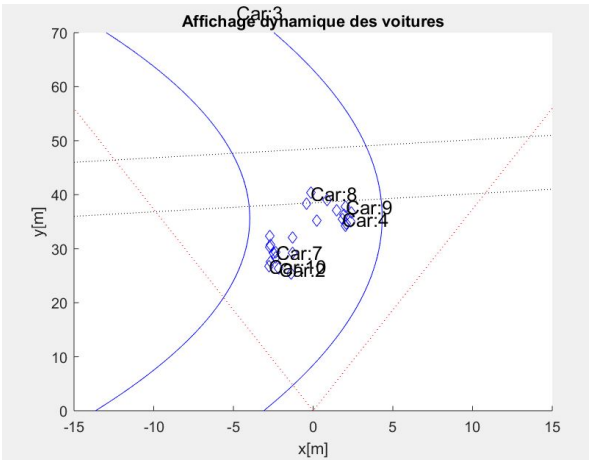


FIGURE 4.24 – Test 2 : Croisement de deux voitures

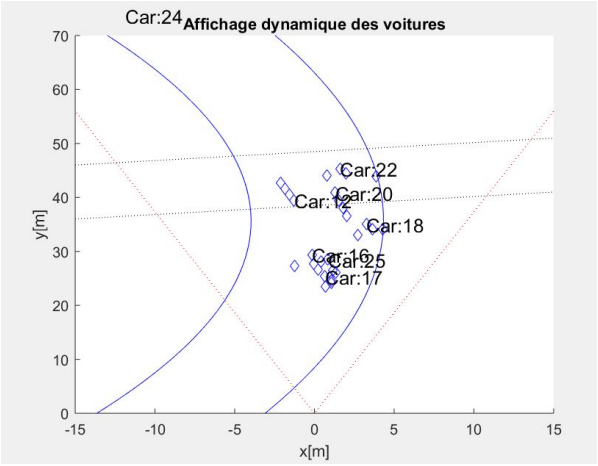


FIGURE 4.25 – Test 3 : Suivi de deux voitures

Discussion des résultats

Pour ce qui est de l'exactitude de la trajectoire, on constate que les voitures étaient bien suivies à tout instant tout au long de leur passage sur la route. Ceci se voit notamment à l'examen du croisement et au suivi de deux voitures.

Cependant, le problème majeur du mauvais comptage revient. En effet, comme on peut le voir ci-dessus, une voiture est représentée par trois à quatre pistes. Ce problème (déjà présent chez les piétons) est accentué ici car le radar obtient de multiples détections pour un même véhicule lors d'un même *frame*. L'algorithme de suivi ne pouvant assigner qu'une seule détection par piste, cela amène à des créations de pistes supplémentaires pour la même voiture.

Ce problème sera résolu à la section (5.3) à l'aide de traitements de données.

4.4.3 Résultats de la fusion

Finalement, des tests sur la fusion de données ont été réalisés. Dans cette section, étant donné que les différentes performances de chaque algorithme de suivi ont déjà été examinées, nous étudions principalement la performance de l'affichage des piétons et des voitures simultanément à la bonne position pour chaque cible dans le plan x-y. Un test a donc été effectué sur la route étudiée. Ci-dessous, on montre une illustration de celui-ci (figure 4.26).

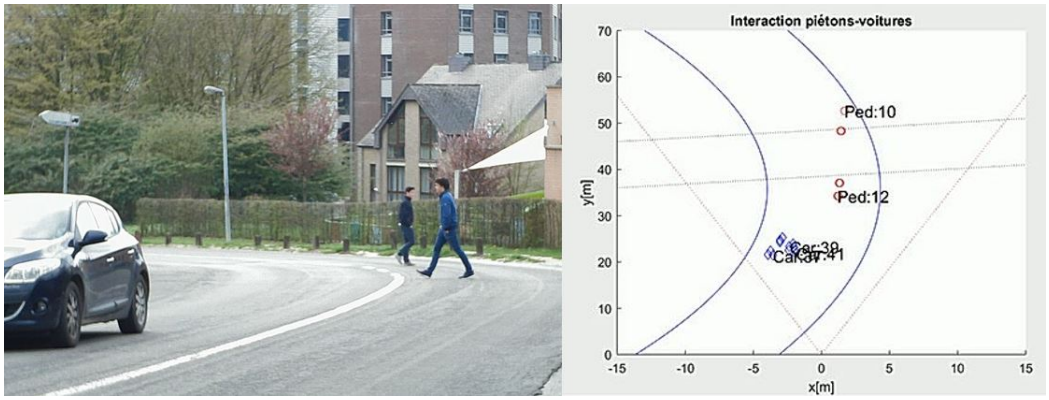


FIGURE 4.26 – Test : fusion des données sur le carrefour étudié

Discussion des résultats

Nous pouvons observer qu'il est possible d'obtenir une analyse entre les piétons et les voitures. L'objectif principal est d'avoir une interaction piétons-voitures. Celle-ci dépend donc du bon positionnement de chaque type de cible à tout instant. Dans ce test, le temps d'affichage est correct pour les voitures et les piétons. L'analyse montre également que les positions approximatives (il y a toujours les erreurs de positionnement) des piétons par rapport aux voitures sont bonnes dans le plan x-y. Ce qui prouve un alignement exact des deux senseurs ainsi qu'un ajustement temporel parfait de la prise de données de ceux-ci. Cependant, les distances entres les piétons et les voitures sont imprécises au vu des défauts de précisions de position des piétons. Sans compter le fait que le nombre de voitures et de piétons n'est pas exacte à cause des créations de fausses pistes. Une interaction piétons-voitures bien représentative n'est donc, à ce stade du développement, pas encore possible. C'est pourquoi, des traitements de données ainsi que des optimisations de paramètres vont être réalisés dans le but d'atteindre correctement les objectifs fixés.

Chapitre 5

Améliorations de l'algorithme de suivi

Le chapitre précédent a exposé le développement de l'algorithme de suivi des voitures et des piétons. Néanmoins, comme les résultats ont pu le démontrer, les données retournées ne suffisent pas à une bonne analyse de la route et manquent de précision. Afin d'améliorer l'analyse initiale, il nous a fallu retravailler les algorithmes de suivi et appliquer des traitements aux données reçues par les capteurs. Des solutions ont été trouvées pour remédier par exemple aux fausses pistes enregistrées ou encore à l'imprécision des trajectoires des piétons. De nombreux paramètres ont été optimisés.

Dans cette section, ces améliorations seront mises en avant. Tout d'abord, une étude suivie d'une optimisation des détecteurs caméra et radar sera réalisée à l'aide de diagrammes de ROC (*Receiver Operating Characteristics*). Ensuite, les pré- et post-traitements de données ajoutés à l'implémentation préliminaire seront décrits. Par après, l'optimisation des paramètres influençant la qualité d'analyse seront présentés. Finalement, la structure globale de la fusion des données sera récapitulée. Pour chaque amélioration, les tests de la section précédente seront, à nouveau, effectués dans le but de démontrer leur efficacité.

5.1 Optimisation des détecteurs

Pour optimiser les paramètres influençant la qualité des détecteurs, nous effectuons une étude ROC. Pour ceci, nous avons travaillé sur chaque détecteur avec des valeurs expérimentales sur un nombre restreint de *frames*. Cette hypothèse engendre des résultats simplifiés mais logiques pour permettre l'optimisation.

5.1.1 Utilité des courbes ROC

Etant donné que tout l'algorithme de *tracking* se base sur les détections des senseurs, il est important de connaître la fiabilité de celles-ci. Lorsque des senseurs sont utilisés pour détecter des objets, ceux-ci exigent un compromis pour atteindre la meilleure précision. Ce compromis se situe entre les faux positifs et les vrais positifs. Prenons comme exemple le cas de la caméra qui doit déterminer si oui ou non il y a un piéton sur le *frame* qu'elle analyse. Si celle-ci détecte un piéton à un certain endroit et qu'il y a en réalité un piéton en ce même endroit, on qualifiera cette détection de vrai positif. Tandis que si la caméra détecte un piéton à un endroit alors qu'il n'y a en réalité pas de piéton, c'est un faux positif. L'analyse de ROC permet de représenter ce compromis [19] et par conséquent de l'optimiser. L'analyse se fera à l'aide d'une courbe qui représentera pour chaque seuil de certitude différent (paramètre du détecteur) le nombre de vrais positifs qu'il y a au dépens du nombre de faux positifs. Pour effectuer cette analyse, il faut

donc mettre en lien les résultats obtenus grâce au modèle de détection utilisé et ce pour des paramètres différents avec les résultats obtenus dans la réalité. Le graphe se présentera sous la forme suivante :

1. Axe vertical : taux de vrais positifs [0 ;1]
2. Axe horizontal : taux de faux positifs [0 ;1]
3. Axe oblique (y = x) : en dessous de cet axe, le modèle de détection génère plus de faux positifs que de vrais positifs

Cette analyse ROC va permettre de choisir le paramètre optimal pour la qualité des senseurs. Nous réalisons donc cela pour le détecteur de piétons ainsi que pour le détecteur de voitures. Bien que la démarche soit similaire, l'analyse entre réalité et modèle a été différente pour les deux senseurs.

5.1.2 ROC pour le détecteur de piétons

Pour le cas de la caméra, le paramètre qui va déterminer la qualité du détecteur est le taux de certitude permettant de confirmer si la ou les boites prises sur le *frame* sont des boites contenant des piétons. Ce paramètre va donc se situer entre 0% et 100%. Plus ce paramètre est bas, plus on laisse la chance à une boite d'être considérée comme un piéton. Étant donné que l'algorithme utilisé analyse la vidéo *frame* par *frame*, le but est de vérifier sur chacun de ceux-ci si, en réalité, il y a présence d'un piéton ou non lorsqu'il y a une détection en faisant varier le paramètre de certitude. Nous avons donc réalisé cette étude sur 24 images, prises lors d'un test (figure 5.1), considérées comme complexes (présence de plusieurs piétons, présence d'une camionnette [1], présence de voitures cachant des piétons,...).

FIGURE 5.1 – *Frames* analysés avec la caméra pour l'étude ROC

Nous avons comparé la réelle présence/absence de piétons avec chaque détection que le modèle nous renvoyait sur chaque image (*frame*) (voir tableau annexe A3) avec une certitude de 10, 20, 30, 40, 50, 60, 70, 80 et 90%. Cela nous a donc permis d'obtenir le nombre de vrais positifs et de faux positifs pour chaque image et pour chaque certitude. Le cas d'une certitude faible rend de nombreux faux positifs ainsi que de nombreux vrais positifs tandis qu'une certitude élevée fait diminuer les faux positifs mais également les vrais positifs. Une image analysée pour 10%

1. Lors de plusieurs expériences, nous avons constaté que la caméra détectait souvent des piétons sur des camionnettes (fausses détections), c'est pour cela que nous considérons ce type d'image comme "critique".

(figure 5.2), une pour 50% (figure 5.3) et une pour 90% (figure 5.4) sont présentées ci-dessous pour illustrer cela (le cadre rouge apparaît dès que la caméra détecte un piéton à cet endroit).

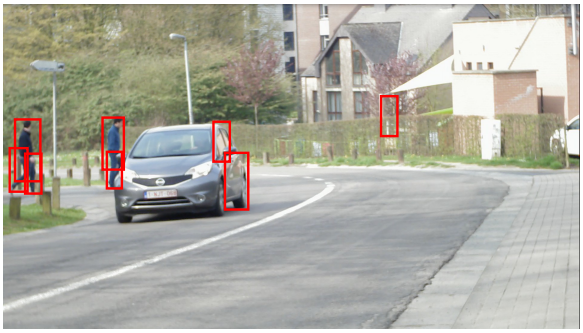


FIGURE 5.2 – Détections de piétons obtenues par la caméra pour un taux de certitude de **10%** (le cadre rouge apparaît dès que la caméra détecte un piéton à cet endroit)

FIGURE 5.3 – Détections de piétons obtenues par la caméra pour un taux de certitude de **50%** (le cadre rouge apparaît dès que la caméra détecte un piéton à cet endroit)

FIGURE 5.4 – Détections de piétons obtenues par la caméra pour un taux de certitude de **90%** (le cadre rouge apparaît dès que la caméra détecte un piéton à cet endroit)

Pour tracer la courbe de ROC, il faut maintenant calculer pour chaque certitude le taux de vrais positifs et le taux de faux positifs. Pour réaliser cela, on utilise la méthode brute [8] de comptage étant donné que l'on a des résultats expérimentaux concrets et que la décision entre fausse détection et vraie détection est simple (une détection d'un piéton présent sur le *frame* = vraie détection et une détection d'un piéton non-présent = fausse détection). On définit donc :

1. VD = Nombre total de vraies détections sur les vingt-quatre images pour un seuil de certitude
2. FD = Nombre total de fausses détections sur les vingt-quatre images pour un seuil de certitude
3. TP = Nombre total de piétons réellement présents sur les vingt-quatre images
4. NPND = Nombre total de non-présences non-détectées sur les vingt-quatre images [2] pour un seuil de certitude

2. Ce terme est assez particulier car il représente le nombre de piétons qu'il n'y a pas sur l'image. Etant donné que dans notre cas il est impossible de choisir cette valeur (cela n'a pas de sens de dire qu'il y a, par exemple, 10 non-présences non détectées sur une image), nous la fixons à 1. L'impact de cette valeur sur le taux de faux positifs sera proportionnel pour chaque valeur du seuil de certitude et n'influencera donc pas l'étude du paramètre.

$$TPR = \frac{VD}{TP}$$

$$FPR = \frac{FD}{FD + NPND}$$

On obtient donc un point du diagramme de ROC [FPR(cert) ;TPR(cert)] pour chaque certitude allant de 10% à 90% (voir table 5.1).

cert [%]	10	20	30	40	50	60	70	80	90
[FPR(cert) ;TPR(cert)]	[0.66 ;1]	[0.56 ;1]	[0.33 ;1]	[0.15 ;1]	[0.04 ;0.95]	[0.04 ;0.95]	[0 ;0.81]	[0 ;0.53]	[0 ;0.29]

TABLE 5.1 – Tableau des coordonnées du diagramme ROC pour le détecteur de piétons

Cela permet de tracer et d'analyser la courbe du diagramme de ROC (figure (5.5)).

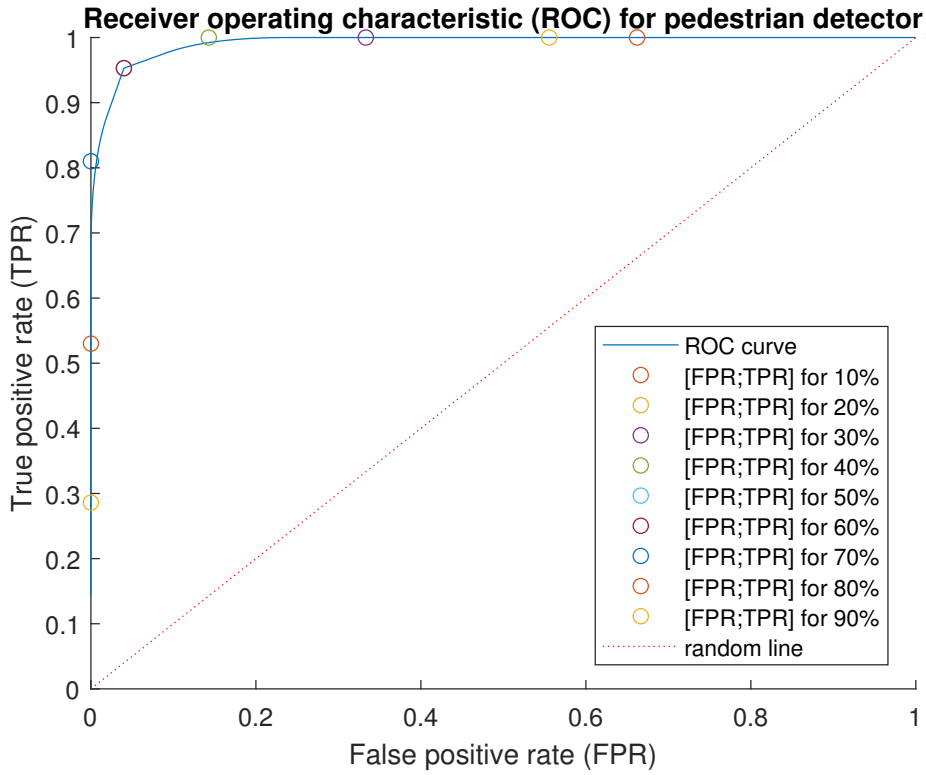


FIGURE 5.5 – Courbe ROC pour le détecteur de piétons

Choix du paramètre de certitude

En analysant la courbe ROC, il apparaît de manière logique que plus le taux de certitude est faible, plus le taux de vrais positifs va tendre vers 1 et le taux de faux positifs également. Pour un taux de certitude élevé, le taux de faux positifs va tendre vers 0 ainsi que le taux de vrais positifs. Un modèle parfait se situe en [0 ;1]. Au vu de la méthode de *tracking* choisie, les faux positifs apparaissent comme de réels problèmes pour deux raisons :

1. Il y a création de fausses pistes qui vont être comptées comme des piétons et qui vont continuer d'être suivies s'il y a des fausses détections qui se répètent proches de la dite fausse piste.
2. Ces fausses détections risquent de contrarier de vraies pistes. S'il y a une fausse détection à proximité d'une vraie piste, celle-ci va suivre le chemin de la fausse détection et donc la trajectoire sera faussée.

Il est donc préférable de se diriger vers un paramètre de certitude donnant un taux de faux positifs se rapprochant de 0 mais avec un taux de vrais positifs suffisamment élevé que pour permettre un suivi complet du piéton (empêcher la perte de la piste). Au vu du graphe, le compromis se fait donc pour le paramètre de certitude à 70% car il permet un taux de faux positifs très proche de 0 et un taux de vrais positifs aux environs de 0.8.

5.1.3 ROC pour le détecteur de voitures

En ce qui concerne le radar, le paramètre à optimiser pour avoir le meilleur compromis entre vraies et fausses détections est le filtrage à l'aide du rapport entre l'intensité du signal reçu et un seuil d'intensité spécifique au radar [3]. En effet, pour filtrer les données reçues, ce rapport peut être diminué ou augmenté. Si cette valeur est faible (aux alentours de 20) cela signifie que beaucoup de données radars sont acceptées pour le tracking de voitures ce qui signifie un risque plus grand de fausses détections. Tandis qu'une valeur plus élévée (environ 50) de ce rapport va restreindre le nombre de détections utiles au tracking. Cela signifie moins de fausses détections mais également un manque de vraies détections.

Expérimentalement, nous avons procédé à nouveau à l'analyse *frame* par *frame* du radar que nous avons pu visualiser à l'aide de la caméra. Nous avons donc pris un test dans lequel il y avait la présence de nombreuses voitures et analysé un *frame* environ toutes les 5 secondes (12 *frames* au total (voir figure 5.6)). Grâce à l'image caméra, nous avons pu comparer si une voiture détectée par le radar à un certain moment était bien présente au moment vidéo correspondant (et inversement pour les fausses détections).

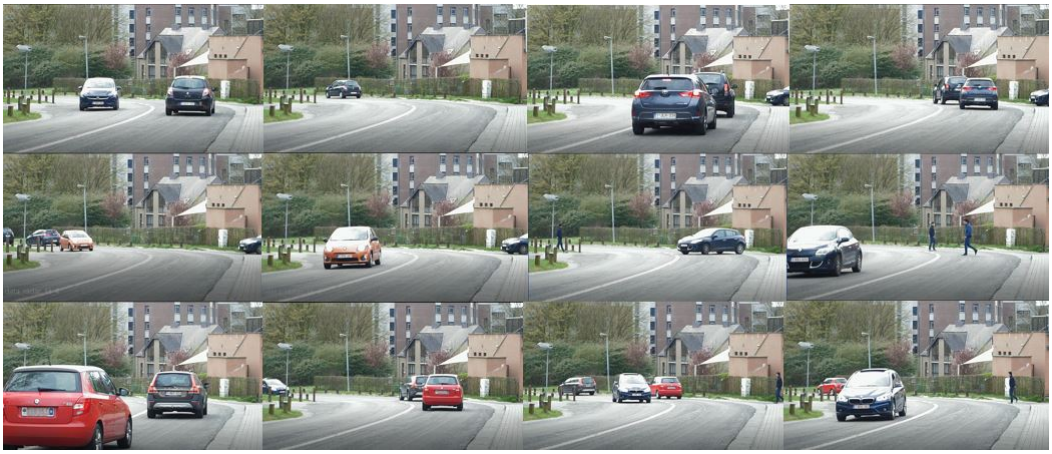


FIGURE 5.6 – Visualisation des *frames* correspondant à l'analyse radar

Nous avons donc comparé la réelle présence/absence de voitures pour un rapport d'intensité de 20, 25, 30, 35, 40, 45, 50 et 55 (voir tableau annexe A3). Des pré-groupements de données ont été réalisés pour assembler les détections, proches entre elles, qui représenteraient la même voiture. L'analyse de faux et vrais positifs d'un *frame* pour trois valeurs d'intensité différentes est illustrée ci-dessous. Tout d'abord, pour un rapport de 20 (figure 5.7) on voit qu'il y a de nombreux faux positifs, à 35 (figure 5.8) le nombre de détections est exact (pas de faux positifs) tandis qu'à 55 (figure 5.9), il manque une vraie détection.

3. Ce rapport permet donc de sélectionner les cibles en fonction de l'amplitude du signal réfléchi.

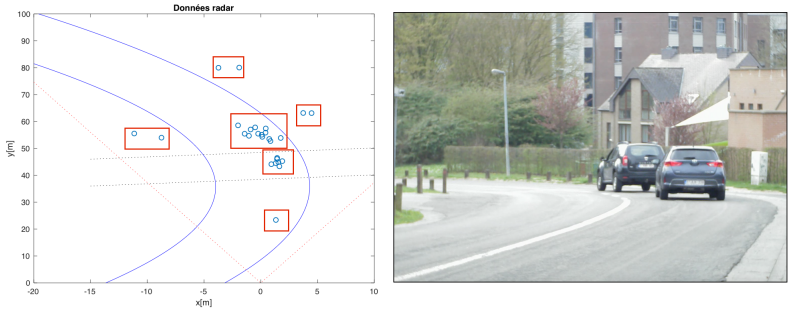


FIGURE 5.7 – Détections de voitures obtenues par le radar pour un rapport d'intensité de **20** (le cadre rouge assemble les données qui correspondraient à une seule voiture)

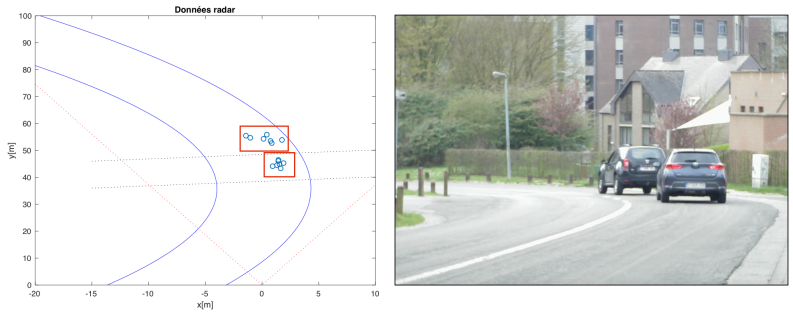


FIGURE 5.8 – Détections de voitures obtenues par le radar pour un rapport d'intensité de **35** (le cadre rouge assemble les données qui correspondraient à une seule voiture)

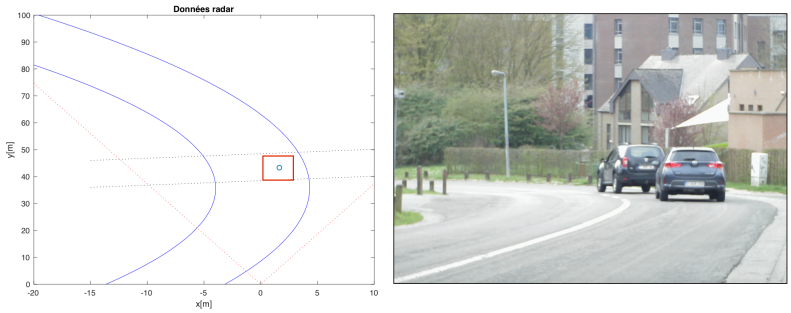


FIGURE 5.9 – Détections de voitures obtenues par le radar pour un rapport d'intensité de **55** (le cadre rouge assemble les données qui correspondraient à une seule voiture)

Ensuite, en réalisant les mêmes calculs pour le taux de vrais positifs et le taux de faux positifs que ceux réalisés pour le détecteur de piétons, on trouve pour chaque valeur du paramètre un point dans le diagramme de ROC [FPR(int) ;TPR(int)] (voir tableau 5.2) :

int	20	25	30	35	40	45	50	55
[FPR(int) ;TPR(int)]	[0.76 ;1]	[0.6 ;0.92]	[0.25 ;0.83]	[0.2 ;0.7]	[0.08 ;0.56]	[0 ;0.52]	[0 ;0.43]	[0 ;0.22]

TABLE 5.2 – Tableau des coordonnées du diagramme ROC pour le détecteur de voitures

La courbe de ROC est alors tracée grâce à ces points pour le détecteur de voitures (figure 5.10).

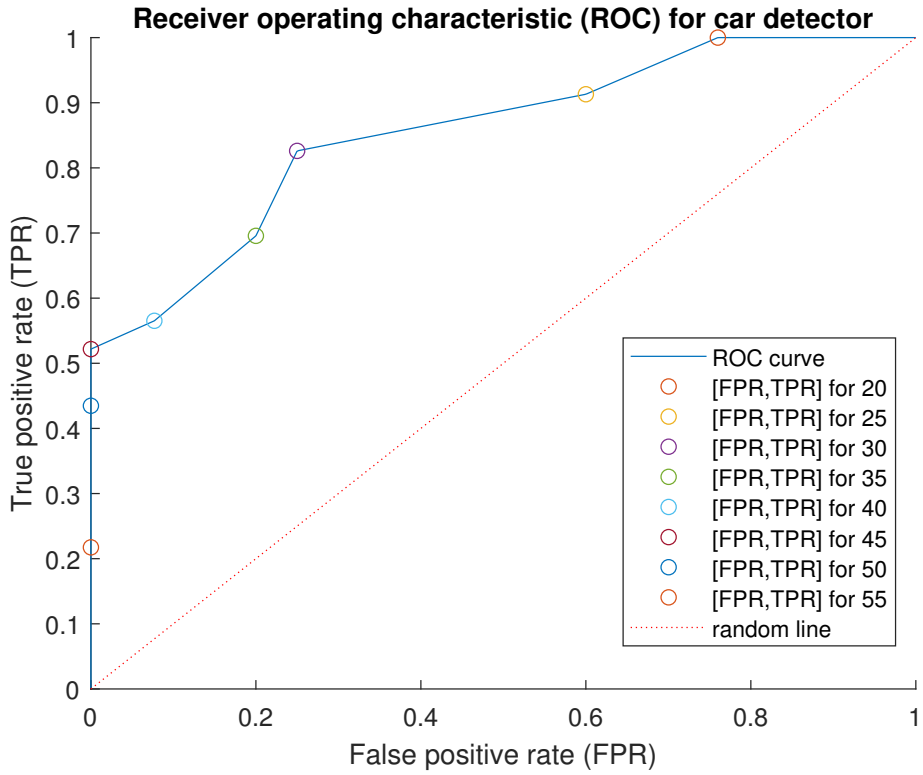


FIGURE 5.10 – Courbe ROC pour le détecteur de voitures

Choix du paramètre de rapport d'intensité

En analysant la courbe ROC, on observe que le taux de faux positifs se rapproche de 0 pour des rapports d'intensité de 40 et plus. Il ne faut cependant pas que ce rapport soit trop élevé car le taux de vrais positifs va alors diminuer fortement. En effet à 40, ce taux de vrais positifs est proche de 0.5. Étant donné que l'algorithme de *tracking* est identique pour les piétons et les voitures, le choix du facteur d'intensité a les mêmes contraintes que le choix du facteur de certitude (il ne faut pas ou presque pas de faux positifs). On choisit donc un seuil de rapport d'intensité égal à 35 c'est à dire un taux de faux positifs proche de 0.2 et un taux de vrais positifs égal à 0.7.

5.2 Traitement de données caméra

A la sortie de l'algorithme de suivi, une base de données regroupant l'ensemble des pistes est obtenue. Celle-ci contient donc les historiques propres à chaque piéton (voir tableau 4.12). Sur base de ces données, un tri va être effectué pour ne garder que des pistes valables. Seuls les pistes de piétons contenant un nombre suffisamment élevé de réelles détections vont être gardées. Cela permet d'écarter les fausses pistes restantes qui modifient le nombre total de piétons. Lors de l'analyse vidéo, chaque piste incrémentera ou non sa variable totalVisibleCount représentant le nombre total de détections (validées par le détecteur) du piéton. Par la suite, les pistes les plus probables (celles avec un totalVisibleCount élevé) seront sélectionnées. Afin de réaliser un comptage précis, ce seuil minimal va être optimisé.

Dans cette section, l'optimisation des paramètres sera décrite. Ensuite, les traitements appliqués sur les données seront expliqués pour finalement présenter les nouveaux résultats obtenus grâce aux améliorations.

5.2.1 Optimisation caméra

Le test sur lequel se base l'optimisation dénombre neuf piétons traversant la route étudiée. Les cas les plus intéressants ont été choisis : voiture qui passe devant un piéton, trois piétons qui se croisent, une traversée oblique d'un piéton en dehors du champ étudié,... [4]
Le paramètre étudié (totalVisibleCount) varie entre 0 (toutes les pistes sont sélectionnées) et 400 (seules les pistes comprenant 400 réelles détections sont sélectionnées). Il est intéressant de coupler l'optimisation de ce paramètre avec un autre paramètre fixé expérimentalement dans la section précédente (section 4.4) qui est le costOfNonAssignement. Pour rappel, ce paramètre décide si une détection va être assignée à une piste ou si cette détection va créer une nouvelle piste (section 4.2.1). Ces deux paramètres influencent le nombre total de piétons. On peut observer à la figure (5.11), la variation du nombre de piétons en fonction des deux paramètres.

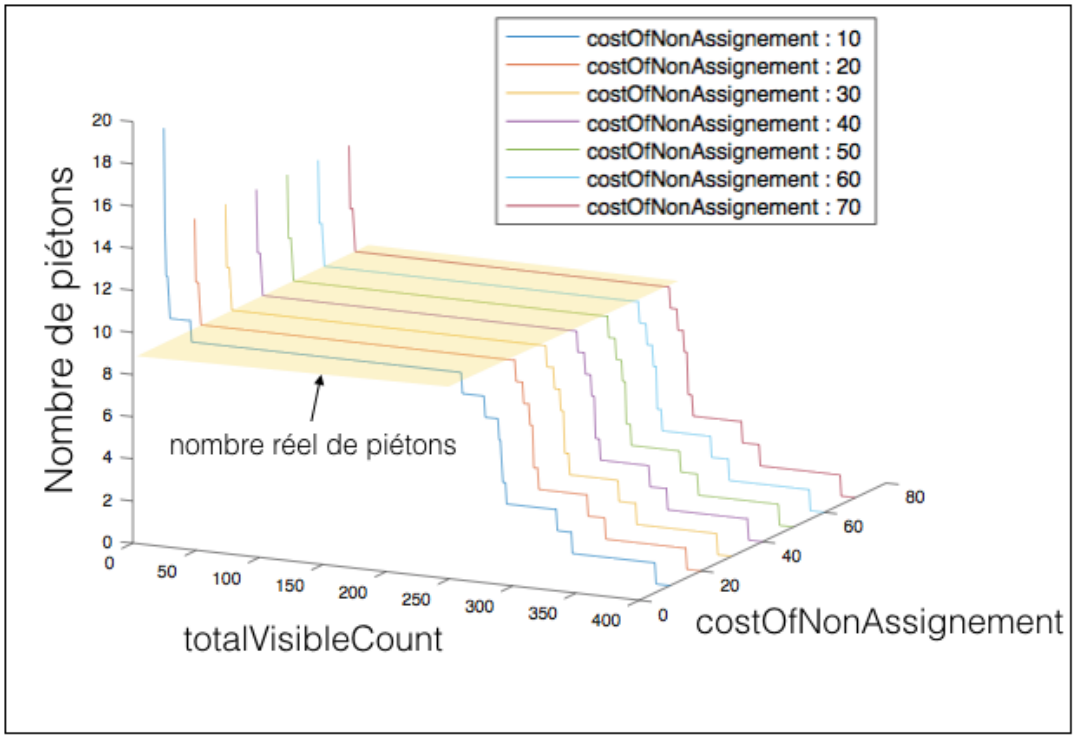


FIGURE 5.11 – Optimisation des paramètres totalVisibleCount et costOfNonAssignement

Le but de l'optimisation est d'obtenir le nombre réel de piétons. Afin de choisir les deux paramètres il faut étudier leur impact. Le totalVisibleCount ne doit pas être trop important pour ne pas éliminer de bonnes pistes [5]. Un trop haut taux de non-assignation (costOfNonAssignement) diminuera la création de nouvelles pistes tandis qu'une valeur trop faible aura l'impact inverse.

On remarque une convergence rapide vers le nombre réel de piétons. A partir d'un costOfNonAssignement de 20 et d'un totalVisibleCount de 7, le bon résultat est obtenu. La réduction du paramètre de coût de non-assignation par rapport à la valeur fixée précédemment (son ancienne valeur est de 40) signifie qu'un plus grand nombre de nouvelles pistes vont être créées. En effet, plus ce nombre est faible, plus l'assignation d'une détection à une piste risque d'être refusée (section 4.2.1) et donc une nouvelle piste est créée. Le résultat du test stipule donc qu'il est plus astucieux de créer des nouvelles pistes ne contenant que quelques détections (fausses pistes) et de les éliminer par la suite avec le post-traitement. Finalement, en prenant une marge de sécurité vis à vis du nombre de détections requises pour qu'une piste soit admise comme réelle, on fixe les

4. La vidéo est disponible dans le drive dédié au mémoire sous le nom de *data_13_3.mp4*.
5. Une bonne piste est une piste contenant réellement un piéton.

paramètres suivants :

$$costOfNonAssignement = 20$$

$$totalVisibleCount = 15$$

5.2.2 Moyenne pondérée et lissage des trajectoires

Nous avons vu précédemment (section 4.1.1) que le détecteur utilisé par la caméra peut mener à des erreurs suite à la variation de la taille de la boite entourant le piéton. Cette erreur est reportée à travers le modèle de transformation d'échelle (*pinhole*). Afin de réduire au maximum ces variations inattendues, une moyenne pondérée en distance est appliquée sur les positions toutes les demi-secondes soit une moyenne sur 15 *frames*. Pour ce faire, on applique un poids w_i à chaque point en fonction de sa distance x par rapport aux autres :

$$w_i = \sum_j \frac{1}{x_i - x_j}$$

$$x_{new} = \frac{\sum(w_i \cdot x_i)}{\sum w_i}$$

Finalement, un lissage de la trajectoire par le biais d'une interpolation polynomiale au sens des moindres carrés du troisième degré [24][29] est appliqué pour un tracé plus agréable au visuel lors de l'affichage dynamique (section 6). Seules les positions en y sont remplacées. En effet, les positions horizontales ne sont pas affectées par la hauteur variable de la *bounding box*.

5.2.3 Résultats après traitement

Suite à ces améliorations apportées à l'algorithme, nous avons réalisé les mêmes tests effectués dans la section (4.4.1). Par conséquent, après avoir optimisé les paramètres et réalisé l'étape de post-traitement en moyenne pondérée, on peut voir les améliorations obtenues (figures (5.12),(5.13),(5.14) et (5.15)). Un lissage de trajectoire est également présenté sur le lieu d'étude [6] (figure 5.16).

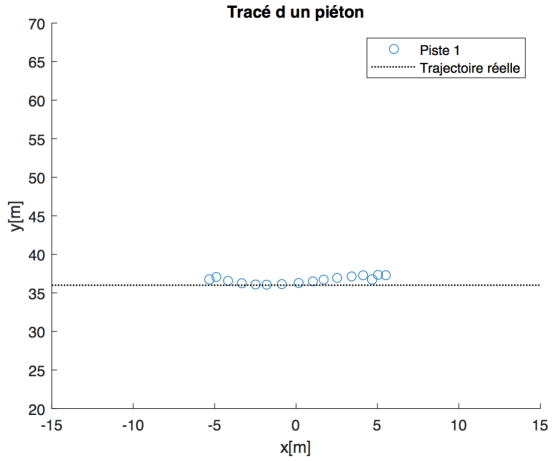


FIGURE 5.12 – Test 1 : Traversée d'un piéton (piste 1)

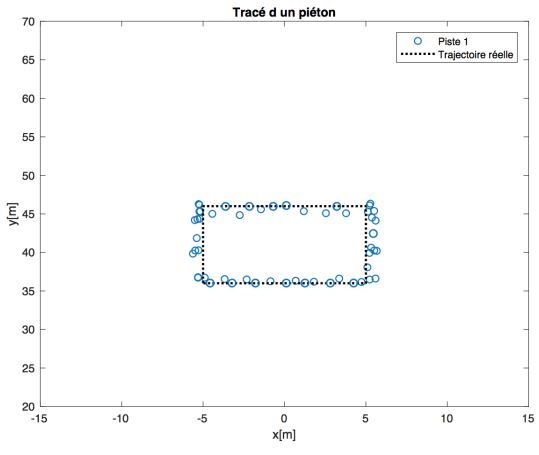


FIGURE 5.13 – Test 2 : Carré effectué par un piéton (piste 1)

6. Les positions plus éparpillées situées sur la gauche sont dues à la présence de petits plots en bois qui cachent les jambes des piétons faussant ainsi la taille de la *bounding box*.

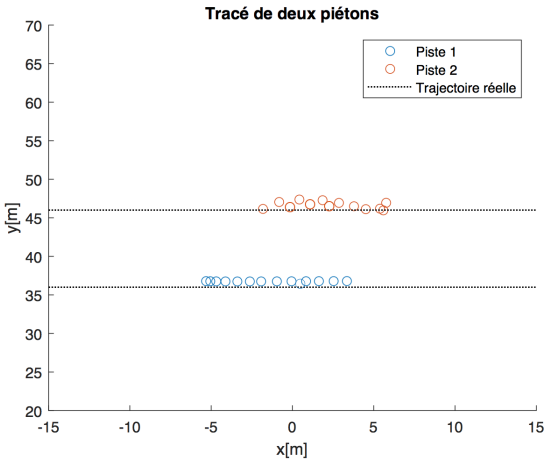


FIGURE 5.14 – Test 3 : Croisement de deux piétons (piste 1 et 2)

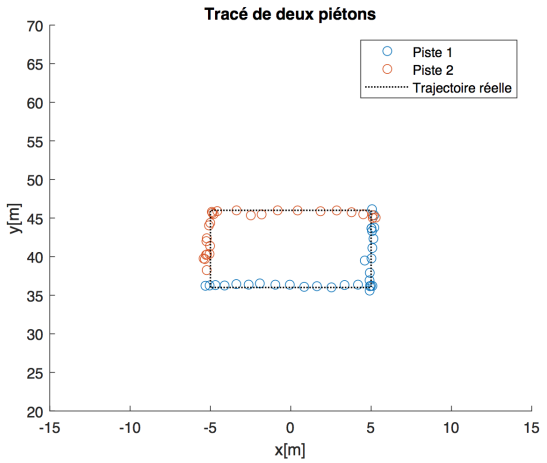


FIGURE 5.15 – Test 4 : tracé en **L** effectué par 2 piétons (piste 1 et 2)

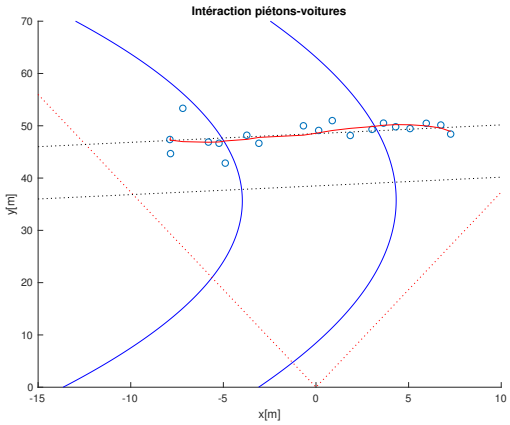


FIGURE 5.16 – Lissage de la trajectoire d'un piéton traversant la route étudiée

Discussion des résultats

On remarque une amélioration considérable de précision par rapport aux tests précédents (section 4.4.1). Pour une simple traversée on observait une erreur moyenne entre le tracé réel et les positions calculées de 1.1863 mètres en y qui a été réduite à 0.4319 mètre avec le post-traitement appliqué. De plus, suite à l'application de la moyenne pondérée, les *outliers*[7] en position ne sont plus présents dans les graphes. Cette amélioration est cruciale pour l'analyse de l'interaction entre les voitures et les piétons. En effet, dans le chapitre (6) nous verrons que la distance minimale entre un piéton et une voiture est calculée, les *outliers* pourraient alors fausser totalement cette information. On remarque également la disparition des faux positifs obtenus précédemment (section 4.4.1) montrant l'utilité du paramètre seuil du totalVisibleCount.

7. Ce sont les positions fort écartées de la trajectoire du piéton. Elles sont des positions mal calculées à cause de la variation de la taille de la boite.

5.3 Traitement de données radar

Dans cette section, les différents traitements appliqués aux données radars seront présentés. Tout d'abord, le pré-traitement effectué sur les données brutes suivi du post-traitement appliqué sur les données obtenues à la sortie de l'algorithme de suivi seront expliqués. Pour finir, une optimisation des paramètres sera réalisée.

5.3.1 Pré-traitement

L'algorithme de suivi a été conçu afin de réaliser la correspondance entre des détections et des pistes. Néanmoins, seule une détection peut être assignée par piste à chaque *frame* (section 4.2.1). Le radar peut quant à lui, renvoyer pour un seul *frame* plusieurs détections correspondant à la même voiture. Ceci pause problème car de nouvelles pistes vont être créées pour la même voiture et fausser le comptage de celles-ci.

Afin de régler ce nombre trop important de détections au même endroit, un pré-traitement des données est réalisé. Celui-ci a pour but de grouper les détections se trouvant sous une distance seuil. Ce traitement se passe en trois étapes (figure 5.17) :

1. La distance euclidienne entre chaque détection sur un *frame* est calculée
2. Les données dont la distance respective est faible sont groupées (distance située sous un seuil fixé)
3. La moyenne des positions et du temps est calculée pour chaque groupe

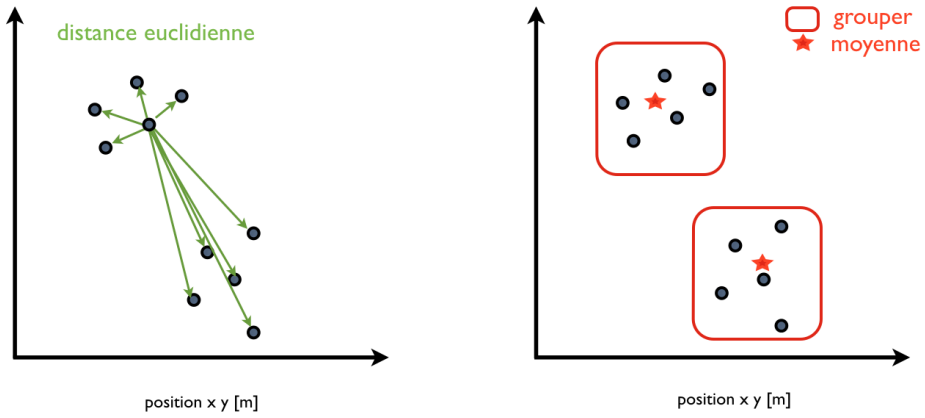


FIGURE 5.17 – Groupement des données

Ce pré-traitement va permettre de faire correspondre les détections multiples (exemple : l'avant et l'arrière de la voiture) à une seule cible.

Une fois ce groupement réalisé, les données passent à travers un filtre de position. Celui-ci représente la limitation de la route étudiée. Les données situées hors du champ d'étude (faux positifs) sont alors abandonnées. Un filtre de vitesse est également appliqué. Une limite de 1 km/h est imposée afin d'éliminer les objets statiques. L'application du filtre sur la géométrie de la route réduit le traitement des données à notre cas d'étude. Il n'est donc pas directement adaptable à d'autres routes. Néanmoins, une amélioration pour remplacer cette méthode est proposée section (7.1). On peut observer le résultat de ce pré-traitement figure (5.18).

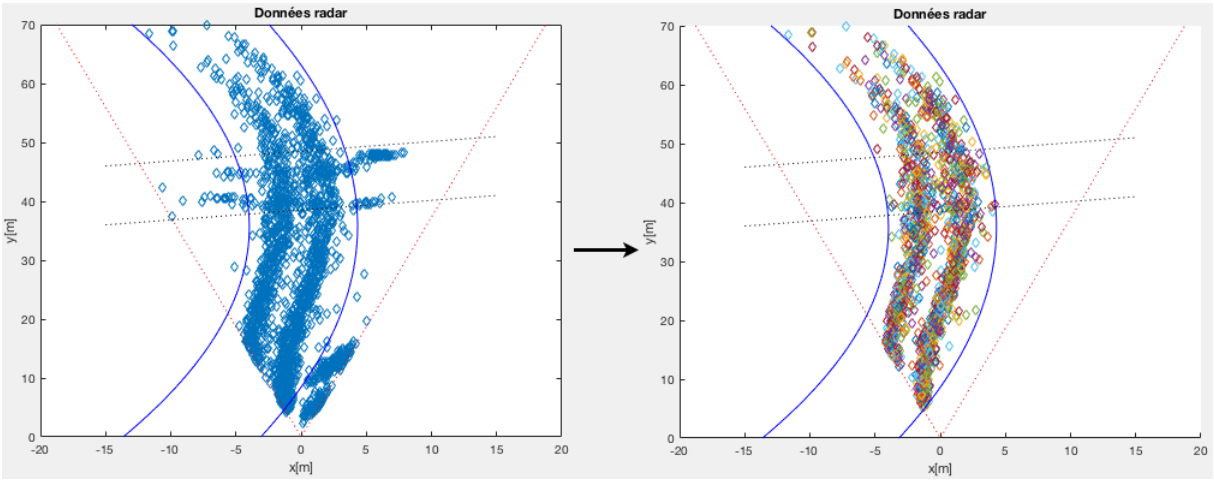


FIGURE 5.18 – Pré-traitement des données radar

5.3.2 Post-traitement

Les données filtrées passent finalement à travers l'algorithme de suivi. A nouveau, quelques faux positifs sont toujours présents à la sortie. La même méthode utilisée pour la caméra est appliquée afin de sélectionner les pistes valides. Un seuil minimal de détection (totalVisibleCount) va être utilisé pour choisir si une piste a eu suffisamment d'assignation de détections pour confirmer qu'elle représente bien une voiture. Ce seuil doit être optimisé avec d'autres paramètres, c'est ce qui est réalisé à la section suivante.

5.3.3 Optimisation des paramètres

Pour les traitements de données proposés, deux paramètres sont à optimiser :

— distanceLimite : la distance seuil pour la formation de groupe

— totalVisibleCount : le seuil minimal de détections pour être considéré comme une vraie piste

A nouveau, il est intéressant d'inclure le paramètre de suivi costOfNonAssignement (section 4.4.2) dans l'optimisation. En effet, cette valeur a également un impact sur le nombre total de voitures et la formation de nouvelles pistes.

Un test est donc réalisé pour déterminer les meilleurs paramètres. Le test sur lequel se base l'optimisation dénombre treize voitures. Les cas les plus intéressants ont, à nouveau, été traités : des voitures qui se suivent , des voitures qui se croisent, des voitures en provenance de la route transversale généralement empruntée par les piétons,... [8]. On peut observer les résultats de la variation des trois paramètres à la figure (5.19). Le but étant, à nouveau, d'obtenir le nombre réel de voitures.

8. La vidéo est disponible dans le drive dédié au mémoire sous le nom de *data_13_4.mp4*

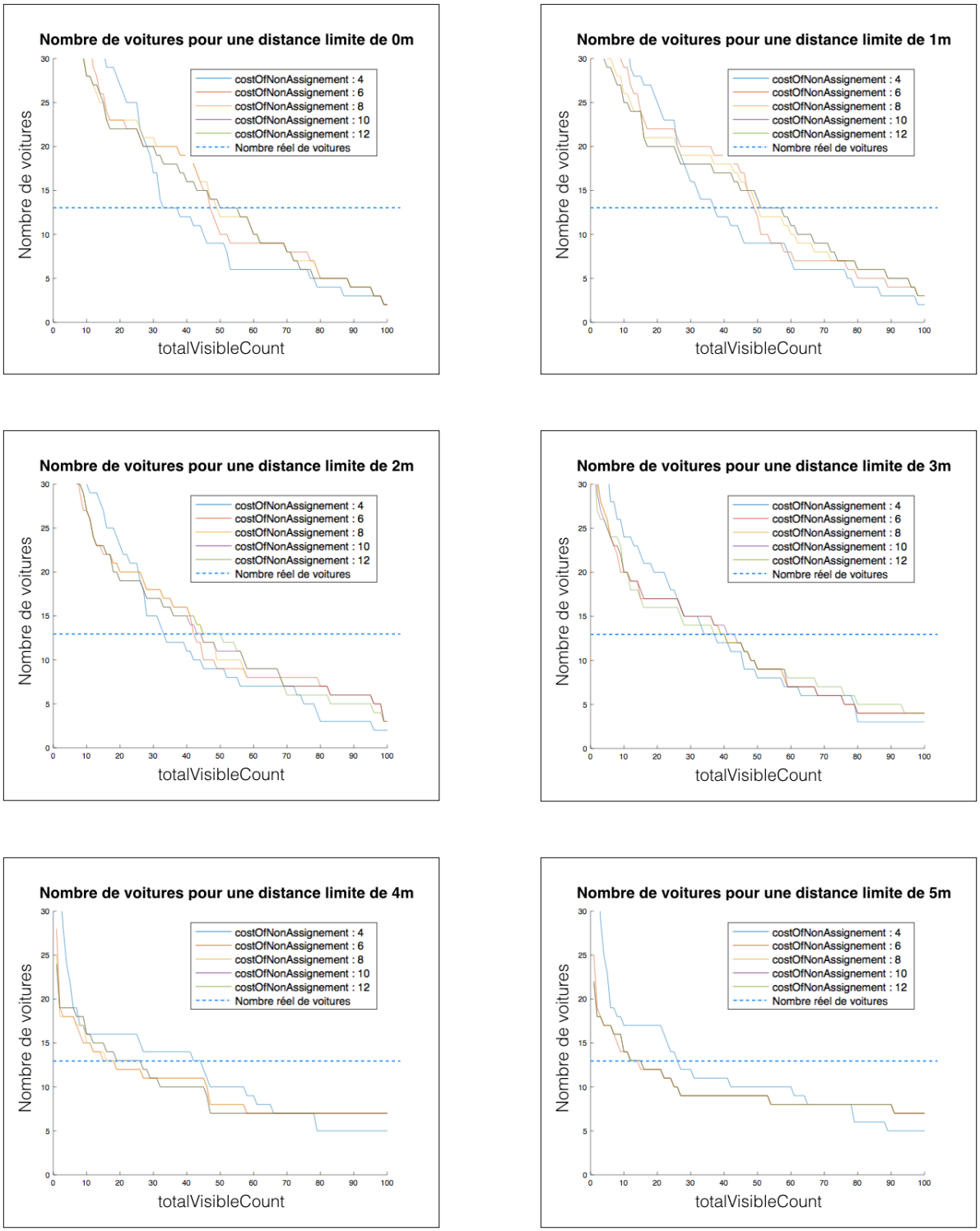


FIGURE 5.19 – Nombre total de voitures en fonction des trois paramètres : costOfNonAssignement, totalVisibleCount (abscisse) et la distance limite (une distance fixée par graphe)

On remarque un certain nombre de candidats possibles (paliers situés sur la ligne en pointillé). Afin de réaliser le meilleur choix, il faut se baser sur la logique. Tout d'abord, appliquer une trop grande distance limite réduit la précision du suivi. En effet, la moyenne des positions dans un groupe de points sera effectuée sur un trop grand nombre de détections (toutes celles situées sous la distance limite). De plus, lors de cas plus complexes à plusieurs voitures (suivi et croisement), les détections pour les voitures risquent d'être groupées et donc ne représenteront qu'une seule détection de voiture. Par ailleurs, appliquer un trop haut taux de non-assignation diminuera la

création de nouvelles pistes. En effet, l'assignation d'une détection trop éloignée de la piste que pour lui être attribuée, ne sera pas refusée. Finalement, le seuil du nombre total de détections nécessaires (totalVisibleCount) ne doit pas être trop important auquel cas de vrais positifs pourraient ne pas être pris en compte.

Après cette analyse, une distance limite de 3.5 mètres a été choisie. Celle-ci représente la longueur moyenne d'une voiture et donc une distance sous laquelle les détections peuvent appartenir à la même cible. Le palier le plus long (figure 5.20) (celui avec le plus de marge) correspond à un coût de non assignation de 5 et un nombre total de détections de 35.

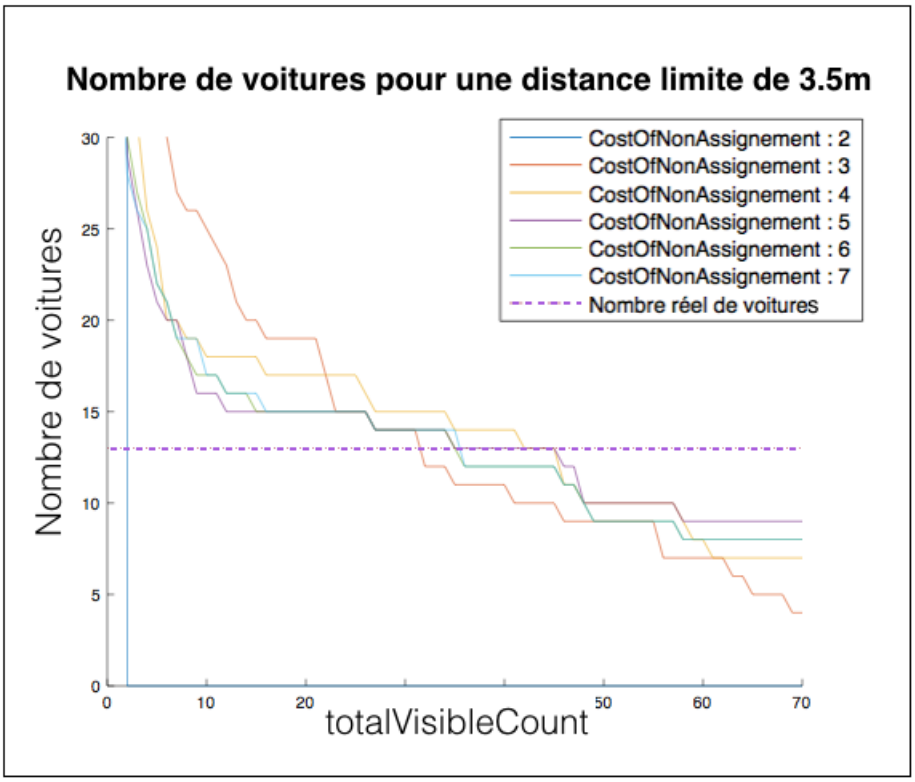


FIGURE 5.20 – Nombre de voitures en fonction du costOfNonAssignement et du totalVisibleCount (abscisse) pour une distance limite de 3.5 mètres

Ces paramètres obtenus respectent le développement logique ci-dessus, ils sont donc fixés à :

$$costOfNonAssignement = 5$$

$$distanceLimite = 3.5[m]$$

$$totalVisibleCount = 35$$

5.3.4 Résultats après traitement

Après avoir filtré les données et avoir optimisé l'algorithme de suivi pour les voitures, les mêmes tests présentés à la section (4.4.2) ont été effectués. Les résultats sont représentés à la figure (5.21),(5.22) et (5.23).

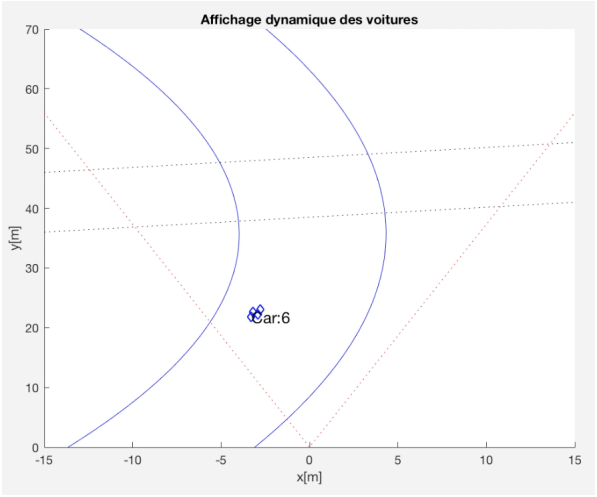


FIGURE 5.21 – Test 1 : Aller simple d'une voiture

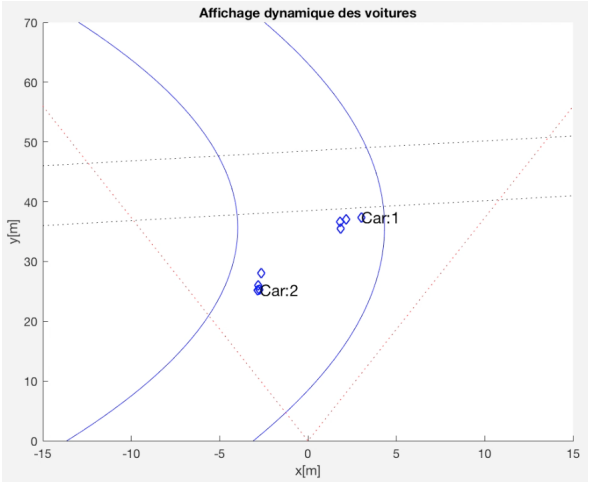


FIGURE 5.22 – Test 2 : Croisement de deux voitures

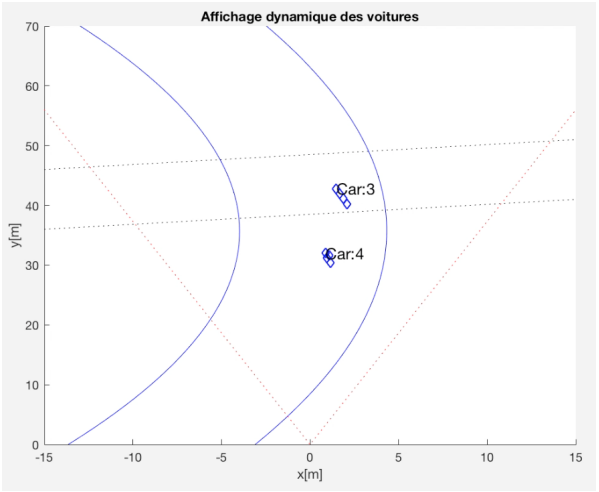


FIGURE 5.23 – Test 3 : Suivi de deux voitures

Discussion des résultats

Il apparaît que suite à ce filtrage des données, le problème concernant la création de pistes multiples pour une seule voiture est réglé. Le groupement de données permet bien de gérer le cas de détections multiples correspondantes à une seul voiture par *frame* et rendant ainsi le comptage des voitures correct. L'optimisation couplée des paramètres costOfNonAssignement et distanceLimite permet également de ne plus avoir d'échange de filtres [9] entre deux voitures qui se croisent. Finalement, les fausses pistes restantes sont bien éliminées lors du post-traitement.

9. Pour rappel, lorsque deux cibles se croisent, il est possible que les deux pistes qui permettent le suivi s'échangent de cibles, ceci est dû au fait que les deux détections sont proches l'une de l'autre et sont donc chacune assignées à la mauvaise piste. Dès lors, la trajectoire des cibles est faussée. C'est ce que nous appelons un échange de filtres.

5.4 Fusion des traitements de données

Suite aux développements précédents, nous pouvons constater une nette amélioration de suivi des voitures et piétons séparément. Il convient à présent de réaliser un test final sur le couplage des données. L'ensemble des paramètres ainsi que l'évolution du traitement de données concernant l'algorithme se retrouvent dans un récapitulatif présenté en début de section.

5.4.1 Structure du traitement des données

La structure initiale présentée section (4.3) a été modifiée pour prendre en compte les pré- et post-traitements de données. Pour une meilleure visualisation, la structure finale de fusion est représentée à la figure (5.24).

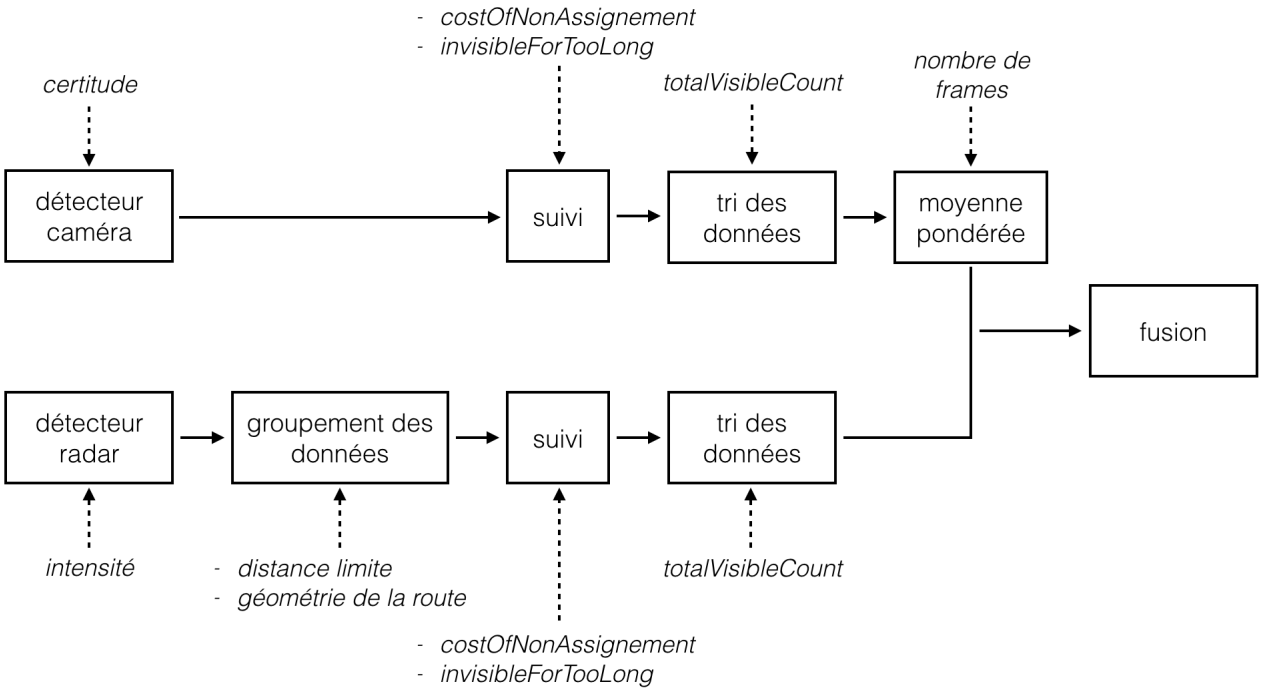


FIGURE 5.24 – Structure du traitement des données et paramètres influents

Dans les tableaux (5.3) et (5.4), se trouvent les valeurs choisies pour les différents paramètres :

	Radar	**Paramètres**	**Valeurs**
Détecteur	Suppression des faux positifs	intensité	35
Pré-traitement	Forme de la route Formation de groupe	/ distance limite	3.5 [m]
Suivi	Nombre consécutif sans détection Taux d'assignation minimal	invisibleForTooLong costOfNonAssignement	15 *frames* 5
Post traitement	Tri des données	totalVisibleCount	35 *frames*

TABLE 5.3 – Traitement de données radar

	camera	paramètres	Valeurs
Détecteur	Suppression des faux positifs	certitude	70%
Pré traitement	/	/	/
Suivi	Nombre consécutif sans détection Taux d'assignation minimal	invisibleForTooLong costOfNonAssignement	60 *frames* 20
Post traitement	Tri des données Moyenne pondérée	totalVisibleCount nombre de frames	15 *frames* 15 *frames*

TABLE 5.4 – Traitement de données caméra

Récapitulatif

La caméra nous permet de capturer en temps réel les images correspondantes aux données radar lancé au même instant. C'est cette vidéo qui est l'entrée de l'algorithme de suivi de piétons. Après l'analyse des détections de piétons et l'utilisation des données temps du radar, une liste de l'ensemble des positions pour chaque piéton est obtenue. Un tri est alors utilisé pour séparer les vrais positifs des faux et une moyenne pondérée en position est appliquée pour chaque piste. Finalement, les positions définitives pour chaque piéton avec leur temps associé sont obtenues et prêtes à être analysées.

Le radar quant à lui enregistre en temps réel l'ensemble des détections reçues en associant le temps correspondant à chaque *frame* analysé. Après le stockage de ces données, un filtrage est appliqué pour ne garder que les détections d'objets en mouvement et présents dans le champ d'étude. Ensuite, les détections multiples par *frame* vont être groupées pour ne reprendre plus qu'une position par cible. On obtient alors une liste de positions classées chronologiquement qui représente l'entrée de l'algorithme de suivi du radar. Après l'analyse des données, un set de positions pour chaque voiture avec leur temps correspondant est retourné. Finalement, un dernier tri est effectué pour éliminer les fausses pistes restantes.

Pour terminer, les listes des positions voitures et piétons sont fusionnées et triées par temps. Un graphe dynamique est alors réalisé et permet l'observation de l'évolution temporel des déplacements des piétons et des voitures.

5.4.2 Résultats

L'ensemble des traitements supplémentaires et l'optimisation des paramètres ont permis une amélioration considérable du suivi de voitures et piétons. Le test global de fusion réalisé à la section (4.4.3) a été analysé à nouveau. La vidéo complète de ce test est disponible sur le drive lié à ce mémoire [14] sous le nom de *"13_4_slides.mp4"*. Ci-dessous, la même illustration du test réalisé précédemment (section 4.3) avec ses améliorations est représentée (figure 5.25).

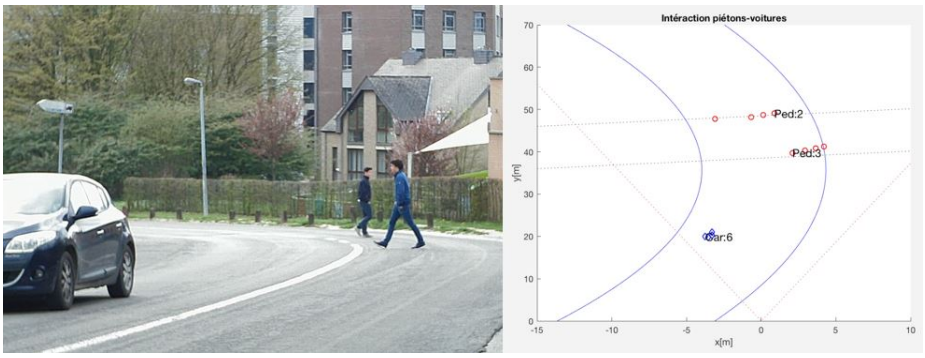


FIGURE 5.25 – Test : fusion des données sur le carrefour étudié

Dans l'annexe A5, un découpage du test complet pour rendre compte de l'aspect dynamique du graphe est présenté (il est toutefois recommandé de regarder la vidéo).

5.4.3 Discussion des résultats

On peut observer à travers le test que l'ensemble des améliorations en radar et caméra se retrouvent également dans le couplage des données. On remarque bien l'adéquation des temps d'apparition des voitures et des piétons : la fusion temporelle des données est respectée. Les interactions de trajectoires entre voitures et piétons représentent également bien la réalité : le dispositif d'alignement des deux capteurs pour observer le carrefour est fonctionnel.

Finalement, on peut conclure que la qualité du suivi des voitures et piétons est bonne. Nous n'observons plus de fausses pistes ou bien d'échanges de filtres entre deux piétons (ou voitures) qui se croisent. Le comptage des cibles est maintenant correcte. De plus, les positions des piétons et des voitures sont précises. Une analyse représentative de leur interaction est à présent possible. Le dispositif est prêt à être confronté à la réalité.

Chapitre 6

Analyse du cas d'étude

Une fois que l'algorithme de *tracking* a été implémenté, amélioré et testé, il nous a semblé important de le confronter à la réalité. Cette section est donc consacrée à la route choisie, analysée à l'aide du système. Pour ce faire, la prise de données s'est faite pendant 30 minutes à 18h qui est une heure de pointe du lieu étudié (les vidéos de cette analyse sont disponibles sur le drive lié à ce mémoire sous le nom de *"30min.mp4"* et *"30min2X.mp4"* [14]).

Avant toute analyse, le cas d'étude peut être rappelé (section 2.4). Celui-ci est un carrefour situé dans un tournant (figure 6.1) où la traversée de piétons est fréquente au vu des bâtiments situés de part et d'autre de la route principale. Étant donné qu'il n'y a pas de panneaux de signalisation ou de passages pour piétons, les voitures arrivent assez rapidement. Cette zone répond aux critères pour être considérée comme dangereuse, notre objectif est de l'affirmer objectivement. Notre système sera placé à environ 40 mètres du croisement (figure 6.2).

FIGURE 6.1 – Cas d'étude vu du haut

FIGURE 6.2 – Cas d'étude vu par le dispositif

6.1 Fréquentation du carrefour

Dans un premier temps et pour avoir plus d'informations concernant le carrefour étudié, l'algorithme, après avoir réalisé l'affichage dynamique (drive), renvoie deux structures utiles pour connaître la fréquentation du carrefour. On obtient donc :

1. Un tableau du nombre de piétons, apparus lors de la prise de données, qui contient pour chaque piéton :
   — son id (le chiffre représentant son ordre d'apparition)
   — son sens de traversée (de la gauche vers la droite, de la droite vers la gauche) ou si il n' a pas traversé, le côté de la route sur lequel il se trouve (gauche, droite)

— le temps auquel il est apparu

2. Un tableau du nombre de voitures, apparues lors de la prise de données, qui contient pour chaque voiture :
   — son id (le chiffre représentant son ordre d'apparition)
   — son sens (si elle est venue vers le dispositif "-" (montée de la route), si elle s'est éloignée du dispositif "+" (descente de la route))
   — sa vitesse moyenne au cours de son apparition
   — sa tendance au niveau vitesse dans le tournant (accélération ou décélération)
   — son temps auquel elle est apparue

Grâce à ces données obtenues (voir tableau (8.3) et (8.4) dans l'annexe A4), plusieurs informations intéressantes peuvent être calculées. Parmi celles-ci, il y a la vitesse moyenne de toutes les voitures, le nombre total de piétons qui ont traversé, le nombre total de voitures et leur tendance générale à accélérer ou décélérer. Pour 30 minutes de test, on dénombre donc :

— Un total de 202 voitures qui roulent en moyenne à du 35 km/h. On observe, cependant, plusieurs voitures roulant à environ 50 km/h. De plus, une majorité des voitures accélèrent dans le tournant (115 contre 87 qui décélèrent).
— Un total de 31 piétons dont 17 traversées (ce sont les traversées qui nous intéressent réellement pour analyser le danger).

Avec ces informations, on peut maintenant se faire une idée quant à la fréquentation de ce carrefour. En effet, 17 traversées pour 30 minutes représentent environ une traversée toutes les 2 minutes et ce avec plus de 5 voitures par minute. Ce nombre de traversées est conséquent sachant qu'aucun passage pour piétons n'existe à cet endroit. De plus, les voitures peuvent atteindre 50 km/h. Tout ceci permet donc de se faire un premier avis sur le danger de ce lieu. Une analyse de l'interaction entre les piétons et les voitures est effectuée à la section suivante (6.2). Celle-ci permettra de réaliser une analyse de danger plus poussée.

Avant de passer à cette section, il est important de vérifier l'exactitude de ces données. Bien que l'algorithme ait prouvé son bon fonctionnement lors des tests, la mise en oeuvre en toute situation pendant un long moment de ce dernier peut faire apparaître de nouvelles erreurs. A l'aide de la vidéo réalisée nous avons pu vérifier le nombre exact de piétons qui ont traversé, le nombre total de piétons ainsi que le nombre de voitures (tableau 6.1).

	Réalité	Algorithme
Nombre de voitures	192	202
Nombre de piétons traversant	18	17
Nombre de piétons sur le côté de la route	9	14
Nombre total de piétons	27	31

TABLE 6.1 – Tableau de comparaison entre la réalité et les informations renvoyées par l'algorithme de *tracking*

En ce qui concerne le nombre total de voitures, on observe que 10 voitures supplémentaires ont été enregistrées, cela représente 5% d'erreur. Ces erreurs sont principalement dues à la présence de bus ou de camions. En effet, le groupement de données (expliqué section 5.3.1) se fait sur des points dont la distance limite est de 3.5 mètres, les bus et camions peuvent donc être comptés comme 2 ou 3 voitures.

Pour le comptage des piétons qui traversent la route, on voit que l'algorithme a manqué un seul piéton, ceci est dû à la traversée de deux piétons l'un à côté de l'autre (ayant la même taille,

le passant cache son compagnon). Ce comptage est très important étant donné que ce sont ces cibles-là qui entrent vraiment dans l'analyse du cas d'étude. C'est le nombre de traversées qui permet de rendre compte de la fréquentation du carrefour.

On voit, cependant, que le nombre total de piétons est moins précis. Ce mauvais comptage vient des piétons sur le bord de la route (piétons qui ne traversent pas). En effet, certains piétons sont comptés à plusieurs reprises car, marchant sur le trottoir, ils partent loin du carrefour étudié mais sont toujours présents dans le champ de vision de la caméra. Dès lors, il arrive que la caméra les détecte, les perd pendant un certain temps puis les détecte à nouveau. De plus, des cyclistes ont été filmés pendant l'analyse, ceux-ci sont pris en compte par le détecteur de piétons. Ces erreurs ne sont pas importantes au vu de l'objectif fixé pour une analyse de danger qui est l'interaction entre les piétons qui traversent et les voitures.

Pour conclure, on voit que pour l'objectif de "mesure" de fréquentations, on peut se fier à notre algorithme développé dans ce travail. En effet, on dénombre 5% d'erreur quant à l'afflux de voitures sur 30 minutes de temps et un nombre presque exact de piétons qui traversent.

6.2 Analyse de danger

Pour permettre d'approfondir l'analyse du carrefour, nous avons décidé de nous baser également sur l'interaction entre les piétons qui traversent et les voitures. On la définit comme étant la distance minimale entre un piéton, sur le point de traverser ou entrain de traverser, et une voiture. A ce moment, il est aussi intéressant d'avoir la vitesse de la voiture. L'algorithme implémenté va donc renvoyer pour chaque piéton détecté en présence de voitures, sa distance minimale entre chaque voiture présente, le temps auquel cette distance est minimale ainsi que la vitesse de chaque voiture à ce moment (tout ceci avec, bien sûr, l'id de chaque piéton et de chaque voiture) (voir tableau annexe A4). On représente ci-dessous un cas d'interaction observé pendant le test de 30 minutes (figure (6.3) et table (6.2)).

FIGURE 6.3 – Interaction observée lors du test de 30 minutes

Piéton	Voiture	Temps [h :min :s]	Distance minimale [m]	Vitesse à l'interaction [km/h]
17	91	18 :19 :58	3.2	40

TABLE 6.2 – Informations reçues dans le tableau renvoyé par l'algorithme pour l'interaction de la figure (6.3). Uniquement les informations sur l'interaction intéressante sont affichées ici.

Il apparaît donc, au vu de ces résultats, que les voitures passent parfois relativement vite

(40 km/h) et sont proches des piétons traversant la route (3 mètres). Ce qui confirme l'idée que celles-ci ne s'attendent pas à rencontrer autant de piétons lors de leur passage dans ce tournant. Ne ralentissant pas, elles constituent un danger pour les piétons.

Finalement, au vu de la fréquentation (section 6.1), on peut supposer que ces interactions piétons-voitures sont fréquentes. Par conséquent, cette zone peut bel et bien être qualifiée de dangereuse et des solutions doivent être trouvées (passage pour piétons, panneaux d'avertissement,...). Des idées d'application en lien avec le dispositif sont proposées à la section (7.3).

Chapitre 7

Améliorations proposées

Maintenant que le procédé implémenté dans ce travail a été présenté et que l'analyse du cas d'étude a prouvé que le dispositif peut accomplir les objectifs fixés. Il est intéressant de regarder vers des améliorations possibles du système. En effet, des défauts persistent quant à la précision de la position des piétons ou encore quant à l'adaptabilité du dispositif pour tout type d'endroit. De plus, d'autres objectifs pour une reprise future du travail peuvent être envisagés.

Dans cette section, deux types d'amélioration seront proposés et développés. Ceux-ci consistent en l'entraînement du radar avec la caméra pour la détection de voitures (cela permettrait de ne plus réaliser de cartographie de la route étudiée) et en l'utilisation de deux caméras en stéréo pour améliorer la précision de la position des piétons détectés. Ces deux améliorations n'ont pas été implémentées dans l'algorithme de *tracking* car il nécessiterait des modifications conséquentes supplémentaires. Cependant, ces recherches sont suffisamment développées, ici, que pour justifier leur choix et les implémenter dans un futur projet.

Finalement, nous développerons des idées d'applications ainsi que des recommandations d'amélioration nécessaires à celles-ci.

7.1 Entraînement du radar à l'aide de la caméra

La caméra et le radar sont deux capteurs utilisés séparément afin de réaliser du *tracking* de piétons et voitures. Leurs données sont, par la suite, fusionnées dans le but d'en faire ressortir une analyse routière. Nous pouvons nous demander s'il est envisageable d'améliorer les détections ainsi que la robustesse de notre algorithme en tirant parti des points positifs d'un capteur par rapport à un autre. En effet, lors du pré-traitement des données en provenance du radar, un tri des données en utilisant une cartographie de la route que l'on analyse (les détections hors du champ d'étude ne sont pas prises en compte) est appliqué. Ceci réduit l'adaptabilité du dispositif, il sera nécessaire de rentrer au préalable la géométrie du nouvel endroit étudié. Dans le but d'augmenter la fiabilité de l'algorithme, il est possible de prendre en compte les images reçues par la caméra et les ajouter dans l'algorithme du radar. Une implémentation permettant cela a été réalisée et est présentée ci-dessous. Cependant, cet algorithme n'a pas été ajouté au code principal car de nombreuses modifications supplémentaires auraient du être effectuées.

7.1.1 Principe de fonctionnement

L'ensemble des détections radar-caméra étant trié chronologiquement, il est possible d'assigner les *frames* de la caméra aux détections radar. La caméra travaillant à du $30fps$ et le radar à du $15fps$, à chaque détection radar seront assignés deux *frames* de caméra. Les images de la caméra permettront alors de vérifier si une ou plusieurs voitures sont présentes et si les détections radar sont vraies ou si elles sont des faux positifs. De manière générale, l'analyse d'images prenant

un temps non négligeable, la vidéo sera sous-échantillonnée. Ce mécanisme permet de réduire considérablement le nombre de pixels de l'image tout en gardant un résultat correct. Dans le cadre de notre analyse, une décimation brute des pixels (garder 1 pixel sur 'n' en largeur et hauteur d'image) suffira à obtenir une image traitable. Une analyse *blob* [31] [32] est ensuite effectuée afin de détecter les voitures.

7.1.2 Analyse blob

Le principe de l'analyse *blob* est de rassembler des pixels connectés entre eux dans une image binaire pour former des groupes. L'image binaire est obtenue en "soustrayant" l'image analysée à un modèle de l'image d'arrière plan [1]. Dans l'algorithme, un seuil d'aire minimum de pixels connectés entre eux est imposé avant de créer une *bounding box* / un groupe . Ce seuil minimum permet de séparer les piétons des voitures, celles-ci ayant un nombre de pixels les représentant bien supérieur à celui des piétons (voir figure 7.1).

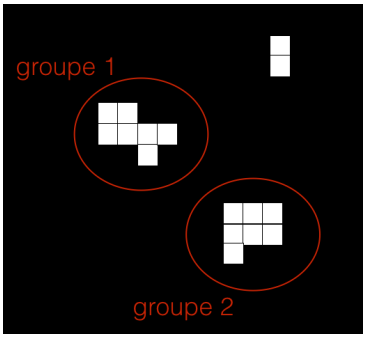


FIGURE 7.1 – Représentation d'une image binaire avec formation de groupes si l'aire minimum de pixels connectés est dépassée

Afin d'obtenir cette image binaire, il faut utiliser un détecteur dit "de premier plan". Ce détecteur va utiliser la méthode de *background substraction* [12] [5] représentée figure (7.2).

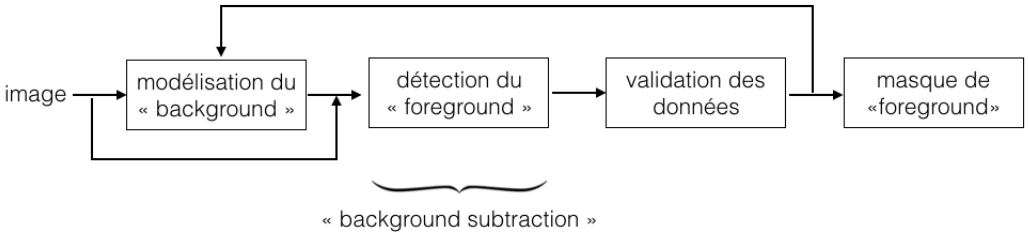


FIGURE 7.2 – Schéma de l'algorithme de la *background substraction*

Dans un premier temps, le détecteur va réaliser son modèle de l'arrière plan (*background*) en utilisant le modèle du mélange gaussien (*Gaussian mixture models* [35]). Le détecteur calcule, sur un nombre de *frames* définis (40 dans l'algorithme testé), l'intensité lumineuse des pixels en la représentant par une somme pondérée de gaussiennes. Une fois cette étape terminée, l'algorithme du schéma (7.2) peut être suivi :

1. Une nouvelle image est analysée.
2. Celle-ci est comparée en intensité de pixels par rapport au modèle du *background*. Si la différence entre la luminosité des deux pixels est supérieure à un certain seuil, alors le pixel

1. Le but est d'analyser si il y a eu du mouvement sur l'image et à quels endroits.

est considéré comme faisant partie du *foreground* (cela signifie qu'il y a eu du mouvement sur l'image).

3. Une fois tous les pixels définis soit en noir (0 = *background*) soit en blanc (1 = *foreground*), un filtre est appliqué afin de ne garder que des pixels valides (ce qui supprime une bonne partie du bruit, par exemple : des feuilles d'arbre qui bougent en arrière plan).
4. L'image résultante est utilisée pour mettre à jour la modélisation du *background*.

Une fois l'étape de la suppression de l'arrière plan (*background substraction*) terminée, on obtient l'image binaire utilisée dans l'analyse *blob*. Les *bounding boxes* résultantes (les boîtes entourant les voitures) possèdent les informations suivantes : $[x\ y\ w\ h]$, soit la position du point supérieur gauche de la boite, sa largeur et sa hauteur. Ces données sont exprimées en pixels, il faut alors repasser en mètres afin de faire la correspondance avec les détections radar. A l'aide du *pinhole model* expliqué précédemment (section 4.1.2), les positions des voitures sont exprimées. Dès lors, on peut regarder si les détections radar des *frames* correspondants sont en accord avec celles-ci. A nouveau, une erreur en position peut apparaître suite à la variation des tailles des *bounding boxes*. Cette erreur est, cependant, moins gênante étant donné que le but de cette amélioration n'est pas d'exprimer directement la position à l'aide de la caméra mais bien de s'assurer qu'une voiture est présente sur le champ d'étude à une certaine position qui sera précisée par les données radar.

Par conséquent, cette technique permet un tri de données radar sans passer par une cartographie des lieux étudiés. Le détecteur est alors valable pour tout type de route.

7.1.3 Résultat de la décimation et de l'analyse blob

En réalisant un ensemble de tests, il a été décidé de décimer l'image par 32, soit de garder un pixel sur 32 (voir figure 7.3) afin d'obtenir un temps d'exécution faible. Les tests ont été réalisés à l'aide d'une vidéo que l'on analyse sur 1500 *frames* (50 secondes). Avec la résolution maximale (1920x1080), l'analyse prend 274.4 secondes contre 80.27 secondes avec la vidéo sous-échantillonnée, soit un gain de 70.74% de temps d'exécution.

FIGURE 7.3 – Exemple de décimation d'une image de résolution 1920x1080 (image de gauche) par 32 (image de droite)

L'analyse *blob* est réalisée en prenant comme aire minimale $30px^2$ et un ratio minimum dit de *background* de 0.6 (critère décidant du *foreground* ou *background*). Ces paramètres ne laissent alors apparaître que les voitures et non les piétons.

La figure (7.4) représente sur la gauche l'image décimée et les *bounding boxes* tandis que sur la droite, le masque de *foreground* final obtenu après l'algorithme, présenté à la figure (7.2). Les quatre images montrent une *bounding box* (cadre en jaune) autour d'une voiture à gauche alors que les piétons situés sur la droite ne sont pas sélectionnés. Cela confirme le choix des paramètres utilisés.

FIGURE 7.4 – Analyse *blob*

Une vidéo (*"pixel.mp4"*) de l'ensemble du test est disponible sur le drive dédié au mémoire [14]. Comme déjà mentionné, l'application même au sein de l'algorithme de *tracking* n'a pas été effectuée mais l'ensemble des outils est fonctionnel comme les tests le démontrent.

7.2 Amélioration de la précision du détecteur avec utilisation de la vision stéréo

Au vu du travail effectué, des améliorations quant aux détections de piétons pourraient être apportées. En effet, comme on a pu le voir, les performances du détecteur, sans traitement de données ultérieures, ne sont pas satisfaisantes au niveau de la précision de la position renvoyée. En fait, comme déjà mentionné dans la section (4.1.1), ceci est principalement dû aux hauteurs des *bounding boxes* qui varient aléatoirement lors de la détection d'une cible ainsi qu'à l'hypothèse forte de la taille d'un piéton fixée à 1.8 mètres (voir le modèle *pinhole* à la section 4.1.2). Nous avons donc effectué des recherches dans le but d'essayer d'améliorer la précision de la position de notre détecteur. Dès lors, une grande amélioration pouvant être apportée à ce dernier serait l'utilisation de la vision stéréo [6] [9]. Celle-ci consiste en l'utilisation de deux caméras filmant une scène simultanément à partir de deux points de vue différents. L'enregistrement et la reconstitution en 3D est alors possible. De multiples techniques existent en stéréoscopie. L'idée que nous voulons implémenter suivrait le principe de la vision binoculaire [2], c'est à dire que les deux caméras sont écartées très légèrement, leurs axes optiques [3] sont parallèles et les deux caméras observent la même scène (figure 7.5). Les restrictions imposées exigent d'abord que les deux caméras soient identiques (mêmes paramètres intrinsèques,...), parallèlement alignées et réglées pour filmer la même scène au même moment.

2. Les caméras sont positionnées de façon similaire à nos yeux.
3. Axe passant par le centre de la lentille et perpendiculaire au plan de la lentille.

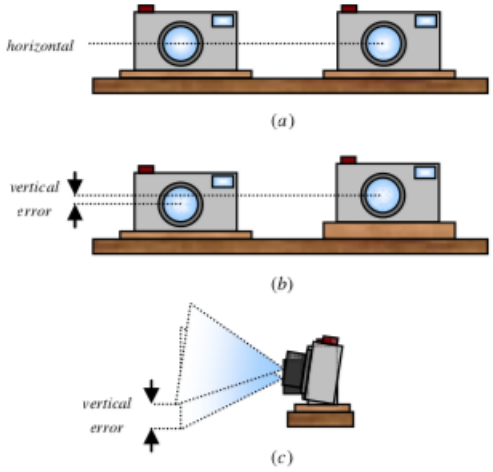


FIGURE 7.5 – (a) Position exacte des deux caméras pour la stéréo vision, (b) et (c) cas de mauvais alignement des caméras à éviter sinon cela entraîne des erreurs de calcul [6]

Dans le but de permettre l'implémentation future ainsi que la justification de la vision stéréo pour la détection de piétons, nous présentons, ici, la méthode permettant de calculer la position d'une cible dans le monde réel en mètres (plan x-y). Cette technique de calcul se base sur la disparité binoculaire entre les deux images prises par les caméras. Elle est définie comme la différence en pixels de la position de la même cible dans les deux images [4]. Dans un premier temps, la méthode permettant de détecter la cible sur les deux images ainsi que l'obtention de la disparité seront expliquées. Ensuite, la technique de calcul des distances dans le monde réel sera décrite. Finalement, nous pourrons discuter ce choix d'amélioration.

7.2.1 Méthode de détections en stéréo

Tout d'abord, le type de détecteur proposé, ici, utilise l'algorithme de détection une seule fois. En effet, la cible va être détectée et localisée sur l'image de gauche (avec la caméra de gauche) à l'aide du détecteur conventionnel implémenté. Ensuite, la même cible est automatiquement localisée sur l'image de droite à l'aide d'un algorithme d'optimisation cherchant la différence minimum entre deux images. La méthode est la suivante. La fenêtre de détection dans l'image de gauche est connue (ses dimensions en pixels, sa position,...). Dans l'image de droite, on sait dès lors définir une zone de recherche dont la hauteur ainsi que la position verticale sont identiques à celle de l'image de gauche (car les deux caméras sont alignées parallèlement) tandis que la largeur est plus grande (étape 1 figure 7.6). Dans cette zone, une fenêtre de la même taille, démarrant au même endroit que la position sur l'image de gauche, glisse horizontalement pixel par pixel et compare les caractéristiques de chaque endroit encadré avec la fenêtre située dans l'image de gauche (étape 2 figure 7.6). La fenêtre ayant les caractéristiques les plus ressemblantes représente donc la *bounding box* encadrant la cible dans l'image de droite. On connaît donc sa position sur l'image de droite et le déplacement qu'elle a dû effectuer sur l'image par rapport à sa position initiale ce qui représente la disparité binoculaire (étape 3 figure 7.6).

4. Effectivement, comme les caméras filment la même scène de façon décalée en position, une cible identique ne se trouve pas au même endroit sur les deux images

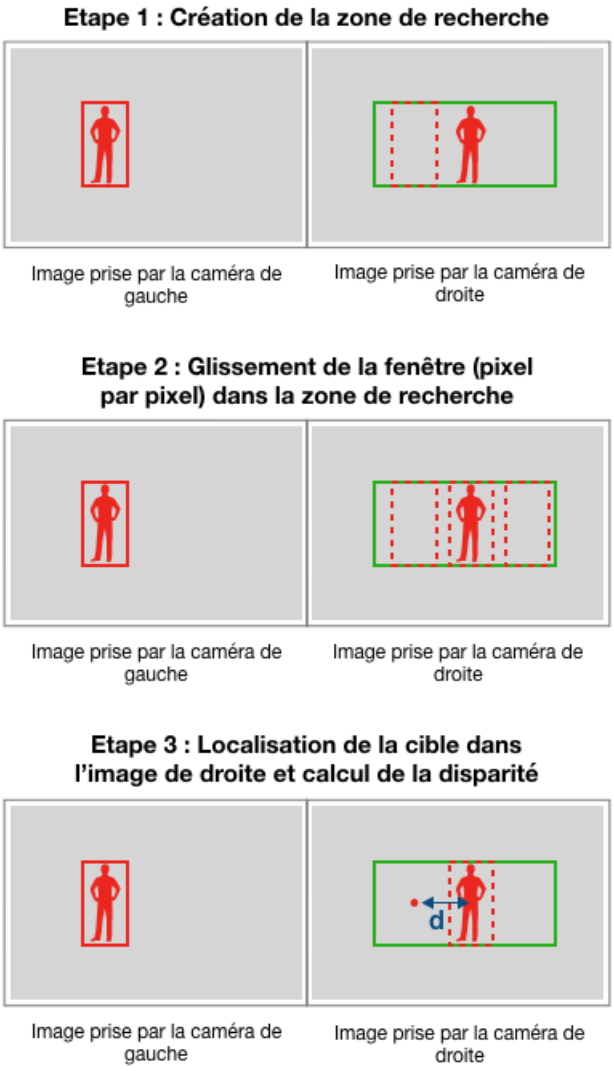


FIGURE 7.6 – Etapes de l'algorithme de détection avec deux caméras en stéréo

7.2.2 Calcul des distances en stéréo

Etant donné que l'on connaît cette distance horizontale pour le même objet sur les deux images, on peut maintenant, calculer sa distance par rapport aux caméras. Pour ce faire, on utilise le schéma figure (7.7). On définit pour celui-ci (en rouge ce sont les paramètres différents pour chaque caméra tandis qu'en noir ce sont les paramètres identiques à chacune) :

— CG : la position de la caméra de gauche
— CD : la position de la caméra de droite
— f : la distance focale de chaque caméra (identique)
— Φ_0 : l'angle d'ouverture de chaque caméra (identique)
— x_0 : la largeur de l'image (en pixels)
— b : la distance entre les deux caméras (en mètres)
— Φ_1 et Φ_2 : l'angle entre l'axe optique et la cible détectée pour chaque caméra
— Deux lignes pointillées : l'axe optique pour chaque caméra (ils sont parallèles)
— D : la distance (en y) entre les caméras et la cible (l'inconnue)

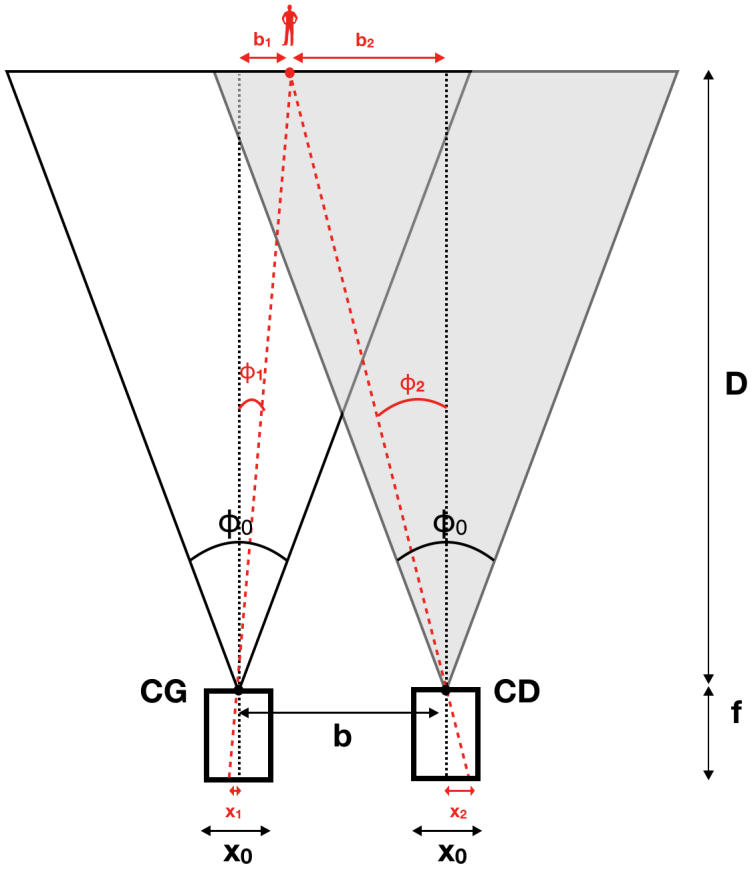


FIGURE 7.7 – Schéma représentant le cas de deux caméras détectant une cible

En utilisant la trigonométrie, on sait que :

$$b = b_1 + b_2 = D \cdot tan(\Phi_1) + D \cdot tan(\Phi_2)$$

$$D = \frac{b}{tan(\Phi_1) + tan(\Phi_2)}$$

On définit alors, dans le domaine caméra :

$$tan(\Phi_1) = \frac{-x_1}{f} \qquad tan(\Phi_2) = \frac{x_2}{f} \qquad tan(\frac{\Phi_0}{2}) = \frac{\frac{x_0}{2}}{f}$$

On obtient alors :

$$(1) \quad \frac{-x_1}{\frac{x_0}{2}} = \frac{tan(\Phi_1)}{tan(\frac{\Phi_0}{2})} \qquad (2) \quad \frac{x_2}{\frac{x_0}{2}} = \frac{tan(\Phi_2)}{tan(\frac{\Phi_0}{2})}$$

En additionnant les équations (1) et (2) et en remplaçant dans l'équation de D, on a :

$$D = \frac{b \cdot x_0}{2 \cdot tan(\frac{\Phi_0}{2}) \cdot (x_2 - x_1)}$$

Avec $(x_2 - x_1)$ qui est la disparité (= d) obtenue précédemment (section 7.2.1). Un facteur d'erreur d'alignement des caméras peut également être pris en compte, ce facteur doit être obtenu expérimentalement et cette angle d'erreur δ s'ajoute dans la tangente :

$$D = \frac{b \cdot x_0}{2 \cdot tan(\frac{\Phi_0}{2} + \delta) \cdot (x_2 - x_1)}$$

On peut également définir cette distance D en fonction de la distance focale, pour ce faire on utilise le principe des triangles semblables :

$$(3) \quad \frac{b_1}{D} = \frac{-x_1}{f} \qquad (4) \quad \frac{b_2}{D} = \frac{x_2}{f}$$

Etant donné que $b = b_1 + b_2$, on obtient directement :

$$D = \frac{b \cdot f}{(x_2 - x_1)}$$

D étant la distance en y, la distance en x peut être obtenue par rapport à chacune des caméras en utilisant les équations (3) et (4).

Au vu de ces calculs, il apparaît que pour déterminer la distance de la cible dans le monde réel il suffit de connaître la distance entre les deux caméras b (en mètres), la largeur de l'image x_0 (en pixels), la distance focale f en pixels (ou l'angle d'ouverture de la caméra Φ_0) et la disparité d en pixels (c'est à dire la différence horizontale entre le même objet vu sur chaque image).

7.2.3 Justification de l'implémentation

Après avoir effectué les différents calculs permettant d'implémenter ce modèle en caméra stéréo, quelques remarques doivent être faites.

Tout d'abord, il apparaît que pour obtenir la distance du piéton, aucune information concernant sa taille n'est nécessaire (l'hypothèse de la hauteur d'un piéton ne doit plus être faite). De plus, la taille de sa *bounding box* l'encadrant n'est plus requise. On voit donc que tout ce qui amenait des défauts à la précision de la position dans le détecteur n'est plus utilisé. Ce qui laisse à penser que la stéréo vision serait une amélioration efficace dans notre algorithme. Des tests avec deux caméras devront quand même être réalisés pour obtenir un modèle d'erreur sur la position quant à cette nouvelle méthode de localisation. Généralement, l'utilisation de la vision stéréo est reconnue comme étant un bon atout pour la précision de la position des cibles.

Pour finir, il faut tout de même mentionner que ce type d'algorithme reprenant deux caméras peut augmenter le coût des calculs et de là ralentir la vitesse de détection. Si, un jour, ce système doit fonctionner en temps réel, cette idée devra être analysée plus profondément aussi bien dans la précision qu'elle apporte que dans le temps supplémentaire qu'elle ajoute à l'algorithme.

7.3 Objectifs et applications futures

Après avoir discuté et développé les deux types d'améliorations futures directement implémentables dans l'algorithme de suivi, il est intéressant de s'attarder sur d'autres objectifs possibles que ce dispositif pourrait remplir. Cette section reprend donc des suggestions d'idées concernant le futur de l'application de ce système.

Tout d'abord, une première amélioration serait de permettre l'utilisation de ce dispositif sur des carrefours à plus grande échelle. Ceux-ci contiendraient beaucoup plus de piétons, de vélos, de motos, de bus et de voitures. Dès lors des modifications doivent être apportées. En effet, comme on a pu le constater dans l'analyse du cas d'étude (chapitre 6), les vélos et les motos sont souvent pris en compte dans le comptage de piétons traversant la route (ce qui fausse le compte de traversée). Il y a également les bus ou les camions qui sont des cas difficiles à gérer pour le comptage de voitures. Il faudrait donc améliorer l'algorithme pour contrôler ces cas-là. De plus, dans le cas de carrefours fréquentés, le dispositif ne pourrait pas se trouver à hauteur d'homme

car sa vision serait souvent gênée par des piétons passant juste devant [5]. L'idée consiste donc à mettre ce dispositif en surélévation et avoir une vue de haut du carrefour sans interruption de vision. Il faudrait donc adapter l'algorithme à ce changement de hauteur. Finalement, un réseau de plusieurs dispositifs pourraient être imaginés (chaque dispositif analyserait sa route transversale traversée par les piétons et sa route longitudinale pour les voitures), leurs données seraient fusionnées pour réaliser une analyse plus précise. Il est évident que pour réaliser ce genre d'applications, le système ne peut plus être branché directement à un ordinateur pour la prise de données, comme nous le faisions pour les tests, mais celle-ci doit se faire à l'aide de micro contrôleur sur système embarqué tel que des Raspberry Pi [22].

Ensuite, une seconde amélioration serait de rendre cette analyse en temps réel. En effet, tout cet algorithme se réalise après la prise de données. Les informations sont prises sur la route pendant un certain temps et après, celles-ci sont traitées sur un ordinateur. Durant tout le travail, peu d'importance a été accordée à l'amélioration de la rapidité des calculs du programme. Cependant, si le fonctionnement se fait en temps réel, plusieurs modifications devraient être apportées notamment et surtout en ce qui concerne le détecteur. Effectivement, celui-ci prend 0.2 seconde à analyser une image, cela aura donc un impact important sur le direct car la caméra filme à du 30 *fps* (cela signifie que pour 1 seconde de vidéo, l'analyse dure 6 secondes). Des décisions devront être prises quant à ce détecteur. La première idée serait de l'utiliser une fois tous les 'n' *frames* et de laisser les prédictions opérées entre les fonctionnements du détecteur. Cela conduirait, cependant, à de nombreuses pertes de piétons et à des manques d'informations quant à la position du piéton à tout instant. Une autre suggestion, plus adéquate, serait de faire appel à un nouveau détecteur axé sur le temps réel (rapidité de détection) plus que sur la qualité de détection. Celui-ci pourrait être par exemple l'algorithme YOLO [6] utilisant un réseau neuronal unique pour la localisation et la classification. Cette façon de détecter est beaucoup plus rapide que notre détecteur et permettrait de faire du temps réel. A nouveau, le processing devra se faire à l'aide de micro contrôleurs nécessitant un type de programmation plus bas niveau tel que le langage C ou C++.

L'idée derrière ce principe de temps réel serait de grouper le système avec des méthodes conventionnelles de signalisation telles qu'un feu ou encore des panneaux alertant du danger. Par exemple, le dispositif serait installé sur des zones fréquentées (comme le cas d'étude) et réagirait en fonction de la situation. Il permettrait d'avertir une voiture roulant rapidement qu'un piéton se trouve proche de celle-ci et est sur le point de traverser et ce grâce à l'interaction analysée entre les deux. Cela permettrait donc d'analyser le danger en direct et, par conséquent, permettre d'accroître la sécurité in situ.

5. Ces cas sont arrivés fréquemment lors de nos tests car nous devions nous mettre sur un trottoir. Des piétons passaient donc souvent devant le dispositif cachant ainsi la quasi totalité du champ de vision pendant un court laps de temps.

6. *You only look once* qui est un algorithme de détection très récent (2016) [18].

Chapitre 8

Conclusion

Le suivi de cibles multiples joue un rôle important dans le monde de l'ingénierie. Tout au long de ce mémoire, une solution innovante de combinaison de deux capteurs a été développée pour répondre aux objectifs fixés. Les deux senseurs sélectionnés, une caméra et un radar, ont été utilisés pour effectuer le suivi de piétons et de voitures respectivement. L'objectif étant de détecter les cibles, récolter leurs positions temporelles et fusionner les données provenant des deux capteurs afin d'effectuer une analyse. Celle-ci permettra de ressortir la fréquentation d'une route en terme du nombre de piétons et de voitures, leurs trajectoires respectives, la vitesse moyenne des véhicules ainsi que l'interaction entre les passants et les voitures.

Afin de remplir ces différents objectifs, les capteurs permettant de réaliser ce travail ont été sélectionnés. Le cas d'étude a également été décrit. Ce dernier est un endroit complexe, fréquenté par beaucoup de piétons et de voitures. Ensuite, une recherche théorique a été effectuée pour situer les technologies utilisées à ce jour. Un choix justifié a pu alors être établi.

La première étape de l'implémentation fut de sélectionner un détecteur de piétons et de voitures. Sans celui-ci, aucun suivi ne peut être réalisé. Le choix du détecteur de piétons s'est basé sur sa qualité et ses performances ainsi que son aspect pratique (disponible dans l'environnement sous lequel nous travaillons). La détection des voitures est quant à elle réalisée directement par le radar KMD2 qui permet de retourner des informations sur des cibles multiples.

La deuxième étape fut de développer un algorithme de suivi. Celui-ci a été imaginé dans le but de pouvoir traiter de façon similaire les données radar et caméra. Cette polyvalence permet une plus grande adaptabilité au vue d'une interaction possible entre les deux capteurs. La complexité du suivi provient principalement de détections bruitées ou manquantes ainsi que la dissociation des cibles multiples. Une structure de piste a été imaginée pour associer à chaque piéton ou voiture un historique temporel de déplacement. L'association des détections à chacune des pistes est alors réalisée par un algorithme complexe et est basée sur la distance séparant les pistes et les détections. Cette technique permet un discernement paramétrique ajustable entre les cibles. Le suivi est également amélioré par l'utilisation de filtres de Kalman qui permettent de prendre en compte les incertitudes des capteurs et de l'environnement. Chaque piste se verra attribuer un filtre adapté à la situation de la cible. Cette personnalisation permet un suivi optimal. Une fois ces méthodes implémentées, des modèles d'erreurs ont été déterminés pour chacun des capteurs utilisés rendant ainsi mieux compte de la réalité.

Après avoir réalisé le suivi, la fusion temporelle des données va permettre l'analyse combinée. Afin d'améliorer la qualité et la précision du suivi des cibles, un ensemble de pré et post-traitements ont été intégrés ainsi que l'optimisation de certains paramètres.

Cette implémentation finale de l'algorithme a alors été testée sur le cas d'étude pendant un intervalle de temps conséquent. Le but étant d'analyser le carrefour en conditions réelles à une heure de pointe. Cette expérience a pu démontrer l'efficacité des recherches effectuées, le fonctionnement de l'algorithme développé et finalement l'utilité que pourrait avoir un tel capteur pour la sécurité routière.

Au vu du travail réalisé, nous pouvons affirmer que les résultats obtenus correspondent aux objectifs fixés. La détection de piétons fonctionne parfaitement mais la précision de la localisation pourrait être améliorée par l'utilisation de deux caméras en stéréo. La détection de voitures quant à elle est très précise en position, néanmoins une sélection des données trop restrictive au cas d'étude est utilisée. Celle-ci peut être contournée par l'entraînement du radar avec la caméra. Concernant l'algorithme de suivi des voitures et des piétons, celui-ci est fonctionnel et adaptable à tout type de situations (chemin, route, carrefour,...).

Au terme de ce travail de recherche, nous sommes conscients de l'évolution réalisée mais aussi des améliorations qu'il reste à apporter. Ce mémoire ouvre donc une porte sur l'avenir de l'utilisation simultanée de deux capteurs et permet de confirmer l'efficacité d'un tel dispositif.

Annexes

A1. Plan du dispositif

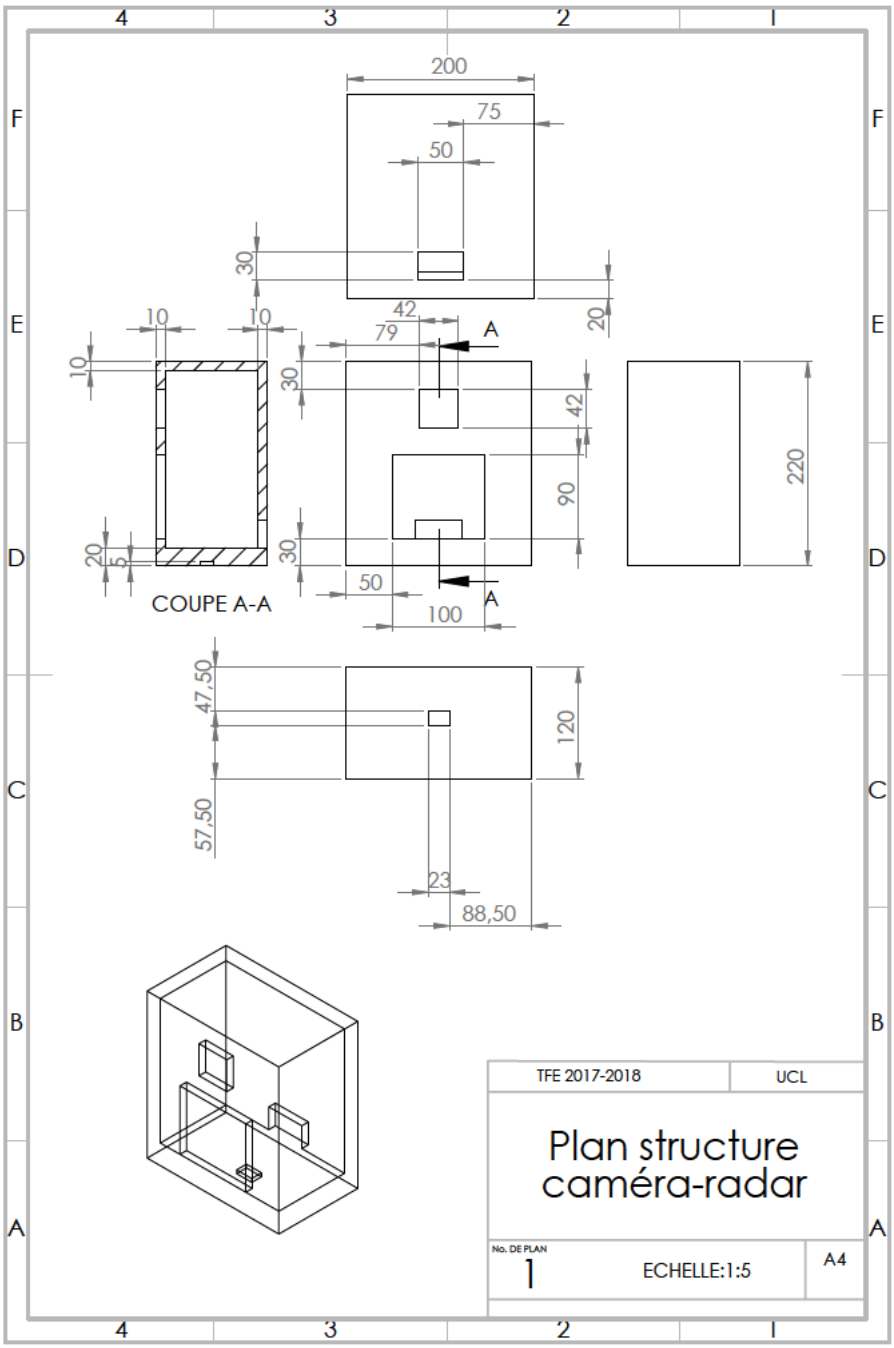


FIGURE 8.1 – Plan du dispositif

A2. Erreur du radar sur la position

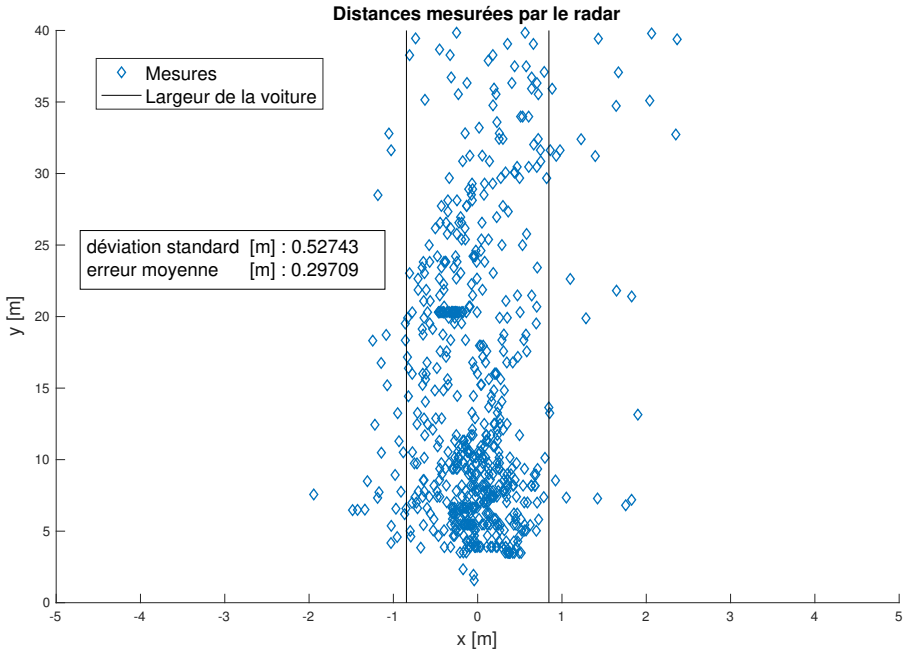


FIGURE 8.2 – Test d'erreur du radar sur la position en x d'une voiture.

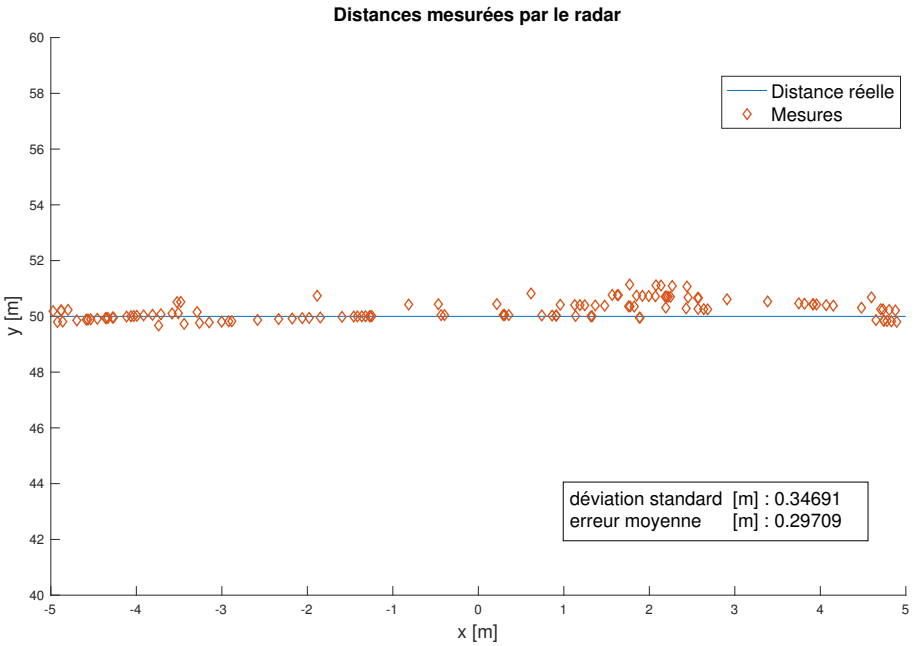


FIGURE 8.3 – Test d'erreur du radar sur la position en y d'une voiture.

A3. Tableau de l'analyse ROC caméra et radar

Analyse radar			
20 dB			
Temps auquel le frame est analysé [s]	Réalité	Fausses détections	Vraies détections
0	2	5	2
5	1	3	1
10	2	2	2
15	2	4	2
20	2	1	2
25	2	3	2
30	1	3	1
35	1	3	1
40	2	4	2
45	3	5	3
50	3	2	3
55	2	3	2
Total :	23	38	23
TPR			1
FPR			0,76
25 dB			
Temps auquel le frame est analysé [s]	Réalité	Fausses détections	Vraies détections
0	2	2	2
5	1	1	1
10	2	1	2
15	2	0	2
20	2	1	2
25	2	2	2
30	1	2	1
35	1	2	1
40	2	3	2
45	3	2	2
50	3	0	3
55	2	2	1
Total :	23	18	21
TPR			0,913043478
FPR			0,6
30 dB			
Temps auquel le frame est analysé [s]	Réalité	Fausses détections	Vraies détections
0	2	0	2
5	1	0	0
10	2	0	2
15	2	0	2
20	2	0	2
25	2	1	2
30	1	1	1
35	1	1	1
40	2	1	2
45	3	0	2
50	3	0	2
55	2	0	1
Total :	23	4	19
TPR			0,826086957
FPR			0,25
35 dB			

Temps auquel le frame est analysé [s]	Réalité	Fausses détections	Vraies détections
0	2	0	2
5	1	0	0
10	2	0	1
15	2	0	2
20	2	0	1
25	2	0	1
30	1	0	1
35	1	1	1
40	2	1	2
45	3	1	2
50	3	0	2
55	2	0	1
Total :	23	3	16
TPR			0,695652174
FPR			0,2

40 dB

Temps auquel le frame est analysé [s]	Réalité	Fausses détections	Vraies détections
0	2	0	2
5	1	0	0
10	2	0	1
15	2	0	2
20	2	0	1
25	2	0	1
30	1	0	1
35	1	1	1
40	2	0	1
45	3	0	1
50	3	0	1
55	2	0	1
Total :	23	1	13
TPR			0,565217391
FPR			0,076923077

45 dB

Temps auquel le frame est analysé [s]	Réalité	Fausses détections	Vraies détections
0	2	0	2
5	1	0	0
10	2	0	1
15	2	0	2
20	2	0	0
25	2	0	1
30	1	0	0
35	1	1	1
40	2	0	2
45	3	0	1
50	3	0	1
55	2	0	1
Total :	23	1	12
TPR			0,52173913
FPR			0,076923077

50 dB

Temps auquel le frame est analysé [s]	Réalité	Fausses détections	Vraies détections
0	2	0	2
5	1	0	0
10	2	0	1
15	2	0	1

20	2	0	0
25	2	0	1
30	1	0	0
35	1	0	0
40	2	0	2
45	3	0	1
50	3	0	1
55	2	0	1
Total :	23	0	10
TPR			0,434782609
FPR			0
55 dB			
Temps auquel le frame est analysé [s]	Réalité	Fausses détections	Vraies détections
0	2	0	0
5	1	0	0
10	2	0	1
15	2	0	0
20	2	0	0
25	2	0	1
30	1	0	0
35	1	0	0
40	2	0	2
45	3	0	0
50	3	0	1
55	2	0	0
Total :	23	0	5
TPR			0,217391304
FPR			0

TABLE 8.1: Tableau de l'analyse ROC des voitures

Analyse caméra							
10%				20%			
Image	Réalité	Fausses détections	Vraies détections	Image	Réalité	Fausses détections	Vraies détections
1	0	2	0	1	0	2	0
2	0	0	0	2	0	0	0
3	0	1	0	3	0	0	0
4	0	1	0	4	0	1	0
5	1	1	1	5	1	1	1
6	1	1	1	6	1	0	1
7	1	0	1	7	1	0	1
8	1	1	1	8	1	1	1
9	1	0	1	9	1	0	1
10	1	3	1	10	1	0	1
11	0	1	0	11	0	1	0
12	0	1	0	12	0	1	0
13	1	1	1	13	1	1	1
14	2	3	2	14	2	3	2
15	3	3	3	15	3	1	3
16	2	2	2	16	2	1	2
17	0	1	0	17	0	0	0
18	0	2	0	18	0	2	0
19	1	1	1	19	1	1	1
20	2	6	2	20	2	2	2
21	2	3	2	21	2	2	2
22	2	7	2	22	2	6	2

23	0	4	0	23	0	2	0
24	0	2	0	24	0	2	0
Total :	21	47	21	Total :	21	30	21
TPR			1	TPR			1
FPR			0,661971831	FPR			0,555555556

30%				40%			
Image	Réalité	Fausses détections	Vraies détections	Image	Réalité	Fausses détections	Vraies détections
1	0	0	0	1	0	0	0
2	0	0	0	2	0	0	0
3	0	0	0	3	0	0	0
4	0	1	0	4	0	0	0
5	1	1	1	5	1	0	1
6	1	0	1	6	1	0	1
7	1	0	1	7	1	0	1
8	1	0	1	8	1	1	1
9	1	0	1	9	1	0	1
10	1	0	1	10	1	0	1
11	0	1	0	11	0	0	0
12	0	0	0	12	0	0	0
13	1	0	1	13	1	0	1
14	2	1	2	14	2	0	2
15	3	1	3	15	3	0	3
16	2	0	2	16	2	0	2
17	0	1	0	17	0	0	0
18	0	0	0	18	0	0	0
19	1	0	1	19	1	0	1
20	2	0	2	20	2	0	2
21	2	1	2	21	2	0	2
22	2	3	2	22	2	1	2
23	0	0	0	23	0	0	0
24	0	2	0	24	0	2	0
Total :	21	12	21	Total :	21	4	21
TPR			1	TPR			1
FPR			0,333333333	FPR			0,142857143

50%				60%			
Image	Réalité	Fausses détections	Vraies détections	Image	Réalité	Fausses détections	Vraies détections
1	0	0	0	1	0	0	0
2	0	0	0	2	0	0	0
3	0	0	0	3	0	0	0
4	0	0	0	4	0	0	0
5	1	0	1	5	1	0	1
6	1	0	1	6	1	0	1
7	1	0	1	7	1	0	1
8	1	0	1	8	1	0	1
9	1	0	1	9	1	0	1
10	1	0	1	10	1	0	1
11	0	0	0	11	0	0	0
12	0	0	0	12	0	0	0
13	1	0	1	13	1	0	1
14	2	0	2	14	2	0	2
15	3	0	3	15	3	0	3
16	2	0	2	16	2	0	2
17	0	0	0	17	0	0	0
18	0	0	0	18	0	0	0
19	1	0	1	19	1	0	1
20	2	0	1	20	2	0	1

21	2	0	2	21	2	0	2
22	2	0	2	22	2	0	2
23	0	0	0	23	0	0	0
24	0	1	0	24	0	1	0
Total :	21	1	20	Total :	21	1	20
TPR		0,952380952		TPR		0,952380952	
FPR		0,04		FPR		0,04	

70%				80%			
Image	Réalité	Fausses détections	Vraies détections	Image	Réalité	Fausses détections	Vraies détections
1	0	0	0	1	0	0	0
2	0	0	0	2	0	0	0
3	0	0	0	3	0	0	0
4	0	0	0	4	0	0	0
5	1	0	0	5	1	0	0
6	1	0	1	6	1	0	1
7	1	0	1	7	1	0	1
8	1	0	1	8	1	0	1
9	1	0	1	9	1	0	1
10	1	0	1	10	1	0	0
11	0	0	0	11	0	0	0
12	0	0	0	12	0	0	0
13	1	0	1	13	1	0	1
14	2	0	2	14	2	0	1
15	3	0	3	15	3	0	2
16	2	0	1	16	2	0	1
17	0	0	0	17	0	0	0
18	0	0	0	18	0	0	0
19	1	0	1	19	1	0	1
20	2	0	1	20	2	0	0
21	2	0	2	21	2	0	1
22	2	0	1	22	2	0	0
23	0	0	0	23	0	0	0
24	0	0	0	24	0	0	0
Total :	21	0	17	Total :	21	0	11
TPR		0,80952381		TPR		0,523809524	
FPR		0		FPR		0	

90%			
Image	Réalité	Fausses détections	Vraies détections
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	1	0	0
6	1	0	0
7	1	0	1
8	1	0	0
9	1	0	0
10	1	0	1
11	0	0	0
12	0	0	0
13	1	0	0
14	2	0	0
15	3	0	1
16	2	0	1
17	0	0	0
18	0	0	0

19	1	0	0
20	2	0	0
21	2	0	1
22	2	0	0
23	0	0	0
24	0	0	0
Total :	21	0	5
TPR			0,238095238
FPR			0

TABLE 8.2: Tableau de l'analyse ROC des piétons

A4. Tableaux de l'analyse du cas d'étude

Piétons		
Piéton	Sens de traversée	Temps d'apparition [h :min :s]
1	'droite'	18 :06 :35
2	'droite'	18 :06 :52
3	'gauche->droite'	18 :07 :10
4	'gauche->droite'	18 :07 :12
5	'droite'	18 :07 :46
6	'droite'	18 :08 :03
7	'droite->gauche '	18 :08 :14
8	'droite'	18 :09 :30
9	'gauche->droite'	18 :10 :38
10	'gauche->droite'	18 :11 :34
11	'droite'	18 :11 :45
12	'droite->gauche '	18 :12 :1
13	'gauche->droite'	18 :13 :51
14	'gauche->droite'	18 :15 :19
15	'droite->gauche '	18 :18 :8
16	'gauche'	18 :18 :19
17	'droite->gauche '	18 :19 :49
18	'droite->gauche '	18 :21 :55
19	'gauche->droite'	18 :22 :34
20	'droite'	18 :24 :18
21	'droite'	18 :24 :45
22	'droite'	18 :25 :3
23	'droite->gauche '	18 :26 :13
24	'droite'	18 :27 :56
25	'droite'	18 :28 :1
26	'gauche'	18 :30 :13
27	'gauche->droite'	18 :30 :29
28	'droite'	18 :31 :60
29	'droite->gauche '	18 :32 :21
30	'droite->gauche '	18 :36 :16
31	'gauche->droite'	18 :36 :16

TABLE 8.3: Structure des piétons renvoyée par l'algorithme de *tracking*

VOITURES				
Voiture	Sens	Vitesse moyenne [km/h]	Accélération/Décélération	Temps d'apparition [h :min :s]
1	'-'	24,7	'accélération'	18 :06 :24
2	'-'	17,4	'décélération'	18 :06 :24
3	'-'	40,8	'décélération'	18 :06 :34
4	'-'	46,6	'accélération'	18 :06 :42

5	'+'	43,1	'décélération'	18 :06 :48
6	'+'	41,3	'accélération'	18 :07 :11
7	'-'	17,0	'accélération'	18 :07 :23
8	'-'	17,4	'accélération'	18 :07 :27
9	'-'	29,3	'décélération'	18 :07 :58
10	'-'	37,0	'accélération'	18 :08 :4
11	'+'	40,6	'décélération'	18 :08 :10
12	'+'	42,7	'décélération'	18 :08 :25
13	'+'	40,8	'accélération'	18 :08 :27
14	'-'	44,3	'accélération'	18 :08 :42
15	'-'	35,4	'accélération'	18 :08 :60
16	'-'	31,2	'accélération'	18 :09 :19
17	'+'	31,0	'décélération'	18 :09 :25
18	'-'	32,2	'décélération'	18 :09 :27
19	'-'	20,3	'accélération'	18 :09 :30
20	'+'	39,6	'décélération'	18 :9 :38
21	'-'	6,0	'décélération'	18 :10 :1
22	'-'	34,0	'accélération'	18 :10 :14
23	'-'	46,0	'accélération'	18 :10 :20
24	'+'	39,0	'décélération'	18 :10 :29
25	'+'	34,4	'accélération'	18 :10 :43
26	'-'	30,4	'décélération'	18 :10 :50
27	'-'	24,4	'décélération'	18 :11 :1
28	'-'	20,4	'accélération'	18 :11 :6
29	'+'	40,5	'décélération'	18 :11 :12
30	'+'	11,8	'accélération'	18 :11 :12
31	'-'	29,6	'accélération'	18 :11 :13
32	'-'	14,6	'décélération'	18 :11 :16
33	'-'	18,7	'décélération'	18 :11 :24
34	'-'	25,2	'accélération'	18 :11 :27
35	'-'	16,9	'accélération'	18 :11 :28
36	'-'	18,0	'accélération'	18 :11 :30
37	'+'	29,4	'décélération'	18 :11 :34
38	'+'	25,8	'accélération'	18 :11 :36
39	'+'	23,1	'décélération'	18 :11 :41
40	'-'	40,1	'accélération'	18 :11 :51
41	'-'	38,1	'accélération'	18 :12 :3
42	'-'	41,9	'accélération'	18 :12 :41
43	'-'	37,4	'accélération'	18 :12 :60
44	'-'	43,6	'décélération'	18 :13 :13
45	'+'	24,5	'accélération'	18 :13 :19
46	'+'	34,4	'accélération'	18 :13 :19
47	'+'	31,0	'accélération'	18 :13 :20
48	'-'	40,1	'accélération'	18 :13 :22
49	'+'	33,5	'décélération'	18 :13 :25
50	'-'	23,9	'accélération'	18 :13 :30
51	'-'	39,9	'accélération'	18 :13 :34
52	'-'	36,5	'décélération'	18 :13 :44
53	'+'	35,9	'accélération'	18 :13 :54
54	'-'	38,5	'accélération'	18 :14 :9
55	'-'	15,0	'décélération'	18 :14 :13
56	'-'	45,0	'décélération'	18 :14 :28
57	'-'	18,2	'accélération'	18 :14 :37
58	'-'	24,8	'accélération'	18 :14 :47
59	'+'	40,0	'décélération'	18 :14 :55
60	'+'	36,5	'décélération'	18 :14 :59

61	'-'	42,8	'accélération'	18 :15 :1
62	'-'	43,7	'accélération'	18 :15 :16
63	'-'	43,4	'décélération'	18 :15 :19
64	'-'	35,5	'décélération'	18 :15 :47
65	'-'	11,6	'accélération'	18 :15 :56
66	'+'	41,9	'décélération'	18 :15 :58
67	'+'	38,8	'accélération'	18 :16 :0
68	'+'	42,1	'décélération'	18 :16 :7
69	'-'	23,7	'accélération'	18 :16 :14
70	'-'	27,0	'décélération'	18 :16 :18
71	'-'	23,9	'accélération'	18 :16 :26
72	'-'	35,9	'accélération'	18 :16 :31
73	'-'	22,8	'accélération'	18 :16 :39
74	'-'	35,7	'accélération'	18 :16 :55
75	'-'	34,7	'accélération'	18 :17 :1
76	'+'	23,7	'décélération'	18 :17 :5
77	'+'	25,5	'décélération'	18 :17 :5
78	'-'	19,8	'décélération'	18 :17 :7
79	'-'	17,4	'accélération'	18 :17 :18
80	'-'	37,0	'accélération'	18 :17 :40
81	'-'	37,3	'décélération'	18 :17 :44
82	'-'	33,9	'accélération'	18 :18 :13
83	'+'	37,6	'décélération'	18 :18 :26
84	'-'	11,9	'décélération'	18 :18 :29
85	'-'	38,1	'décélération'	18 :18 :34
86	'-'	40,8	'accélération'	18 :18 :42
87	'-'	18,2	'accélération'	18 :19 :17
88	'-'	32,5	'accélération'	18 :19 :50
89	'-'	29,1	'décélération'	18 :19 :53
90	'+'	28,1	'décélération'	18 :19 :54
91	'+'	37,3	'accélération'	18 :19 :54
92	'-'	30,6	'décélération'	18 :19 :56
93	'-'	38,3	'accélération'	18 :20 :3
94	'+'	34,5	'décélération'	18 :20 :10
95	'-'	32,3	'décélération'	18 :20 :11
96	'+'	18,6	'décélération'	18 :20 :21
97	'-'	40,1	'accélération'	18 :20 :34
98	'-'	34,9	'décélération'	18 :20 :38
99	'-'	42,8	'accélération'	18 :20 :47
100	'-'	20,7	'accélération'	18 :20 :56
101	'-'	38,6	'accélération'	18 :20 :59
102	'-'	45,7	'accélération'	18 :21 :9
103	'+'	24,0	'décélération'	18 :21 :31
104	'+'	26,4	'accélération'	18 :21 :32
105	'-'	36,5	'décélération'	18 :21 :53
106	'-'	26,9	'accélération'	18 :22 :9
107	'+'	38,1	'décélération'	18 :22 :21
108	'-'	40,7	'décélération'	18 :22 :28
109	'+'	24,6	'décélération'	18 :22 :32
110	'+'	29,9	'accélération'	18 :22 :35
111	'-'	35,7	'accélération'	18 :22 :37
112	'+'	31,4	'décélération'	18 :22 :42
113	'-'	36,1	'accélération'	18 :22 :52
114	'+'	44,2	'décélération'	18 :23 :18
115	'+'	34,2	'décélération'	18 :23 :28
116	'-'	23,0	'accélération'	18 :24 :2

117	'+'	25,7	'décélération'	18 :24 :8
118	'-'	35,3	'accélération'	18 :24 :9
119	'+'	33,3	'accélération'	18 :24 :41
120	'-'	41,7	'accélération'	18 :24 :52
121	'-'	17,3	'accélération'	18 :24 :57
122	'+'	28,1	'accélération'	18 :25 :0
123	'+'	40,7	'décélération'	18 :25 :5
124	'+'	41,5	'décélération'	18 :25 :9
125	'-'	5,2	'accélération'	18 :25 :11
126	'+'	45,6	'décélération'	18 :25 :12
127	'-'	19,1	'accélération'	18 :25 :40
128	'+'	37,6	'accélération'	18 :25 :49
129	'-'	34,9	'accélération'	18 :25 :52
130	'-'	47,1	'accélération'	18 :26 :9
131	'+'	34,1	'accélération'	18 :26 :34
132	'+'	34,3	'accélération'	18 :26 :47
133	'+'	44,6	'décélération'	18 :26 :56
134	'-'	42,1	'décélération'	18 :26 :57
135	'-'	12,7	'accélération'	18 :27 :18
136	'+'	24,3	'décélération'	18 :27 :22
137	'+'	32,7	'accélération'	18 :27 :22
138	'+'	24,5	'décélération'	18 :27 :23
139	'-'	19,1	'accélération'	18 :27 :25
140	'-'	22,4	'accélération'	18 :27 :27
141	'-'	28,5	'décélération'	18 :27 :31
142	'-'	37,0	'décélération'	18 :27 :42
143	'-'	32,4	'décélération'	18 :27 :48
144	'+'	20,0	'décélération'	18 :27 :52
145	'+'	22,3	'décélération'	18 :27 :53
146	'-'	19,5	'accélération'	18 :28 :6
147	'+'	40,7	'décélération'	18 :28 :14
148	'+'	37,6	'accélération'	18 :28 :18
149	'-'	27,8	'décélération'	18 :28 :33
150	'+'	29,0	'décélération'	18 :28 :48
151	'+'	26,1	'décélération'	18 :28 :50
152	'+'	23,7	'accélération'	18 :28 :55
153	'-'	40,9	'accélération'	18 :29 :1
154	'+'	38,0	'décélération'	18 :29 :8
155	'-'	36,5	'accélération'	18 :29 :22
156	'+'	36,3	'décélération'	18 :29 :39
157	'+'	32,0	'accélération'	18 :30 :18
158	'-'	29,7	'accélération'	18 :30 :30
159	'+'	36,8	'accélération'	18 :30 :45
160	'+'	34,0	'décélération'	18 :30 :47
161	'+'	44,0	'accélération'	18 :30 :58
162	'-'	32,1	'accélération'	18 :31 :12
163	'+'	30,6	'décélération'	18 :31 :13
164	'+'	22,3	'accélération'	18 :31 :15
165	'+'	30,9	'décélération'	18 :31 :16
166	'-'	39,3	'décélération'	18 :31 :18
167	'+'	48,5	'accélération'	18 :31 :22
168	'-'	35,5	'accélération'	18 :31 :32
169	'-'	34,1	'accélération'	18 :31 :43
170	'+'	38,4	'accélération'	18 :31 :54
171	'-'	48,3	'accélération'	18 :32 :9
172	'-'	38,6	'accélération'	18 :32 :17

173	'-'	35,0	'accélération'	18 :32 :37
174	'+'	37,6	'décélération'	18 :32 :40
175	'-'	47,4	'accélération'	18 :33 :4
176	'+'	34,9	'accélération'	18 :33 :17
177	'-'	14,4	'décélération'	18 :33 :23
178	'+'	30,2	'accélération'	18 :33 :30
179	'+'	33,8	'décélération'	18 :33 :32
180	'+'	40,7	'décélération'	18 :33 :34
181	'+'	24,3	'décélération'	18 :33 :40
182	'-'	34,0	'accélération'	18 :33 :45
183	'-'	39,0	'accélération'	18 :33 :48
184	'-'	38,2	'accélération'	18 :33 :57
185	'+'	37,2	'décélération'	18 :34 :29
186	'-'	38,3	'décélération'	18 :34 :54
187	'+'	36,3	'accélération'	18 :35 :2
188	'+'	40,0	'décélération'	18 :35 :37
189	'+'	5,4	'décélération'	18 :35 :40
190	'-'	36,3	'accélération'	18 :35 :44
191	'+'	29,0	'décélération'	18 :35 :59
192	'-'	34,2	'décélération'	18 :36 :1
193	'-'	37,4	'accélération'	18 :36 :2
194	'+'	34,1	'accélération'	18 :36 :11
195	'-'	39,2	'accélération'	18 :36 :15
196	'+'	35,8	'décélération'	18 :36 :22
197	'-'	30,9	'accélération'	18 :36 :41
198	'-'	33,5	'accélération'	18 :36 :46
199	'-'	5,5	'accélération'	18 :36 :52
200	'-'	45,0	'accélération'	18 :37 :2
201	'-'	41,9	'accélération'	18 :37 :16
202	'-'	19,5	'accélération'	18 :37 :24

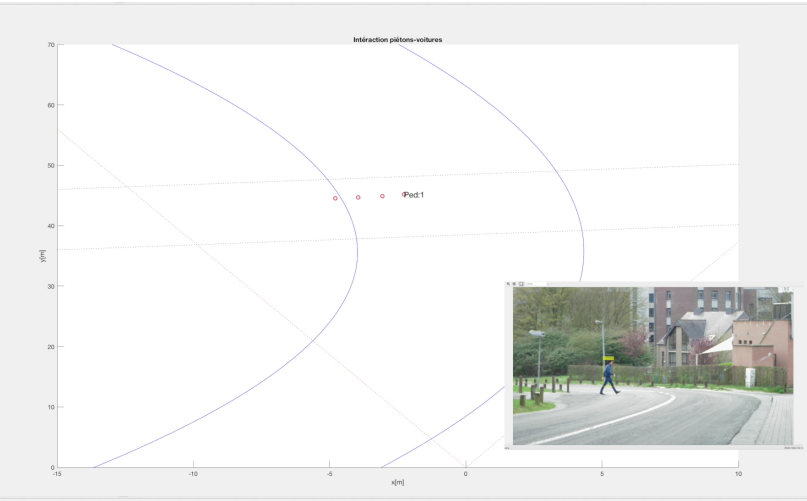
TABLE 8.4: Structure des voitures renvoyée par l'algorithme de *tracking*

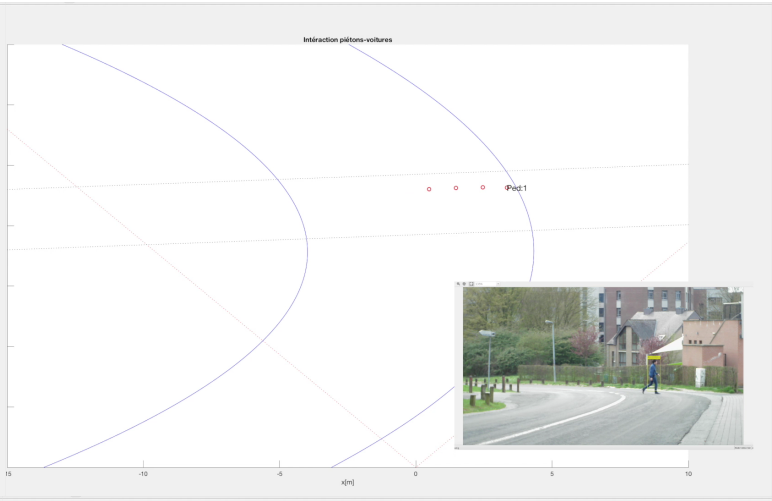
Interaction piétons-voitures				
Piéton	Voiture	Temps de l'interaction [h :min :s]	Distance minimale [m]	Vitesse à l'interaction [km/h]
1	3	18 :06 :36	13,7	41,2
	4	18 :06 :42	11,6	41,9
3	6	18 :07 :15	25,1	39,2
4	6	18 :07 :12	2,3	36,4
5	9	18 :07 :58	12,0	35,7
6	9	18 :08 :03	46,4	30,9
	10	18 :08 :04	7,6	30,2
7	11	18 :08 :15	10,4	37,1
8	19	18 :09 :32	18,0	17,2
	20	18 :09 :41	32,1	40,5
9	25	18 :10 :48	4,9	36,4
	26	18 :10 :52	1,9	30,2
	27	18 :11 :01	23,6	37,8
10	34	18 :11 :34	37,1	31,6
	36	18 :11 :34	36,0	31,8
	37	18 :11 :39	0,6	4,8
	38	18 :11 :42	0,3	2,7
	39	18 :11 :49	0,3	4,1
	40	18 :11 :51	2,7	36,4
11	39	18 :11 :45	24,4	28,9
12	41	18 :12 :05	7,7	38,5

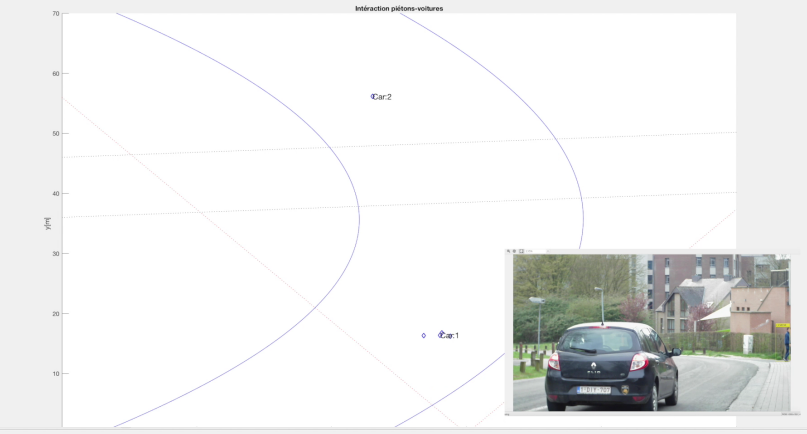
13	53	18 :13 :58	5,1	34,4
14	63	18 :15 :20	4,4	42,6
15	82	18 :18 :14	1,1	31,6
16	83	18 :18 :30	2,7	36,4
	84	18 :18 :36	3,9	7,6
	85	18 :19 :36	5,9	41,2
17	88	18 :19 :51	6,4	28,2
	89	18 :19 :53	13,3	35,7
	90	18 :19 :55	21,8	30,2
	91	18 :19 :58	3.2	39,8
	92	18 :19 :56	7,9	36,1
	93	18 :20 :03	14,3	34,4
18	105	18 :21 :56	14,5	37,1
19	109	18 :22 :35	4,9	23,4
	110	18 :22 :35	27,8	28,9
	111	18 :22 :37	6,8	33,0
20	119	18 :24 :42	44,7	29,2
21	119	18 :24 :46	5,1	30,9
	120	18 :24 :52	10,5	41,9
22	122	18 :25 :06	26,3	30,9
	123	18 :25 :09	19,7	38,5
	124	18 :25 :13	22,5	37,1
	125	18 :25 :11	52,7	5,5
	126	18 :25 :17	6,8	39,2
24	144	18 :27 :58	14,0	6,9
	145	18 :28 :03	3,9	27,5
	146	18 :28 :07	7,0	7,6
25	145	18 :28 :03	1,5	27,5
26	157	18 :30 :22	8,1	36,4
27	158	18 :30 :32	4,0	24,0
28	171	18 :32 :09	26,1	46,0
	172	18 :32 :17	21,5	35,7
	173	18 :32 :38	7,6	34,4
30	194	18 :32 :16	15,6	33,0
	195	18 :32 :17	7,6	39,5
	196	18 :32 :25	10,1	34,4
31	195	18 :24 :16	5,5	37,8
	196	18 :24 :26	6,9	35,0

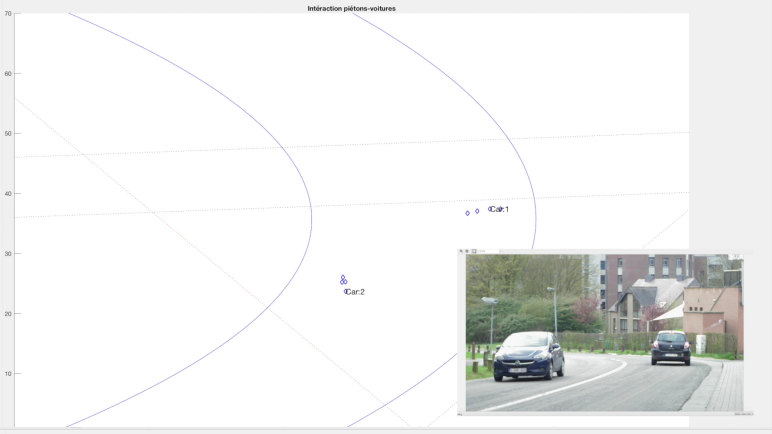
Table 8.5: Structure des interactions piétons-voitures

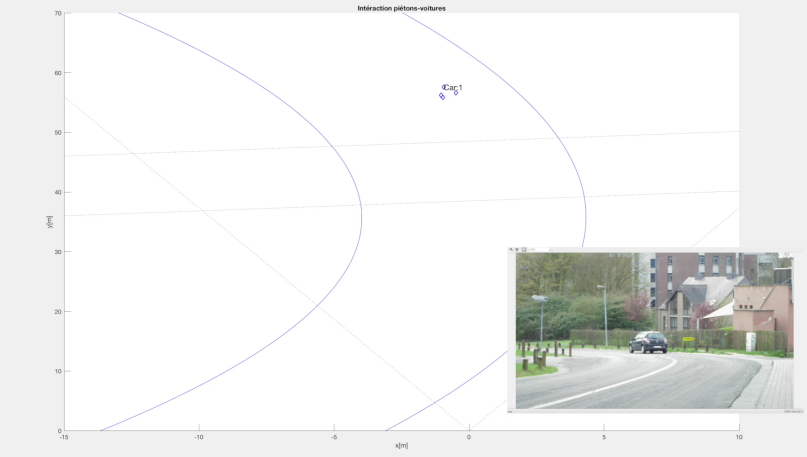
A5. Découpage de la vidéo du test de fusion

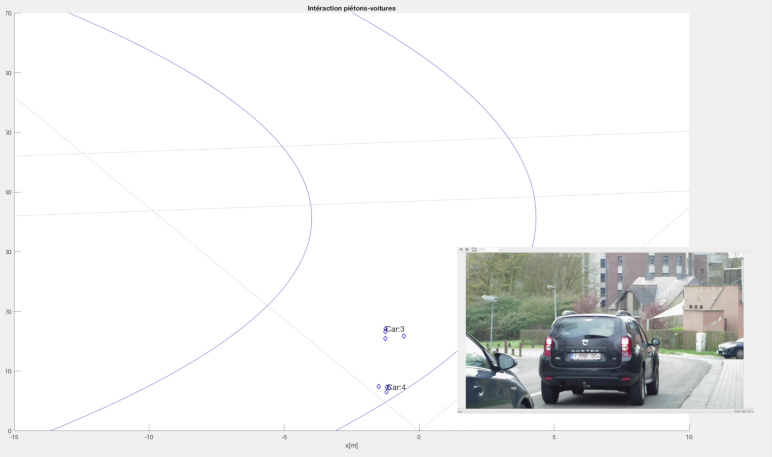


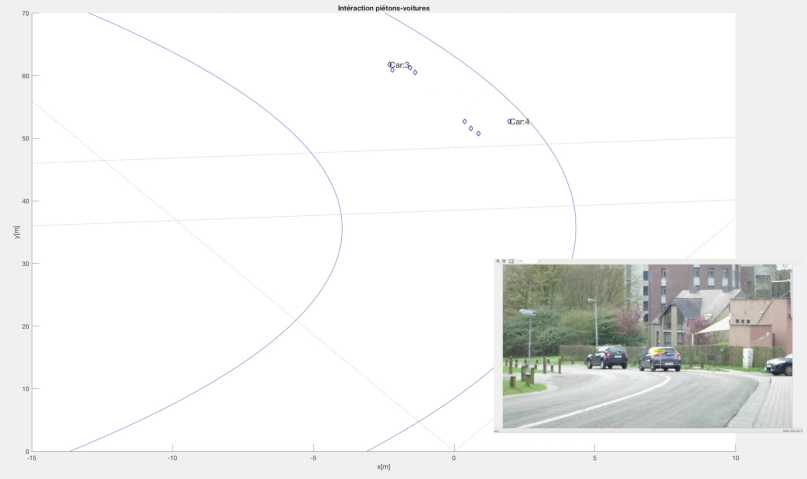


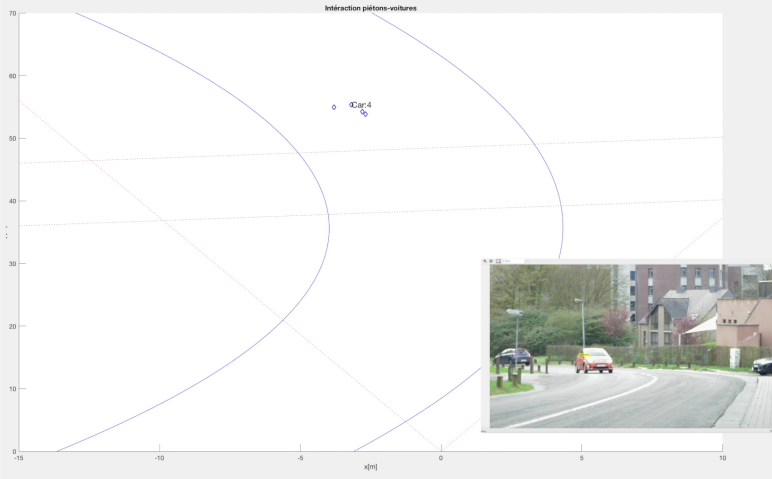


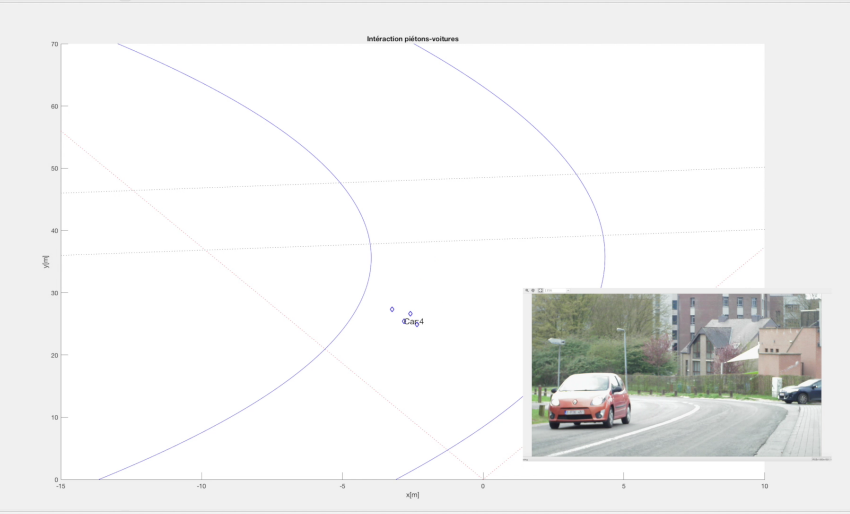


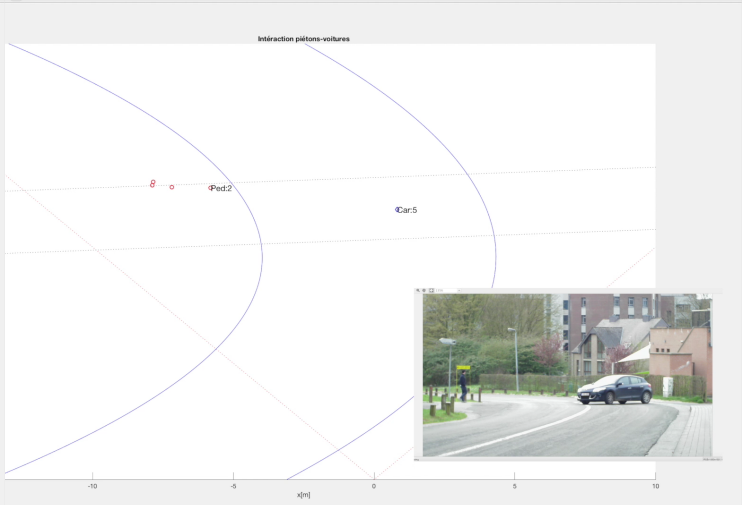


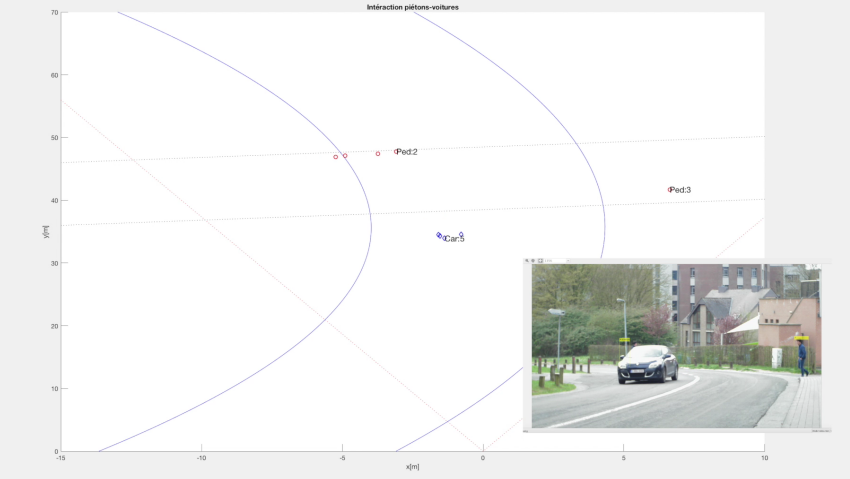


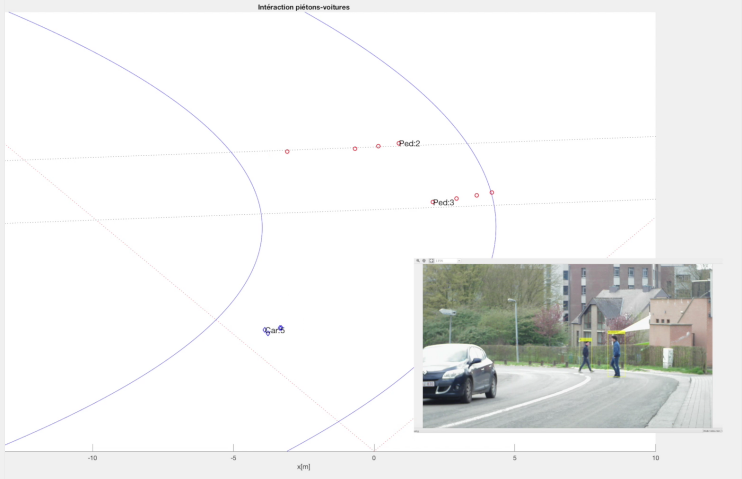


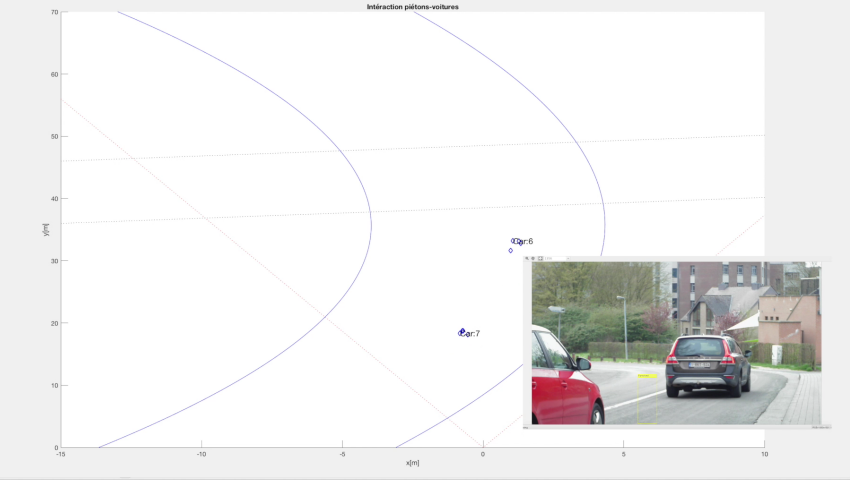


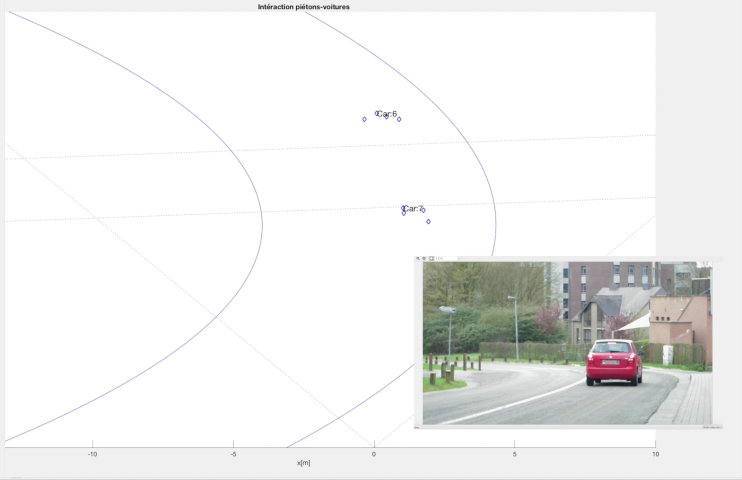

Intéraction piétons-voitures
Car:4
x[m]
y[m]


Intéraction piétons-voitures
Ped:2
Car:5
x[m]

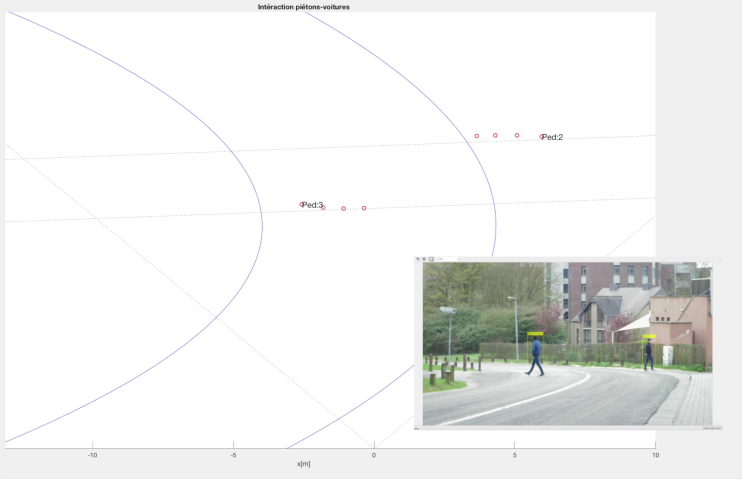

Intéraction piétons-voitures
Ped:2
Ped:3
Car:5
x[m]
y[m]

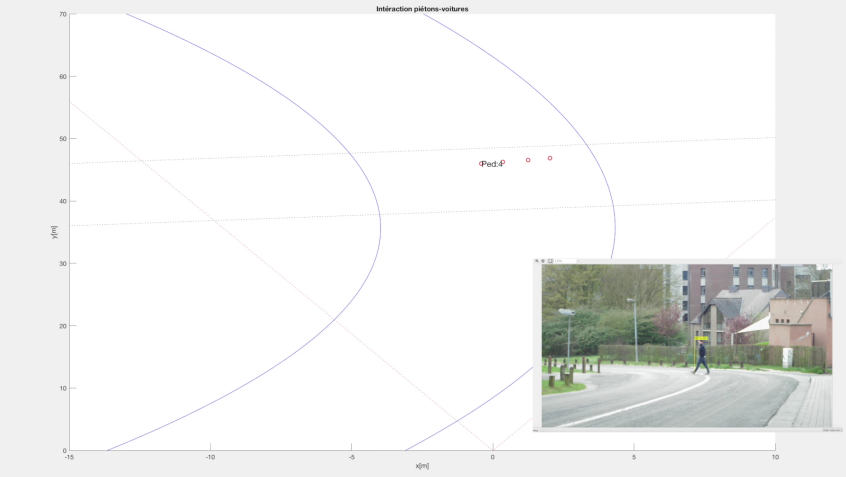

Intéraction piétons-voitures
Ped:2
Ped:3
x[m]

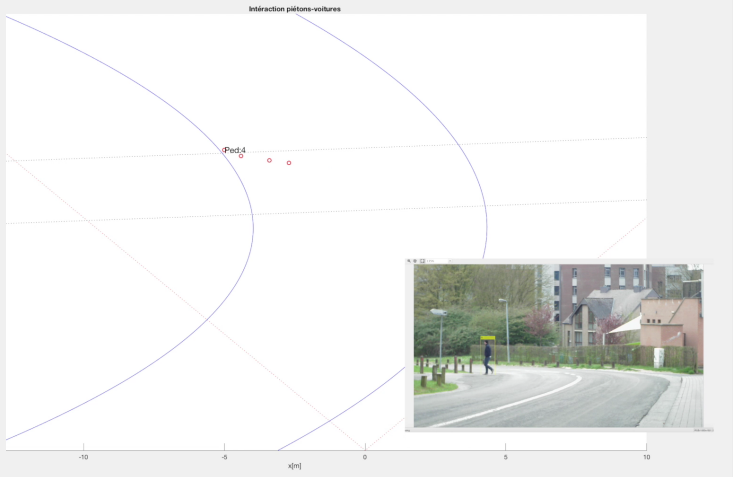

Intéraction piétons-voitures
Car:6
Car:7
x[m]
y[m]


Intéraction piétons-voitures
Car:6
Car:7
x[m]

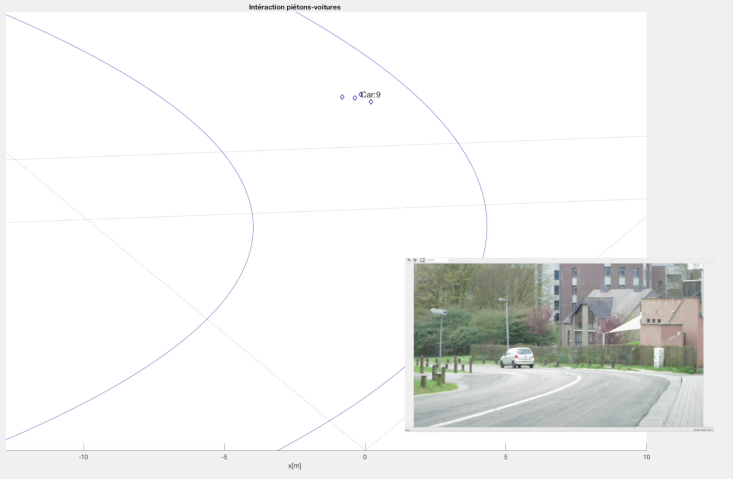

Intéraction piétons-voitures
Ped:4
Car:7
x[m]
y[m]

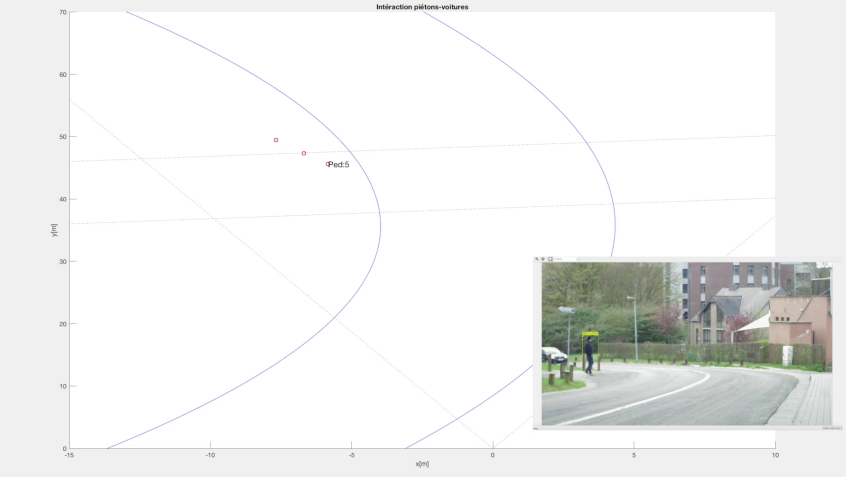

Intéraction piétons-voitures
Ped:2
Ped:3
x[m]

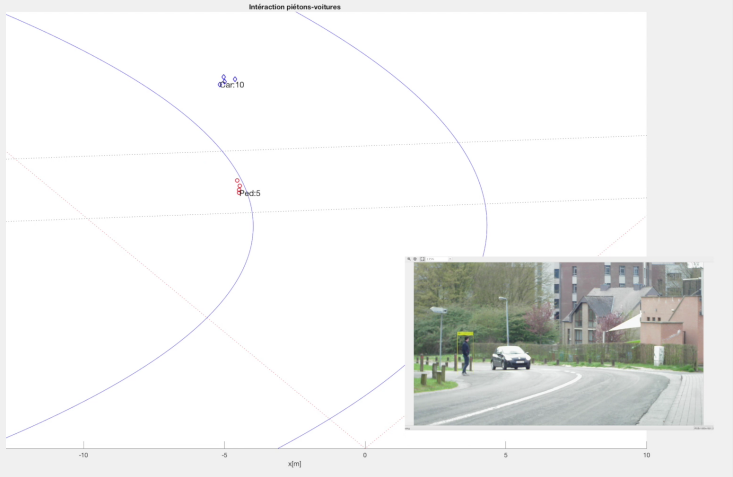

Intéraction piétons-voitures
Ped:4
y[m]
x[m]

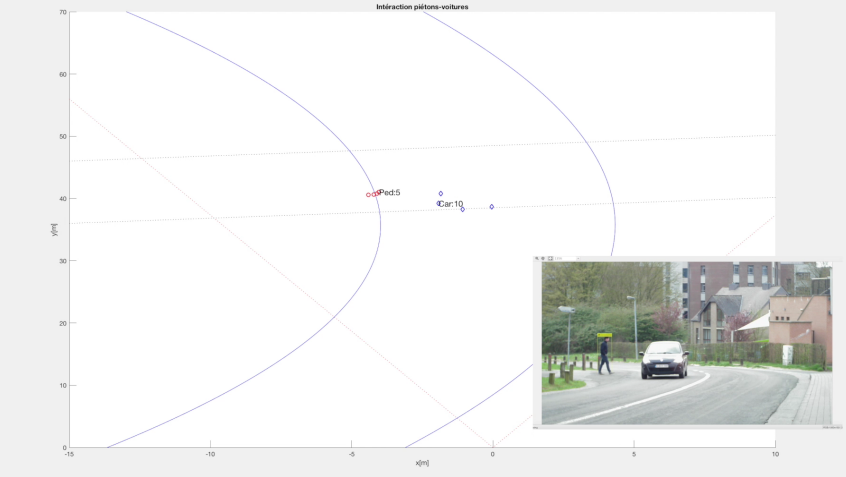

Intéraction piétons-voitures
Ped:4
x[m]

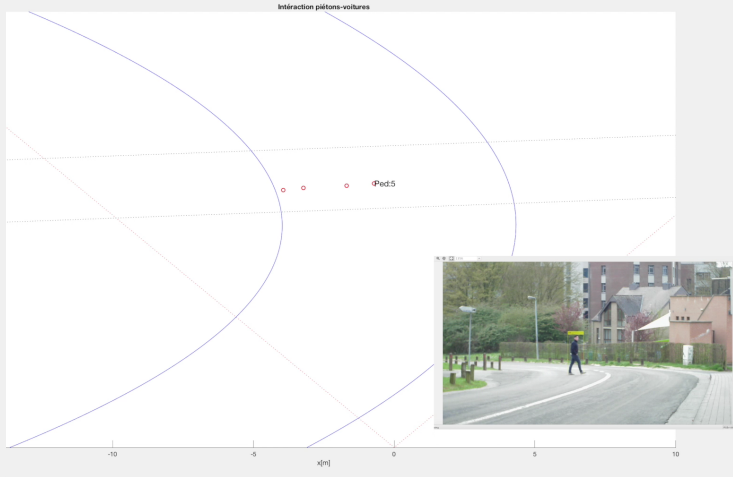

Intéraction piétons-voitures
Car:9
y[m]
x[m]


Intéraction piétons-voitures
Car:9
x[m]


Intéraction piétons-voitures
Ped:5
y[m]
x[m]


Intéraction piétons-voitures
Car:10
Ped:5
x[m]


Intéraction piétons-voitures
Ped:5
Car:10
y[m]
x[m]


Intéraction piétons-voitures
Ped:5
x[m]

Bibliographie

[1] Samuel S. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE A&E SYSTEMS MAGAZINE VOL. 19, NO. 1 PART 2 : TUTORIALS–BLACKMAN*, janvier 2004.

[2] R Chandrasekhar. *How to Write a Thesis : A Working Guide*. The University of Western Australia, 24 janvier 2008.

[3] Piotr Dollar et al. Integral channel features. *BMVC.*, 2009.

[4] Jan Hosang, Rodrigo Benenson et Bernt Schiele. *How good are detection proposals, really ?* MPI Informatics Saarbrücken, Germany, 2014.

[5] S-C. S. Cheung et C. Kamath. *Robust techniques for background substraction in urban traffic video*. IST/SPIE's Symposium on Electronic Imaging, 3 novembre 2003.

[6] Jernej Mrovlje et Damir Vrančić. *Distance measuring based on stereoscopic pictures*. 9th International PhD Workshop on Systems and Control : Young Generation Viewpoint, octobre 2008.

[7] Markus Enzweiler et Dariu M. Gavrila. Monocular pedestrian detection : Survey and experiments. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, pages 2179–2195, 16 octobre 2008.

[8] J. V. Candy et E. F. Breitfeller. *Receiver Operating Characteristic (ROC) Curves : An Analysis Tool for Detection Performance*. Lawrence Livermore National Laboratory, 2013.

[9] Manaf A. Mahammed, Amera I. Melhum et Faris A. Kochery. Object distance measurement by stereo vision. *International Journal of Science and Applied Information Technology (IJSAIT), Vol.2 , No.2*, pages 05–08, mars 2013.

[10] F Schwenker et G Palm. Tree-structured support vector machines for multi-class pattern recognition. *International Workshop on Multiple Classifier Systems*, pages 409–417, 2001.

[11] Derek Bradley et Gerhard Roth. Adaptive thresholding using the integral image. *Journal of Graphics Tools 12.2*, pages 13–21, 2007.

[12] Ahmed Elgammal, David Harwood et Larry Davis. *Non-parametric Model for Background Subtraction*. University of Maryland, 2000.

[13] Rasmus Rothe, Matthieu Guillaumin et Luc Van Gool. *Non-maximum Suppression for Object Detection by Passing Messages between Windows*.

[14] Cardinael Sébastien et Malcourant Arthur. *Ensemble des vidéos du TFE*. https ://-drive.google.com/drive/folders/1AnfPvaH5tZ4kVhcfc5DEtY39c7HgHSUK ?usp=sharing, modifié le 10 juin 2018.

[15] Christophe Deleval et Maxime Lahy. *Multiple object tracking combining camera and radar.* TFE Ecole polytechnique de Louvain (EPL), Année académique 2016-2017.

[16] Piotr Dollar, Serge J Belongie et Pietro Perona. The fastest pedestrian detector in the west. *BMVC. Vol. 2. 3.*, 2010.

[17] D. Tome, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi et S. Tubaro. Deep convolutional neural networks for pedestrian detection. *Signal Processing : Image communication 47 (2016)*, pages 482–489, mars 2016.

[18] Joseph Redmon et Santosh Divvala. *You Only Look Once : Unified, Real-Time Object Detection.* University of Washington et Allen Institute for Artificial Intelligence.

[19] James Fogarty, Ryan S. Baker et Scott E. Hudson. Case studies in the use of roc curve analysis for sensor-based estimates in human computer interaction. *Human Computer Interaction Institute Carnegie Mellon University*, mai 2005.

[20] Jri Lee Yi-An Li , Meng-Hsiung Hung et Shih-Jou Huang. A fully-integrated 77-ghz fmcw radar transceiver in 65-nm cmos technology. *IEEE journal of solid-state circuits, VOL. 45, NO. 12*, pages 2746–2755, 12 décembre 2010.

[21] Pavlina Konstantinova, Alexander Udvarev et Tzvetan Semerdjiev. *A Study of a Target Tracking Algorithm Using Global Nearest Neighbor Approach.* International Conference on Computer Systems and Technologies - CompSysTech', 2003.

[22] https ://www.raspberrypi.org/. *Raspberry Pi*, Consulté le 10 mai 2018.

[23] Andrzej Wojtkiewicz Jacek Misiurewicz Marek Nałe¸cz Konrad Je¸drzejewski Krzysztof Kulpa. *Two-dimensional signal processing in FMCW radars.* Instytut Podstaw Elektroniki, Année académique 2017-2018.

[24] V. Legat. *Mathématiques et méthodes numériques.* Ecole Polytechnique de Louvain, Année académique 2017-2018.

[25] MathWorks. *Assign detections to tracks.* https ://nl.mathworks.com/help/vision/ref/assigndetectionstotracks.html, Consulté le 10 février 2018.

[26] MathWorks. *kalmanFilter.* https ://nl.mathworks.com/help/vision/ref/vision.kalmanfilter-class.html, Consulté le 10 février 2018.

[27] Mathworks. *Understanding Kalman Filters.* https ://nl.mathworks.com/videos/understanding-kalman-filters-part-3-optimal-state-estimator–1490710645421.html, Consulté le 10 février 2018.

[28] MathWorks. *Motion-Based multiple object tracking.* https ://nl.mathworks.com/help/vision/examples/motion-based-multiple-object-tracking.html, Consulté le 11 novembre 2017.

[29] MathWorks. *Polyfit.* https ://nl.mathworks.com/help/matlab/ref/polyfit.html, Consulté le 20 février 2018.

[30] Mathworks. *Single Camera Calibrator App.* https ://nl.mathworks.com/help/vision/ug/single-camera-calibrator-app.html, Consulté le 20 février 2018.

[31] MathWorks. *Blob analysis.* https ://nl.mathworks.com/help/vision/ref/vision.blobanalysis-system-object.html ?s_tid=doc_ta, Consulté le 23 avril 2018.

[32] MathWorks. *Label connected components*. https ://nl.mathworks.com/help/vision/ref/label.html, Consulté le 23 avril 2018.

[33] Catarina Moreira. *Learning To Rank Academic Experts*, 2011.

[34] James Munkres. Algorithms for the assignement and transportation problems. *Journal of the Society for Industrial and Applied Mathematics, Volume 5, Number 1*, pages 32–38, mars 1957.

[35] Douglas Reynolds. *Gaussian Mixture Models*. MIT Lincoln Laboratory, 2015.

[36] R. Ronsse. *Robotics - LMECA2732*. Ecole Polytechnique de Louvain, 2016-2017.

[37] St.Gallen. *Digital Radar - Specification Command-Description*. RFbeam, 21 avril 2016.

[38] L. Vandendorpe. *Télécommunications - LELEC1360*. Ecole Polytechnique de Louvain, 2015-2016.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**