Géolocalisation des objets détectés par une caméra monoculaire à bord d'un véhicule

par

Abderaouf TERMOUL

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE
M. Sc. A.

MONTRÉAL, LE 27 AVRIL 2023

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Maarouf Saad, directeur de mémoire
Département de génie électrique à l'École de technologie supérieure

M. Pier-Marc Comtois-Rivet, codirecteur
Département de robotique et intelligence artificielle l'Institut du véhicule innovant, St-Jérome

M. Vahé Nerguizian, président du jury
Département de génie électrique à l'École de technologie supérieure

M. Christian Belleau, membre du jury
Département de génie mécanique à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 24 AVRIL 2023

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens à exprimer ma gratitude à toutes les personnes qui ont contribué à la réussite de mon mémoire.

Je tiens tout d'abord à exprimer ma sincère gratitude envers mon directeur de recherche, le Professeur Maarouf Saad, pour son suivi attentif, ses conseils éclairés, ses propositions constructives et sa grande bienveillance tout au long de ce projet. Son soutien indéfectible a été d'une aide inestimable pour mener à bien ce travail.

Je remercie également mon co-directeur, Pier-Marc Comtois-Rivet , pour tout le savoir-faire qu'il m'a transmis tout au long du projet, ainsi que pour sa patience à toute épreuve.

Un grand merci à Anthony Courchesne et Yassine Kali pour leur soutien infaillible, leurs idées et leurs informations précieuses, ainsi que pour leur disponibilité chaque fois que j'en avais besoin.

Mes remerciements s'adressent, également, aux membres du Jury d'avoir accepté d'évaluer mon mémoire.

Je n'oublie pas de remercier mes parents pour leur soutien financier et moral tout au long de mon cursus. Ce travail n'aurait pas été possible sans leur aide et leur encouragement constants. Je remercie également ma femme pour son soutien moral sans faille.

Enfin, je tiens à exprimer ma gratitude à tous mes amis du GRÉPCI, en particulier Saif Sinan, David Bedolla Martinez et Mahoumd Abadlah pour leur aide précieuse et leur soutien indéfectible. Merci à tous pour votre soutien et votre contribution à la réalisation de ce projet.

Géolocalisation des objets détectés par une caméra monoculaire à bord d'un véhicule

Abderaouf TERMOUL

RÉSUMÉ

Ce travail se fait dans le cadre d'un projet de recherche financé par le Ministère des transports du Québec (MTQ) sur l'acquisition de données pour véhicules autonomes. Particulièrement, l'aspect de géolocalisation d'objets entourant le véhicule sera traité. Ainsi, le but est de résoudre la problématique de la géolocalisation des objets détectés par une caméra monoculaire montée sur un véhicule pour automatiser la localisation GPS des anomalies routières telles que les lampadaires défectueux et les nids de poule. L'objectif principal est de cartographier les routes de la province du Québec en trouvant les positions GPS des objets d'intérêt pour créer une base de données qui pourrait être exploitée pour diverses applications telles que l'automatisation du processus de repérage des anomalies routières ou encore pour fournir des informations aux véhicules autonomes sur la position des anomalies telles que les nids de poule ou les feux de circulation.

Les méthodes de géolocalisation basées sur les lignes d'orientation ont été privilégiées en raison de leur simplicité et de leur faible demande en termes de temps de calcul. Les algorithmes de détection, de suivi et de géolocalisation ont été développés pour générer les lignes d'orientation et estimer les positions GPS des objets détectés. Les résultats expérimentaux ont montré, pour l'échantillon étudié, une précision de 88% dans un rayon de 20 mètres pour la géolocalisation des lampadaires. Des améliorations sont possibles pour renforcer la performance de l'algorithme, notamment en calibrant la pose de la caméra et en améliorant la configuration GPS.

Mots-clés: Géolocalisation, lignes d'orientation, caméra monoculaire

Geolocation of objects detected by a monocular camera on board a vehicle

Abderaouf TERMOUL

ABSTRACT

This work is being carried out as part of a research project funded by the Quebec Ministry of Transport on data acquisition for autonomous vehicles. Specifically, the aspect of geolocating objects surrounding the vehicle will be dealt with. So the aim of this project is to address the challenge of geolocating objects detected by a monocular camera mounted on a vehicle, with the goal of automating the GPS-based localization of road anomalies such as faulty streetlights and potholes. The main objective is to map Quebec by identifying the GPS positions of objects of interest, in order to create a database that could be used for various applications, such as automating the process of identifying road anomalies or providing information to autonomous vehicles about the location of anomalies such as potholes or traffic lights.

The localization methods based on orientation lines were chosen due to their simplicity and their low demand in terms of computation time. Detection, tracking, and geolocation algorithms were developed to generate orientation lines and estimate the GPS positions of detected objects. Experimental results showed 88% accuracy within a 20-meter radius for geolocating streetlights. Improvements are possible to strengthen the algorithm's performance, particularly by calibrating the camera's pose and improving the GPS configuration.

Keywords: Geolocation, Monocular camera, Line-based geolocation

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

GPS	Global Positioning System
ENU	East North Up
NED	North East Down
EPSG	European Petroleum Survey Group
ORB	Oriented FAST and Rotated BRIEF
SIFT	Scale-Invariant Feature Transform
SURF	Speeded-Up Robust Features
CNN	Convolutional Neural Network
ROS	Robot Operating System
FOV	Field of View
GSV	Ground Sampled Distance
fps	Frames per Second
mAP	mean Average Precision
YOLO	You Only Look Once
SPP	Spatial Pyramid Pooling
CSP	Cross Stage Partial Network
FLOPS	Floating Point Operations per Second
CUDA	Compute Unified Device Architecture
CIoU	Complete Intersection over Union
ORB	Oriented FAST and Rotated BRIEF
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
SLAM	Simultaneous Localization And Mapping
NMS	Non-Maximum Suppression

LISTE DES SYMBOLES ET UNITÉS DE MESURE

m	mètre
FPS	Frame per seconde

CHAPITRE 1

INTRODUCTION

Les véhicules autonomes sont une application émergente qui peut apporter de nombreuses améliorations à notre vie quotidienne, notamment l'amélioration de la fluidité de la circulation en utilisant des technologies de communication entre véhicules, la réduction de la fatigue et de la charge mentale pour les conducteurs, l'amélioration de l'accessibilité pour les personnes ayant des difficultés à conduire et la prévention des accidents de voiture dus à l'erreur humaine puisqu'avec cette technologie la distraction n'est plus un facteur à prendre en compte. En outre, les véhicules autonomes peuvent grandement améliorer l'efficacité des transports publics et la gestion du trafic grâce à la communication entre eux qui conduit à l'optimisation des itinéraires à emprunter et au respect des distances de sécurité. Ce sujet représente de grands défis pour la communauté scientifique ces dernières années en raison de la complexité de l'application sur différents axes : perception de la scène, géolocalisation, prise de décision, navigation et contrôle. Néanmoins, plusieurs avancées dans ce domaine ont été réalisées et nos voitures deviennent de plus en plus autonomes.

Cependant, l'arrivée des véhicules autonomes au Québec représente un défi pour diverses raisons, notamment les conditions météorologiques, l'état des routes, les données disponibles pour l'exploitation, etc. Dans cette optique, le Ministère des Transports du Québec (MTQ) a lancé un projet de recherche. Ce projet a été confié à l'ÉTS et à l'Institut du Véhicule Innovant (IVI) qui vise à cartographier les routes du Québec afin de le préparer à l'intégration des véhicules autonomes sur ses routes.

Le but principal de ce projet est de cartographier les routes du Québec, y compris les panneaux de signalisation, les lampadaires et autres éléments, ainsi que de localiser les anomalies routières telles que les nids de poule, les lampadaires défectueux et les glissières de sécurité cassées. Actuellement, ces anomalies sont localisées manuellement par des patrouilleurs circulant sur le réseau routier. Cette approche rend difficile le maintien de l'état des structures en temps réel,

d'autant plus que les patrouilleurs ont de nombreuses autres tâches à accomplir. Elle présente également un risque pour les agents qui doivent s'arrêter et localiser les anomalies visuellement.

Le but de ce projet est d'automatiser la détection et la localisation des objets présents sur la route à partir des images capturées par une caméra installée sur une voiture équipée d'un ordinateur. Cette automatisation vise à fournir une carte complète contenant des données exploitables, telles que les différents éléments de la route. Pour atteindre cet objectif, il est nécessaire de développer un algorithme capable de détecter les objets d'intérêt dans les images capturées par la caméra et de déterminer leur position géographique en coordonnées GPS en fonction de leur position dans les images et de la position du véhicule au moment de la capture. Ces données pourront être exploitées par les agents du ministère des Transports afin d'intervenir pour réparer les anomalies routières et améliorer l'état des routes. De plus, l'implémentation de ces données dans les cartes des véhicules autonomes permettra de repérer plus facilement toutes les signalisations et les anomalies, telles que les nids-de-poule, et de les éviter. Cela améliorera la sécurité et le confort des usagers de la route, ainsi que la durée de vie des véhicules en réduisant les risques d'endommagement. En somme, ce projet vise à améliorer la qualité des infrastructures routières et à faciliter les déplacements de la population.

Dans ce contexte, de nombreuses recherches ont été menées pour développer des méthodes d'automatisation de la détection et de la localisation des objets sur les routes. Généralement, la localisation d'un objet nécessite une combinaison de données provenant de différents types de capteurs, tels que LIDAR et Radar, qui peuvent fournir des informations de distance et d'angle avec une grande précision entre la voiture et l'objet détecté (Gao *et al.*, 2018). Les caméras stéréoscopiques, quant à elles, peuvent donner une estimation de la profondeur, aidant à géolocaliser plus facilement l'objet détecté en termes d'algorithme. Cependant, ces technologies peuvent être coûteuses ou nécessiter une grande capacité de calcul, comme c'est le cas pour les caméras stéréoscopiques (Kim *et al.*, 2015).

L'objectif de ce mémoire est de mettre en place un algorithme de géolocalisation d'objets repérés par un véhicule équipé d'une caméra monoculaire, d'un ordinateur et d'un capteur GPS.

Cependant, étant donné que la capacité de calcul est limitée, cela représente un défi majeur dans ce projet, car nous visons une application en temps réel. Pour atteindre cet objectif, nous avons utilisé des techniques d'intelligence artificielle pour identifier et détecter les objets d'intérêt sur les images. Nous avons ensuite testé et validé des algorithmes basés sur la géométrie dans l'espace pour déterminer la position GPS des objets repérés. L'application de ces méthodes a permis de développer un système de géolocalisation rapide et efficace pour les objets détectés.

Ce rapport décrira les étapes de développement, les résultats obtenus et les perspectives d'amélioration de ce système. Ce dernier est divisé en 4 parties :

- Le chapitre 2 porte sur l'état de l'art de la géolocalisation des objets détectés par un véhicule ainsi que les techniques de détection et de suivi des objets.
- Le chapitre 3 décrit l'algorithme utilisé pour détecter les objets (YOLO) sur les images, l'architecture de ce modèle en détaillant le fonctionnement de chaque composant ainsi que le suiveur (tracker) utilisé.
- Le chapitre 4 explique l'approche utilisée pour estimer la position GPS des objets, en configurant le capteur GPS, en calibrant la caméra et en effectuant des calculs géométriques.
- Le chapitre 5 présente les résultats obtenus et les pistes d'amélioration pour optimiser le système.

CHAPITRE 2

REVUE DE LITTÉRATURE

2.1 Introduction

Dans le cadre de ce projet de grand intérêt ces dernières années, plusieurs chercheurs ont contribué à améliorer le procédé de géolocalisation d'objets détectés à l'aide d'une caméra à bord d'un véhicule en mouvement. Bien que la grande majorité des travaux et avancements sur le sujet ont été faits sur les véhicules aérien sans pilote, ces apports ont fait évoluer l'état de l'art, notamment en ce qui concerne les systèmes basés sur la vision et les algorithmes de l'intelligence artificielle ainsi que les algorithmes de la reconstruction projective 3D. Parallèlement, les avancements du côté matériel dans ce domaine ont permis de réduire le coût de ce processus de géolocalisation, plus précisément, le système de positionnement par satellites (GPS) et les caméras.

Dans ce chapitre, nous allons voir les techniques les plus utilisées dans le même contexte de ce projet qui sont divisées sur deux axes de recherche :

- Détermination de la position GPS des objets.
- Détection et suivi des objets avec une caméra monoculaire.

2.2 Géolocalisation d'un objet détecté par un véhicule

La géolocalisation d'un objet est le processus de détermination de ses coordonnées GPS. Le GPS est le système le plus utilisé dans ces applications car il fournit une description unique de la position qui peut être réutilisée et convertie dans un autre système de coordonnées local, tel que le référentiel tangentiel Est-Nord-Haut (ENU). Le GPS permet également d'uniformiser la représentation des mesures de la position dans un système de coordonnées global, tel que le GNSS (Géolocalisation et Navigation par un Système de satellites). (Gao *et al.*, 2021).

En se basant sur les données acquises à partir de capteurs GPS qui sont principalement la position absolue et l'orientation du véhicule dans le référentiel du monde réel, plusieurs systèmes ont été conçus afin d'obtenir la position GPS d'un objet détecté par ce dernier.

Dans plusieurs articles scientifiques, une fusion de données entre plusieurs sources de données telles que LIDAR, caméra stéréoscopique ou caméra RGBD a été utilisée, car elle permet d'exploiter plusieurs informations en même temps pour obtenir une position précise ce qui rend la fusion de données de perception prometteuse malgré les défis qu'elle présente.

Dans Kim et al. (2017), les auteurs avaient pour objectif de localiser le véhicule. Ils proposent une fusion des données venant d'une centrale inertielle (IMU), LIDAR et l'odométrie des roues. Cette fusion a été faite en utilisant un filtre à particules. D'auteurs chercheurs utilisent un filtre de Kalman sous ses différentes formes. Dans Deilamsalehy & Havens (2016), le filtre de Kalman étendu a été utilisé afin de fusionner les données venant de l'IMU, la caméra et LIDAR afin d'estimer la position et l'orientation du véhicule. Dans Shetty & Gao (2017), les auteurs utilisent un filtre de Kalman sans parfum (UKF) pour fusionner les données du LIDAR et du GPS, en proposant une nouvelle méthode pour modéliser de manière efficace la covariance de l'erreur de position en se basant sur les nuages de points LIDAR. Toutes les mesures et les matrices de covariance d'erreur trouvées sont entrées dans un filtre de Kalman sans parfum (UKF) afin d'obtenir une estimation de la position d'un véhicule aérien autonome.

L'idée de fusionner les données du LIDAR avec des images a également été explorée par Krylov & Dahyot (2018). Dans cet article, les auteurs cherchent à géolocaliser les feux de circulation sur la route en combinant les informations provenant de différentes sources. Ceux-ci incluent les résultats de la segmentation d'objets effectuée par un réseau Faster CNN, l'estimation de la profondeur monoculaire à partir d'images de Google Street View, les résultats de la triangulation et les données du LIDAR. Le pipeline de l'algorithme est présenté dans la Figure 2.1.

Dans cet article, le processus était divisé en trois étapes. D'abord, ils utilisent un Faster CNN pour faire une segmentation sémantique afin de localiser l'objet sur l'image. Les résultats de

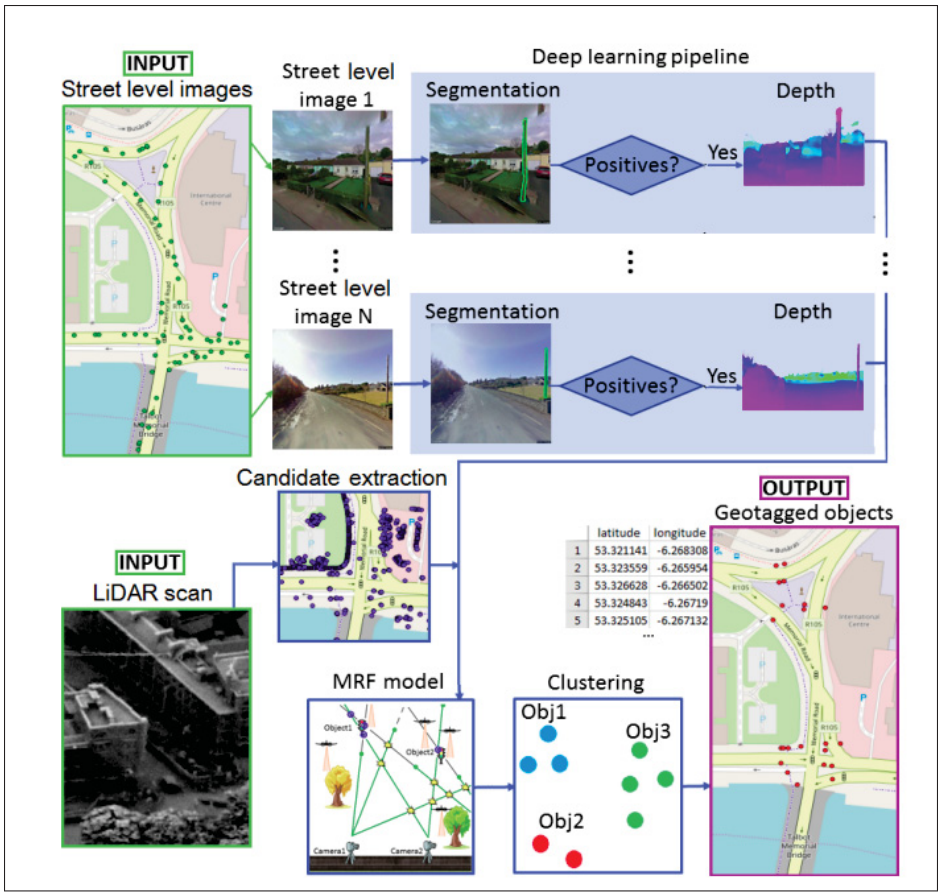

FIGURE 2.1 Algorithme proposé dans Krylov & Dahyot (2018)

segmentation passent à un autre réseau neuronal qui sert à estimer la profondeur monoculaire (Godard *et al.*, 2017).

Parallèlement, l'extraction des points candidats du LIDAR a été traitée comme un problème de correspondance du modèle qui est un problème connu en traitement d'image comme le processus de recherche d'un motif ou d'une petite partie d'une image sur une autre image. Dans leur cas, ils ont utilisé un modèle de feux de circulation.

Finalement, les informations recueillies sont transmises au processus de fusion de données basé sur la théorie des champs aléatoires de Markov (MRF) (Li, 2009). Ce processus prend en compte les points d'intersection entre les lignes sortant des caméras en direction de l'objet.

Ces lignes sont générées à partir des résultats de la première étape, notamment les distances entre les points d'intersection obtenues par triangulation, les distances estimées par le réseau d'estimation de la profondeur monoculaire, et les points candidats les plus proches parmi les données du LIDAR. La figure 2.2 montre un exemple de trois objets observés à partir de trois positions de caméra différentes. Les estimations de la profondeur monoculaire sont représentées en vert et les correspondances LIDAR sont indiquées par une icône de drone. Tous les objets se trouvent à l'intersection des rayons de vue projetés en direction des feux de signalisation, qui ont été identifiés et délimités dans des images de rue segmentées. Ensuite, en considérant la distance entre chaque point d'intersection des lignes et le point candidat LIDAR le plus proche, ainsi que la distance entre la caméra et l'estimation de la profondeur monoculaire, une décision sera prise quant à la pertinence de chaque point détecté.

Dans Kuutti et al. (2018) et Campbell et al. (2018), les auteurs ont fait une revue de la littérature sur les différents capteurs intégrés dans les véhicules autonomes dans le but de localiser le véhicule et les objets qui l'entourent. Ils mentionnent que malgré la précision offerte par le LIDAR, sa consommation énergétique importante et le coût de sa mise en œuvre très élevé représentent un inconvénient, d'autre part, pour une caméra c'est plutôt l'inverse, la consommation énergétique est beaucoup moins importante et le coût de l'implémentation est faible, mais sa précision dans de telles applications est moins bonne. Cette contrainte relève le défi de profiter des avantages des caméras monoculaires, en utilisant les avancements dans le domaine de la vision par ordinateur pour optimiser la précision. Il reste que l'utilisation de la caméra monoculaire seulement représente plusieurs difficultés, notamment l'ambiguïté du passage de représentation 3D vers une représentation 2D ainsi que l'incertitude de la pose de la caméra (Gao *et al.*, 2021).

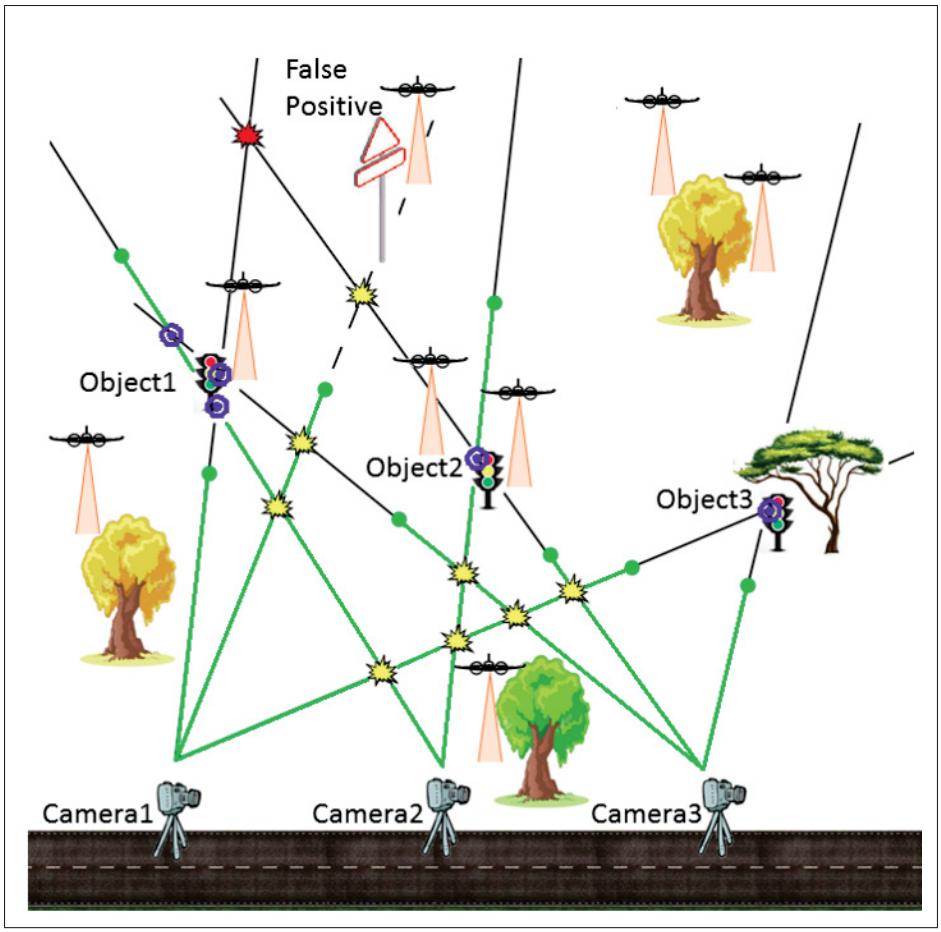

FIGURE 2.2 Exemple de problème de géolocalisation d'objets basé sur les intersections de rayons de vue au niveau de la rue (Krylov & Dahyot, 2018)

2.3 Géolocalisation d'un objet détecté par une caméra monoculaire

2.3.1 ORB-SLAM

Mur-Artal et al. (2015) ont développé une approche appelée cartographie et localisation simultanées-ORB (ORB-SLAM) qui est devenue une méthode très utilisée dans les contextes des systèmes de la vision monoculaire. Cette dernière se base sur l'extraction des caractéristiques d'une scène avec la méthode Oriented FAST and Rotated BRIEF (ORB) introduite dans Rublee et al. (2011). L'ORB a été choisie car elle est plus efficace en termes de temps d'exécution

comparé à d'autres extracteurs de caractéristiques comme Scale Invariant Feature Transform (SIFT) (Lowe, 2004) ou Speeded-up robust features (SURF) (Bay *et al.*, 2008). La Figure 2.3 montre un exemple de résultat d'utilisation de ces extracteurs.

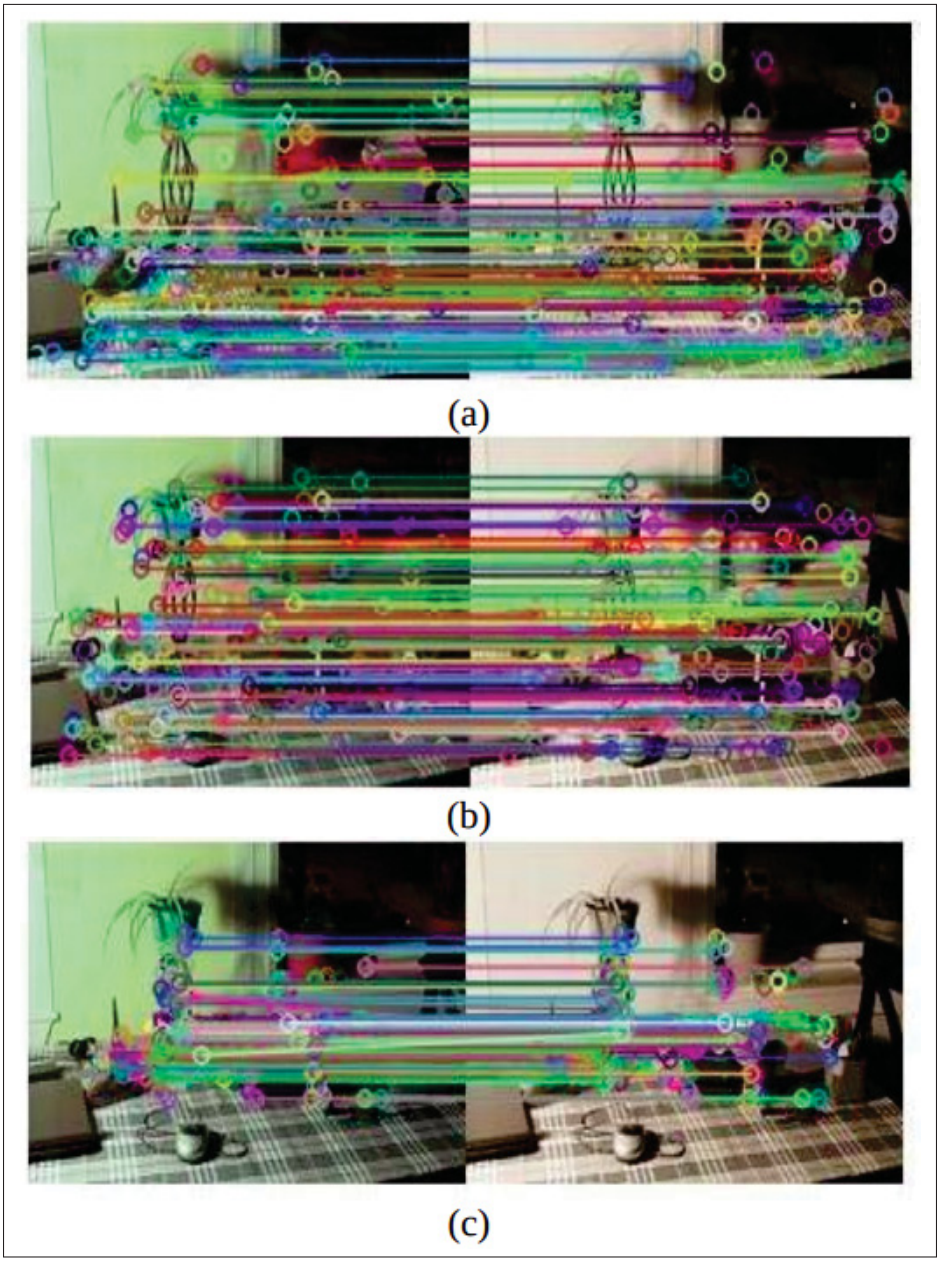

FIGURE 2.3 Résultats d'extraction des caractéristiques et de correspondance : (a) SIFT, (b) SURF, (c) ORB (Karami *et al.*, 2017)

Les caractéristiques extraites sont utilisées pour les trois tâches principales de l'algorithme : suivi, cartographie locale et fermeture de boucle (c'est à dire que le véhicule est retourné à sa position initiale).

Dans Zhao et al. (2020), les auteurs utilisent une approche basée sur ORB-SLAM pour géolocaliser un objet mobile présent sur vidéo qui a été prise par un satellite. En effet, après avoir analysé les résultats expérimentaux de SIFT et ORB présentés dans le même article, ils trouvent que SIFT trouve plus de correspondances qu'ORB en donnant le même nombre de points clés aux deux algorithmes.

SIFT repose sur la différence de gaussiennes qui consiste à faire la soustraction entre une image floue et la même image, mais en changeant le degré de flou. Cette dernière permet d'extraire les coins avec une bonne précision. D'autre part la méthode ORB est plus rapide en exécution car elle repose sur les caractéristiques élémentaires indépendantes robustes binaires (BRIEF). Cette dernière est basée sur un nombre limité de tests de différence d'intensité de pixel afin de représenter une partie de l'image qui contient un point clé sous forme de vecteur binaire, ce qui permettra par la suite de retrouver rapidement les correspondances (Calonder *et al.*, 2010).

Cette combinaison entre la différence de gaussiennes et BRIEF permet de profiter de la précision offerte par SIFT et l'exécution rapide d'ORB. Ensuite, pour chaque image, la position du satellite est obtenue à travers les mesures du capteur GPS qui présente des imprécisions. Pour améliorer l'estimation de la position du satellite, les auteurs ont combiné les concepts de l'odométrie visuelle avec SLAM de telle sorte que : pour l'estimation de mouvement par image, ils utilisent l'homographie entre les images, et pour enlever l'ambiguïté d'échelle causée par l'odométrie visuelle, ils combinent le résultat d'homographie avec les données GPS du satellite pour déterminer un facteur d'échelle (voir Figure 2.4). Finalement, les coordonnées en 3D de l'objet en question sont déterminées par triangulation.

D'autres chercheurs mentionnent que malgré les performances du SLAM dans le contexte de la géolocalisation, la méthode demande beaucoup de ressources computationnelles et de mémoire (Aldegheri *et al.*, 2019). Aussi, la méthode présente des difficultés lors de l'initialisation avec

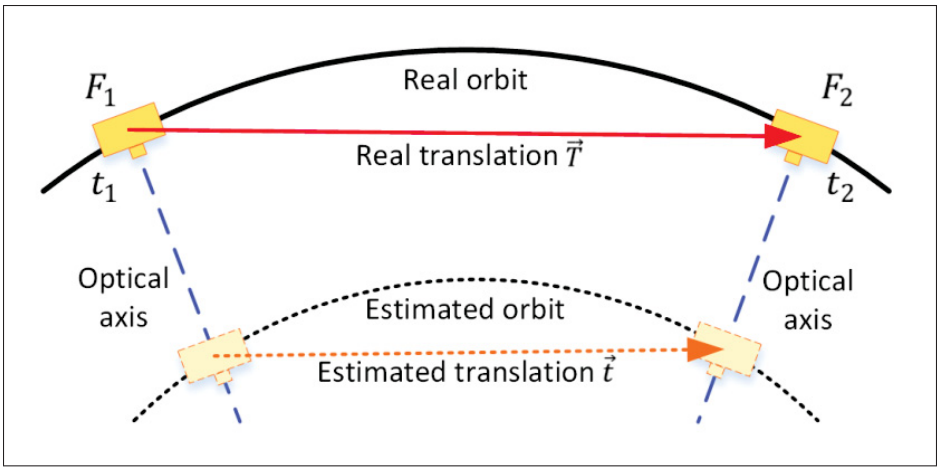

FIGURE 2.4 La différence d'estimation de la translation entre l'odométrie visuelle et les données GPS (Zhao *et al.*, 2020)

une séquence d'images qui n'est pas riche en texture ou lorsque la caméra tourne avec un grand angle entre les images (Bustos *et al.*, 2019).

2.3.2 Structure From Motion

D'autres chercheurs utilisent l'algorithme de Structure From Motion (SFM) introduit dans Schonberger & Frahm (2016) qui sert à la reconstitution des points 3D à partir d'une séquence d'images 2D. L'algorithme est défini comme suit : la première étape est d'extraire les caractéristiques avec un extracteur comme SIFT ou SURF sur une séquence d'images. Ensuite, la recherche des correspondances entre les points clés trouvés sur les différentes images et finalement, l'estimation de la pose de la caméra en utilisant les concepts de la géométrie épipolaire. Une fois la pose de la caméra est estimée, les coordonnées des points en 3D seront déterminées par triangulation. L'amélioration de l'estimation est faite après par comparaison de la reprojection de chaque point triangulé sur le plan d'image 2D avec les coordonnées des points 2D utilisés pour la triangulation. D'autres chercheurs utilisent l'algorithme de Bundle adjustment pour minimiser cette erreur (Wu, 2013).

Dans Sun et al. (2018), les auteurs mentionnent les difficultés trouvées pour appliquer la SFM pour estimer la position GPS en utilisant des images ordinaires et un GPS de téléphone mobile.

La SFM offre un calcul mathématique solide pour estimer la pose de la caméra pour chaque image, sauf que le calcul des points clés et la recherche des correspondances pour chaque paire d'images ainsi que les tests exhaustifs sur tous les points pour estimer la pose et éliminer les valeurs aberrantes impliquent une demande importante des ressources de calcul, ainsi, le temps d'exécution plus lent pour une application en temps réel.

L'approche proposée dans Gené-Mola et al. (2020a) vise à trouver les coordonnées 3D des fruits détectés par un réseau neuronal de segmentation Mask R-CNN introduit dans He et al. (2017) en utilisant la SFM, où l'extracteur de caractéristiques est cette fois le réseau de neurones. Cela permet de sélectionner le type d'objet qu'on veut localiser dans la scène et de réduire les calculs, car après l'application du masque sur les images, il ne restera que les objets d'intérêt (voir Figure 2.5) , ensuite pour éliminer les faux positifs, ils utilisent l'algorithme Machine à vecteurs de support (SVM). Les auteurs ont trouvé que cette méthode améliore la précision de localisation 3D, mais le temps d'exécution reste important.

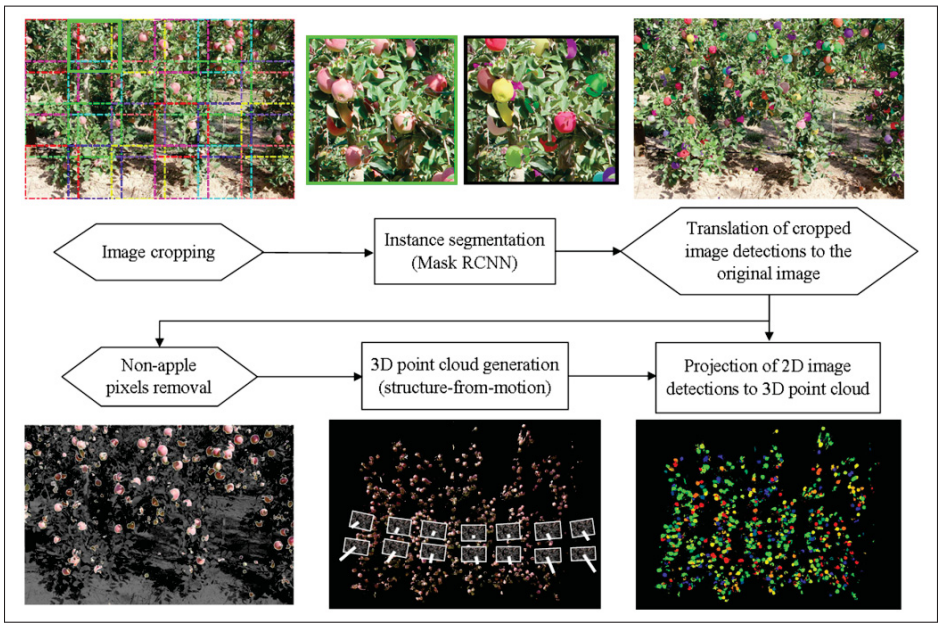

FIGURE 2.5 L'application du MASK R-CNN avec Structure From Motion (Gené-Mola *et al.*, 2020a)

2.3.3 Line Of Bearing

Dans Zhang et al. (2018), les auteurs visent à détecter et géolocaliser les poteaux sur les images tirées de Google Street View (GSV). D'abord, ils ont créé une base de données avec les images prises de GSV dans une région géographique définie, après, ils ont annoté les poteaux (3500 poteaux) qui représentent l'objet d'intérêt dans leur étude. Ces annotations ont été utilisées pour l'entraînement de CNN utilisé.

Après avoir entraîné le réseau neuronal, ce dernier était utilisé pour localiser les poteaux sur une séquence d'images. Sa sortie est le cadrage (bounding box) de la zone où l'objet se trouve, une fois l'objet est détecté et localisé, ils génèrent la ligne d'orientation (line of bearing) correspondante à chaque détection, en connaissant le champ d'ouverture de la caméra utilisée (Field Of View) et l'azimut relatif entre l'image et le point de vue où cette image était prise (Figure 2.6) et finalement les coordonnées GPS peuvent être calculées par triangulation entre chaque deux points de vues. Ensuite le point qui correspond à l'objet est estimé par regroupement (clustring).

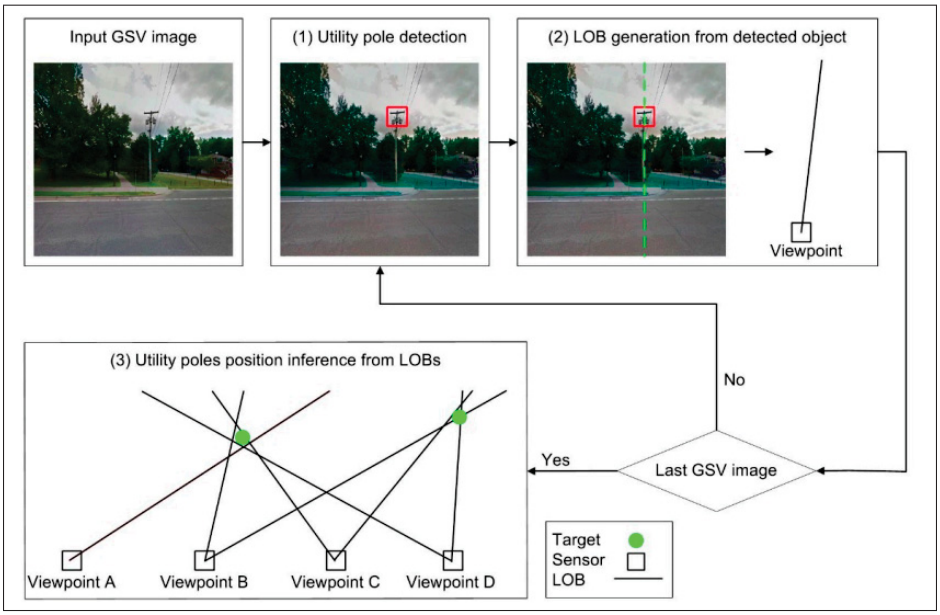

FIGURE 2.6 Exemple des lignes d'orientation (Zhang *et al.*, 2018b)

Le problème qui se présente dans cette approche est les faux positifs, qui sont produits soit par les imprécisions de détection, soit par les erreurs des positions GPS pour chaque point de vue. Dans le même article, les auteurs ont proposé une solution appelée, brute-force-based lines of bearing, qui consiste à éliminer certains points qui ne respectent pas la contrainte montrée sur la Figure 2.7.

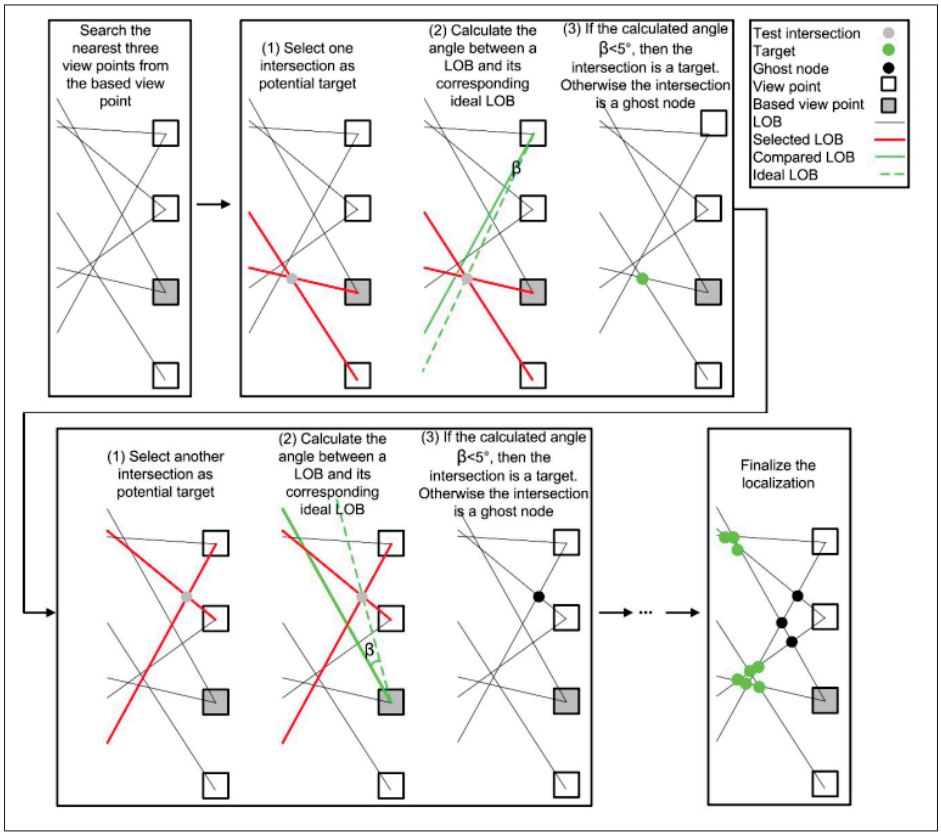

FIGURE 2.7 Exemple de brute-force-based lines of bearing (Zhang *et al.*, 2018b)

Cette méthode ne nécessite pas de grandes capacités de calcul, elle est simple à mettre en œuvre et sa précision donnée est raisonnable avec toutes les hypothèses qui ont été faites. Dans le cas étudié, les poses de la caméra ont été prises de GSV qui présente des erreurs. D'autre part, la méthode utilisée pour éliminer les faux positifs repose sur des seuils fixes et donc ne tient pas compte des rotations qu'on peut avoir entre les différentes scènes.

Comme nous avons vu à travers les articles cités précédemment, la détection des objets est la première étape avant d'entamer la géolocalisation. Dans la section prochaine, nous allons voir les différents outils et approches utilisés par les chercheurs afin d'assurer une bonne détection des objets présents dans l'image suivie par une revue sur les techniques utilisées pour suivre le même objet dans une séquence d'images.

2.4 Détection des objets par une caméra monoculaire

2.4.1 Techniques traditionnelles de détection

Avant le développement d'apprentissage profond, les techniques de détection utilisées ont été basées sur le calcul de descripteurs locaux comme le cas de SIFT qui vise au début de trouver des points clés par différence de Gaussienne en floutant l'image par un flou Gaussien avec différentes intensités. Ensuite ces images seront soustraites les unes des autres afin de déterminer les points extrêmes, cette opération sera reprise en changeant le facteur d'échelle de l'image afin de trouver les points qui sont invariants aux changements d'échelle. Après, chaque point clés aura un descripteur qui est le gradient calculé sur les pixels voisins du point clé (Lowe, 2004). Ces points permettent de retrouver des associations entre les images.

Une autre approche appelée Histogramme des Gradients Orientés (HOG) qui est proposée dans (Dalal & Triggs, 2005) qui utilise le même concept de calcul des descripteurs, mais cette fois-ci avec la magnitude et l'angle du gradient pour calculer les caractéristiques. En effet, pour chaque région de l'image, l'histogramme est calculé en utilisant l'amplitude ainsi que les orientations du gradient. Cette méthode était comparée avec un algorithme basé sur l'apprentissage profond dans le cadre de détection de poisson présenté dans Sung et al. (2017). Les auteurs ont trouvé que le réseau de neurones utilisé est meilleur que le HOG tant en précision qu'en temps d'exécution.

Dans Minaeian et al. (2018) les auteurs ont utilisé la localisation des points clés de l'arrière plan d'une scène pour l'estimation du mouvement de la caméra placée sur un drone, ensuite, ils ont fait la segmentation du premier plan de l'image afin de détecter les objets en mouvement. Leur

approche donne une bonne précision de détection avec un temps d'exécution raisonnable, sauf qu'elle ne permet pas de détecter tous les types d'objets ou de sélectionner un type d'objet à détecter (Zhao *et al.*, 2019).

2.4.2 Techniques de détection basées sur l'apprentissage profond

Certains chercheurs divisent les méthodes de détection d'objets basées sur l'apprentissage profond en deux groupes, les méthodes en deux étapes où on génère les régions d'intérêt et puis on les classe et les méthodes en une étape où on n'a pas besoin de générer des régions d'intérêt (Du *et al.*, 2020).

L'algorithme de régions avec des réseaux de neurones convolutionnels R-CNN présenté dans Girshick et al. (2014), cherche au début à sélectionner 2000 régions d'intérêt par un algorithme de recherche sélective présenté dans Uijlings et al. (2013). Ensuite ces régions proposées seront l'entrée de réseaux de neurones convolutionnels qui extraient les caractéristiques de chaque région et leur attribuent une classe. Cette segmentation permet de cerner la zone délimitante de chaque objet, ensuite, ils appliquent l'algorithme appelé suppression non maximale pour identifier les meilleures prédictions d'objets candidats (voir Figure 2.8 et Figure 2.9).

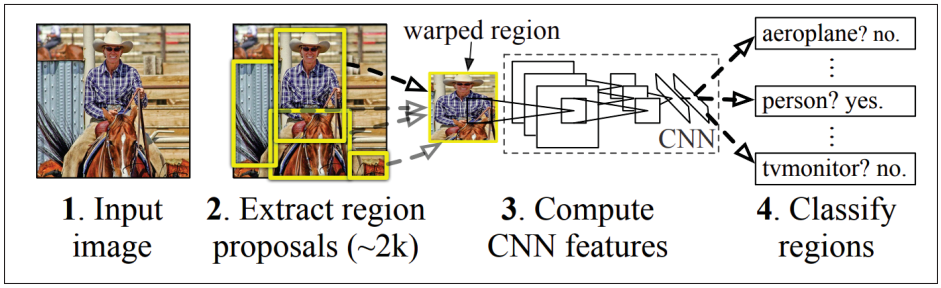

FIGURE 2.8 Pipeline de R-CNN (Girshick *et al.*, 2014)

Cet algorithme est lent et coûteux en termes de temps de calcul ce qui a poussé le même auteur de proposer une architecture différente sous le nom Fast R-CNN présentée dans Girshick (2015). Le Fast R-CNN permet de réduire considérablement le temps d'exécution. La différence principale entre la nouvelle architecture et celle d'avant est que cette fois-ci l'entrée du CNN

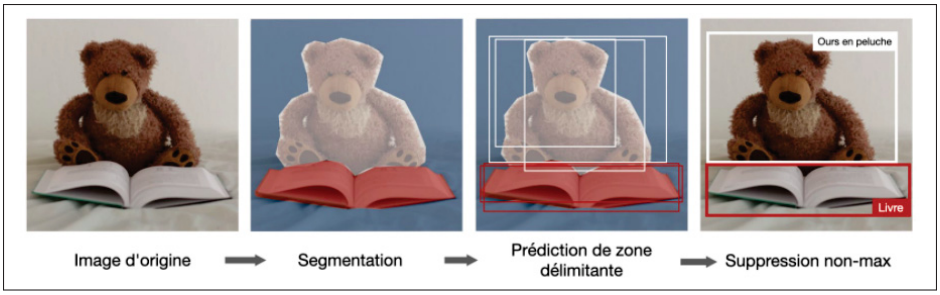

FIGURE 2.9 Exemple de R-CNN (Afshine Amidi, 2021)

ne sera pas 2000 régions, mais ça sera l'image elle-même. Ensuite, le CNN génère une carte des caractéristiques pour toute l'image, puis à partir de cette carte les régions d'intérêt seront identifiées. Dans Ren et al. (2015), les auteurs ont proposé une architecture un peu différente appelée Faster R-CNN. Ils sont arrivés à réduire encore le temps d'exécution en remplaçant l'algorithme de recherche sélective des régions d'intérêts par un réseau de neurones qui a été entraîné pour sélectionner les régions d'intérêt à partir des caractéristiques trouvées par le CNN. Cette technique a accéléré dix fois le processus de détection comparé au Fast R-CNN.

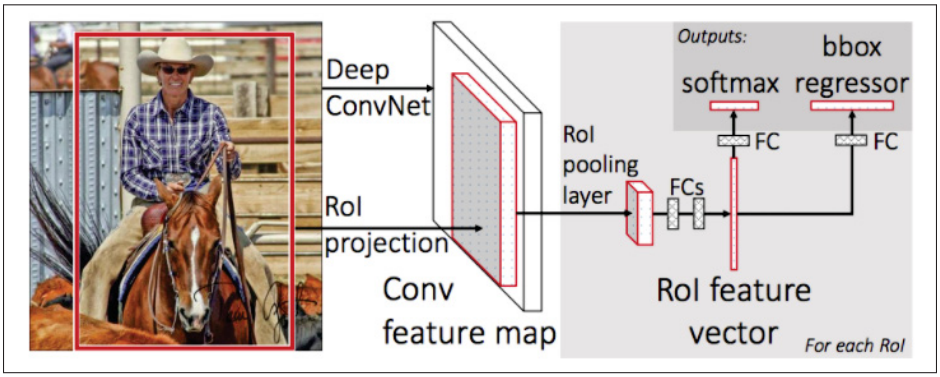

FIGURE 2.10 Pipeline de Fast R-CNN (Girshick, 2015)

Dans Redmon et al. (2016), les auteurs proposent une nouvelle méthode de détection appelée You Only Look Once (YOLO) qui a contribué à l'avancement des applications en temps réel à cause de son exécution rapide et qui est devenue l'algorithme typique de la méthode en une étape. L'algorithme fonctionne en divisant l'image sur des régions, chacune est responsable de la détection et de la localisation de l'objet présent dans cette région. Chaque région retourne

une prédiction sur le cadrage qui entoure l'objet (bounding box), ainsi que la classe de l'objet et la probabilité que l'objet soit présent dans cette partie de l'image. Ce processus cause de nombreuses prédictions en double en raison de plusieurs régions qui prédisent le même objet avec différents cadrages proposés, et dans ce cas la dernière étape de l'algorithme est d'utiliser la suppression non maximale pour trouver la meilleure prédiction.

Il y a eu plusieurs versions de cet algorithme. YOLOv2 présenté dans Redmon & Farhadi (2017) vise à améliorer le processus de détection et résoudre quelques problèmes de YOLO, car à la base YOLO peut prédire la présence d'un seul objet dans chaque région. Donc si la région contient plusieurs objets, ils ne seront pas détectés, ce qui a été résolu dans YOLOv2 en permettant la prédiction de plusieurs objets à partir d'une seule région.

Redmon & Farhadi (2018) présente une autre version, YOLOv3, qui fait la prédiction sur 3 facteurs d'échelles différents, et son architecture neuronale étant plus large que YOLOv2, ça lui permet d'être plus efficace en termes de détection des petits objets présents sur l'image.

Bochkovskiy et al. (2020) présente la version YOLOv4, où plusieurs améliorations ont été apportées à l'ancienne version, en citant l'utilisation d'une nouvelle technique d'augmentation des données appelée Mosaïque et aussi l'utilisation des DropBlock qui est une technique pour forcer le réseau à apprendre même quand une petite partie de l'image est cachée par exemple. Cela permet de réduire le phénomène de surapprentissage. Cette version est plus rapide et plus précise que YOLOv3, (voir Figure 2.11). YOLOv5 (Jocher *et al.*, 2021) est la version open source de YOLOv4 qui utilise la même architecture, mais elle est construite avec le framework pytorch (Paszke *et al.*, 2017).

Dans YOLOX (Ge *et al.*, 2021), les auteurs sont arrivés à dépasser la précision de YOLOv5 (voir Figure 2.13). Dans cette architecture les auteurs ont utilisé un détecteur sans ancre après avoir montré son efficacité dans plusieurs recherches, ce qui a réduit le nombre de paramètres à régler par rapport aux architectures à base d'ancre (anchor-based) tel que YOLOv3 et YOLOv4. Dans cette version, le concept de la tête découplée avec YOLO a été introduit où les deux processus de classification et de localisation d'objet ont été séparés (voir Figure 2.12).Ces deux derniers

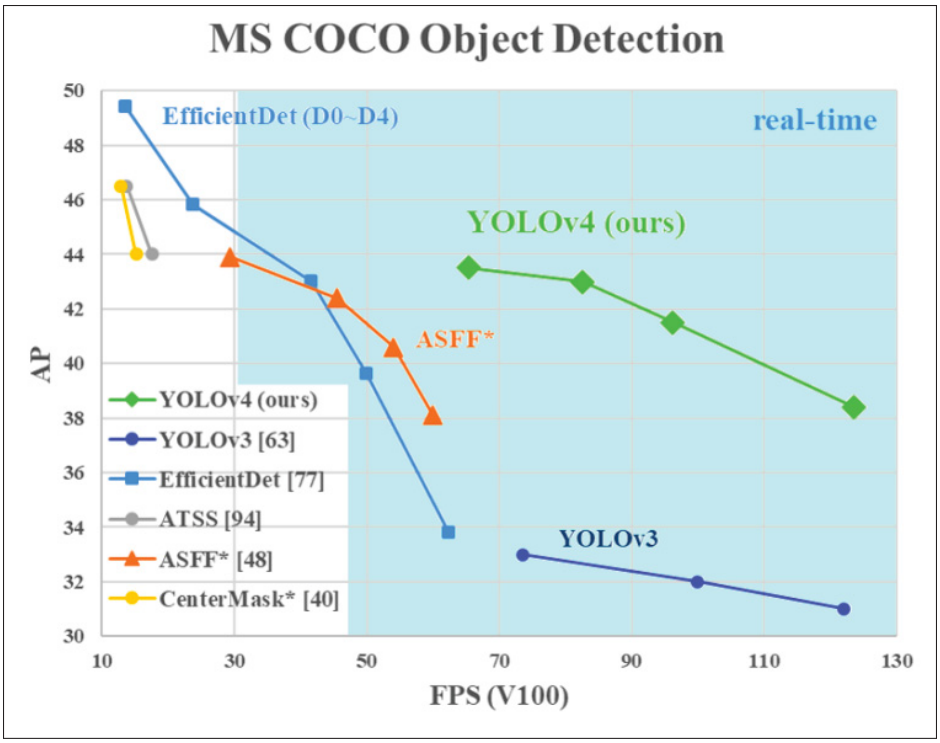

FIGURE 2.11 Comparaison de YOLOv4 avec d'autres méthodes en termes de la précision moyenne (AP) et le temps de traitement (FPS) (Bochkovskiy *et al.*, 2020)

changements ont permis d'avoir une meilleure vitesse de détection que les anciennes versions. L'approche de l'augmentation des données a été améliorée par l'ajout de la méthode Mixup (Zhang *et al.*, 2017), aussi, une technique appelée SimOTA a été développée pour optimiser le processus d'attribution des classes aux boites englobantes détectées durant l'entraînement du réseau.

Dans YOLOv6, le concept de la tête découplée était adopté ainsi que la méthode de détection sans ancre. Aussi, dans cette version, la fonction de perte de régression de la boîte englobante (SIoU) introduite dans Gevorgyan (2022) a été utilisée pour superviser l'apprentissage du réseau (Li *et al.*, 2022). Les performances obtenues ont dépassé même ceux de YOLOX (voir Figure 2.13).

Dans YOLOv7, les auteurs sont arrivés à une précision moyenne 1,5 fois supérieure à la précision moyenne de YOLOv4 avec 75% de paramètres en moins à entraîner. Dans cette version, le

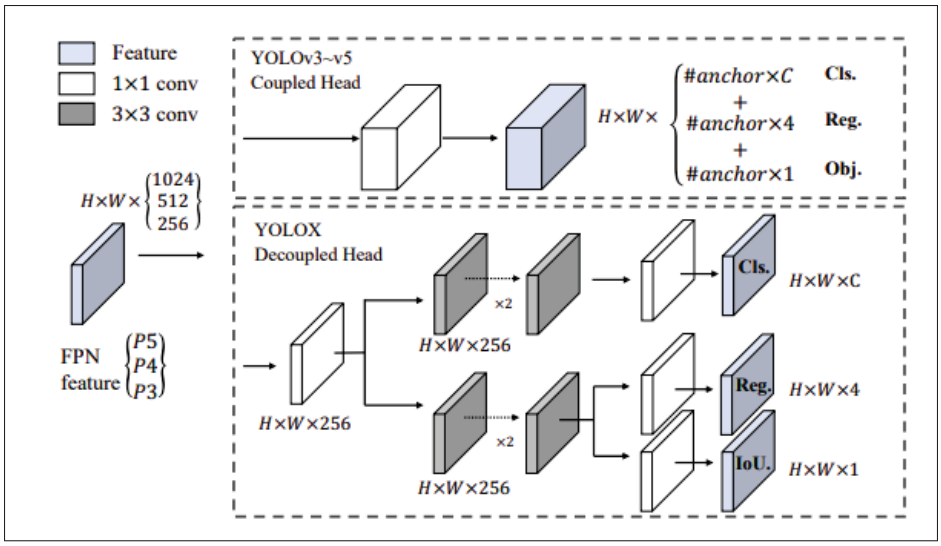

FIGURE 2.12 Comparaison de la tête de YOLOv3 avec YOLOvX (Ge *et al.*, 2021)

bloc de calcul E-ELAN (Extended Efficient Layer Aggregation Network) a été utilisé dans le backbone de cette architecture, ce choix était basé sur la recherche qui dit que plus la distance nécessaire pour que le gradient se propage à travers les couches d'un réseau est courte, plus le réseau sera capable d'apprendre efficacement. Ensuite, ce réseau a été entraîné sur la base de données MS COCO à zéro, sans utiliser les poids d'un réseau pré-entraîné (Wang *et al.*, 2022).

YOLOv8 (Li *et al.*, 2023) est la dernière version du modèle de détection d'objets et de segmentation d'images développé par Ultralytics. Il est conçu comme un cadre qui prend en charge toutes les versions précédentes de YOLO, ce qui facilite la comparaison de leur performance.

2.5 Suivi d'un objet par une caméra monoculaire

Dans la revue de la littérature faite dans Li et al. (2013), les auteurs divisent le processus de suivi d'un objet en 4 étapes principales :

- Initialisation de la position de l'objet, par l'intermédiaire d'un détecteur ou manuellement.

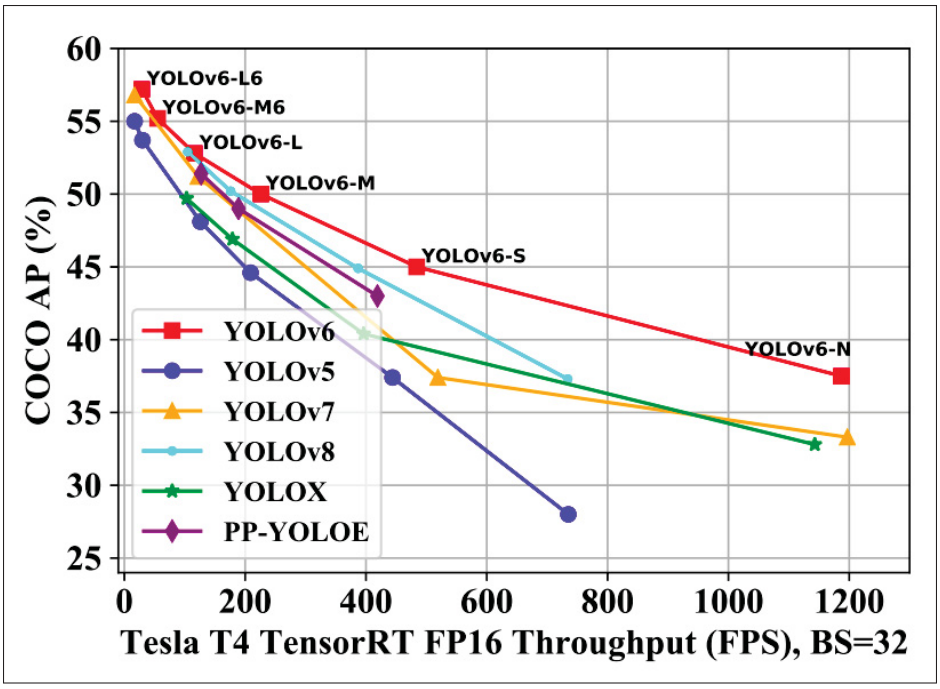

FIGURE 2.13 Comparaison des performance de YOLOv6 avec les différentes versions de YOLO (Li *et al.*, 2023)

- Modélisation de l'apparence, cela consiste à modéliser l'apparence visuellement et statistiquement, c'est-à-dire la modélisation de type de descripteurs et du modèle mathématique utilisés pour identifier l'objet.
- Estimation du mouvement qui est fait généralement par un estimateur d'état comme un filtre de Kalman (Kalman, 1960).
- Localisation d'objets sur l'image.

En général, les approches de suivi d'un objet sont divisées en deux catégories : méthodes de modèles génératifs et méthodes de modèles discriminatoires (Zhao *et al.*, 2019). Dans les modèles génératifs, la première image modélise la région de l'objet à suivre. L'image suivante considère la région la plus similaire comme la nouvelle position de l'objet. Cette méthode repose sur des modèles génératifs comme le suivi du décalage moyen ou le filtre à particules (Arulampalam *et al.*, 2002). Ce dernier (le filtre à particules) a été appliqué dans un contexte de suivi d'objets sur la route par une caméra montée dans un véhicule autonome et a démontré de

bonnes performances (Zhang *et al.*, 2022). Zhao et al.(2019) mentionnent que cette méthode a de la difficulté à suivre les petits objets qui se déplacent avec une grande vitesse.

D'autre part, la méthode de modèles discriminatoires fait une séparation entre l'objet à suivre et l'arrière-plan. Elle se repose sur l'utilisation d'un classificateur qui est entraîné à distinguer entre les deux classes (objet, arrière-plan). Ensuite, ce réseau sera utilisé pour prédire la localisation d'objets sur chaque image. Ce modèle était utilisé dans Li et al. (2016), afin de faire le suivi des objets par une caméra monoculaire montée sur un drone. Il était utilisé aussi dans Wojke et al. (2017) dans le but de faire le suivi de plusieurs objets en temps réel. Dans leur approche, YOLO est utilisé pour détecter et localiser l'objet sur l'image. Ensuite, ces détections étaient les entrées d'un CNN pré-entraîné qui calcule une matrice d'association liée à chaque détection et qui contient les caractéristiques d'apparence des objets. Les vecteurs calculés ont été passés à un filtre de Kalman pour prédire la localisation des objets sur les images futures.

2.6 Conclusion

Dans ce chapitre, nous avons exploré les techniques les plus fréquemment utilisées pour repérer, localiser et suivre un objet à partir d'une séquence d'images afin de le géolocaliser. La variété de ces techniques nous donne la possibilité de combiner différentes approches ou de profiter des avantages de plusieurs techniques pour répondre aux besoins du projet. Dans le chapitre suivant, nous examinerons les techniques employées pour détecter et suivre un objet dans une séquence d'images pour ce projet.

CHAPITRE 3

DÉTECTION ET SUIVI D'UN OBJET SUR LA ROUTE

3.1 Introduction

Afin d'atteindre l'objectif du projet, le pipeline présentée dans la Figure 3.1 a été proposé. Dans ce chapitre, nous allons passer en revue les différentes techniques utilisées pour détecter et suivre les objets d'intérêt sur la route. Pour la détection, nous avons opté pour la version YOLOv5 de l'algorithme You Only Look Once (YOLO), car elle est actuellement l'une des plus performantes sur le marché. En effet, sa bibliothèque est mature, elle a été testée et améliorée par ses développeurs, et elle a été utilisée avec succès dans de nombreux projets pour détecter différents types d'objets. D'autre part, pour le suivi des objets détectés dans une séquence d'images, nous avons utilisé la librairie open source "Norfair", qui permet de faire le suivi d'un objet 2D en temps réel. Cette librairie offre une manipulation facile de ses paramètres ce qui offre la flexibilité pour ajuster son fonctionnement (Alori *et al.*).

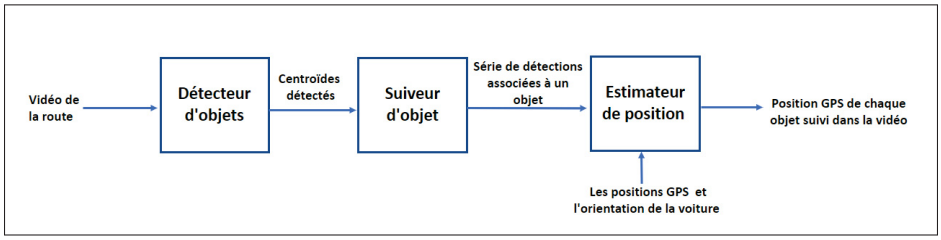

FIGURE 3.1 Pipeline de l'algorithme proposé

3.2 Détection des objets sur la route

Le but d'utiliser un détecteur d'objet est de prédire la localisation et la classe d'un objet sur une image, ce qui est accompli en utilisant une boîte englobante qui définit par des coordonnées x, y, la largeur et la hauteur et est souvent utilisée pour détecter et classer les objets sur une image. La boîte englobante peut être représentée par un rectangle qui entoure l'objet, permettant ainsi de localiser et d'identifier l'objet de manière précise. Parmi les différents détecteurs d'objets

disponibles, You Only Look Once (YOLO) se démarque comme l'état de l'art de la détection d'objets. Sa rapidité de fonctionnement en fait un choix idéal pour les applications en temps réel.

3.2.1 L'algorithme YOLO

Loin de l'architecture neuronale de YOLO qui change un peu dans chaque version, le principe de fonctionnement et la composition principale de ce réseau restent les mêmes. YOLO est basé sur les réseaux de neurones convolutifs (CNN) afin de prédire la classe et l'emplacement d'un objet sur l'image.

L'objectif d'utiliser cet algorithme est d'avoir deux informations, la première est de détecter la présence d'un objet sur l'image et la deuxième est l'emplacement de cet objet sur l'image. Le fonctionnement de YOLO peut être résumé dans les points suivants :

- Pré-traitement : L'image doit être pré-traitée pour être adaptée au modèle (changement de la taille, normalisation...).
- Division en grille : L'image est ensuite divisée en une grille de cellules, chaque cellule sera responsable de prédire la présence et l'emplacement des objets sur sa surface. Le nombre de cellules de la grille et la taille de chaque cellule peuvent varier en fonction de YOLO utilisée.
- Extraction de caractéristiques : YOLO utilise un CNN pour extraire des caractéristiques de l'image (Figure 3.2), où les composants du vecteur sortant sont :

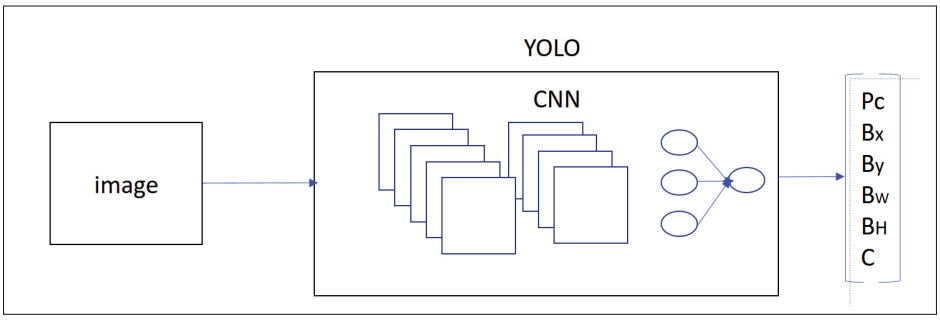

FIGURE 3.2 Illustration de L'algorithme de YOLO

- Pc : probabilité de l'existence d'un objet d'une classe c,

- bx, by, bh, bw : bx, by sont les pixels qui indiquent le coin gauche de la boite englobante et bh, bw sont la largeur et la hauteur de la boite,
- C : la classe de l'objet localisé.

- Prédiction : YOLO utilise des caractéristiques extraites par le CNN pour prédire la probabilité qu'un objet d'une certaine classe soit présent dans chaque cellule, ainsi que les coordonnées de la boîte englobante (bx, by, bh, bw) entourant l'objet.
- Post-traitement : Les prédictions du modèle sont ensuite traitées pour supprimer les faux positifs ou les boîtes englobantes superposées en utilisant la suppression non maximale (NMS). La suppression non maximale est une technique utilisée pour éliminer les doublons et les redondances dans les prédictions du modèle. Pour cela, tout d'abord, toutes les prédictions sont triées en fonction de leur score, pour traiter en premier lieu les prédictions les plus probables et éliminer celles dont la probabilité Pc est inférieure à un seuil donné. Ensuite, l'intersection sur l'union (IoU) est calculée entre la prédiction la plus probable et les prédictions moins probables. L'IoU est une mesure de similarité entre deux boîtes englobantes qui prend en compte leur chevauchement. Si l'IoU de la prédiction en cours de traitement avec une autre prédiction déjà traitée est supérieure à un seuil prédéfini, la prédiction en cours de traitement sera éliminée. Sinon, si l'IoU de la prédiction en cours de traitement avec toutes les autres prédictions déjà traitées est inférieure au seuil, la prédiction en cours de traitement sera prise en compte (Figure 3.3).

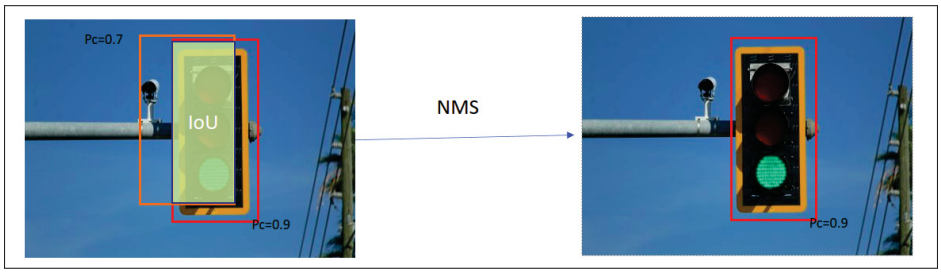

FIGURE 3.3 Suppression non maximale

- Évaluation : L'exactitude des prédictions du modèle est évaluée à l'aide de métriques. Voici la définition de quelques termes utilisés dans l'évaluation d'un modèle :

- Le vrai positif (TP) est quand le modèle prévoit la présence d'un objet qui appartient à une classe C et qu'il est réellement là dans les données (voir Figure 3.4 où le détecteur est censé détecter des nids de poule).

FIGURE 3.4 Exemple de vrai positif de détection de nids de poule

- Le vrai négatif (TN) décrit les cas où le modèle a correctement prédit l'absence d'une certaine classe d'objets.
- Le faux positif (FP) est quand le modèle prévoit la présence d'un objet qui appartient à une classe C mais qu'il n'y en a pas dans les données (voir Figure 3.5 où le détecteur est censé détecter des nids de poule).
- Le faux négatif (FN) est quand le modèle prévoit qu'il n'y a pas un objet qui appartient à une classe C alors qu'il y en a réellement dans les données.
- La matrice de confusion est un moyen de vérifier la performance d'un modèle de classification ou de détection en comparant les prédictions qu'il a effectuées avec les résultats réels. Il se présente sous forme d'un tableau à 2 dimensions qui montre les résultats sous forme de comptages pour chaque combinaison possible entre les étiquettes prédites et les étiquettes réelles. Elle donne une vue d'ensemble des résultats des Vrais Positifs (TP), Faux Positifs (FP), Faux Négatifs (FN) et Vrais Négatifs (TN) (voir Figure 3.6).

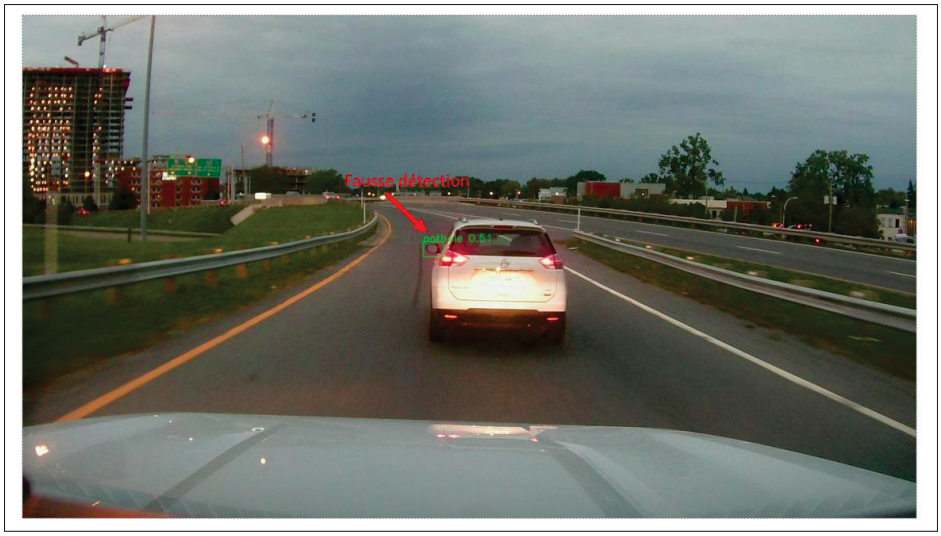

FIGURE 3.5 Exemple de faux positif de détection de nids de poule

Réalité

Prédiction du modèle

	Négatif	**positif**
Négatif	Vrai Négatif	Faux Négatif
Positif	Faux positif	Vrai positif

FIGURE 3.6 Exemple de matrice de confusion

Les quatre valeurs de cette matrice permettent de calculer des métriques de performance comme suit :

- Précision : La précision est le taux correct de prédictions de détection d'un objet qui appartient à une classe C . Elle est calculée en divisant le nombre de vrais positifs par le nombre total de prédictions de présence.

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

- Rappel : le rappel mesure la proportion d'objets qui ont été correctement détectés par le modèle. Il est calculé en divisant le nombre de vrais positifs (TP) sur le nombre total

d'objets réels.

$$Rappel = \frac{TP}{TP + FN} \tag{3.2}$$

- F-mesure : F-mesure est une métrique qui combinent les résultats de la précision et du rappel en une seule valeur. Elle est calculée en utilisant une formule spécifique qui tient compte à la fois de la précision et du rappel. Elle donne une indication globale de la performance d'un modèle en termes de détection.

$$F - mesure = \frac{2 * Precision * Rappel}{Precision + Rappel} \tag{3.3}$$

Ces métriques mesurent la capacité du modèle à détecter et à classer correctement les objets, ainsi que sa capacité à éviter les faux positifs. Le modèle peut alors être affiné et ré-entraîné en fonction de ces résultats afin d'améliorer ses performances.

3.2.2 Augmentation des données

L'augmentation des données est une technique pour générer de nouvelles données à partir des données existantes en utilisant des méthodes comme la rotation, la translation, la réflexion, le zoom et le flou pour augmenter la taille de l'ensemble de données utilisé pour l'entraînement des modèles d'apprentissage automatique. Cela conduit à une amélioration des performances des modèles ainsi qu'à une réduction du phénomène de surapprentissage.

Dans le cas de YOLOv5, les données d'entraînement sont augmentées à l'aide de différentes techniques comme le redimensionnement, mais l'une des techniques les plus intéressantes qui a été implémentée dans cette version est l'augmentation en mosaïque (Bochkovskiy *et al.*, 2020) . Cette technique consiste à combiner quatre images en quatre tuiles de rapport aléatoire pour créer une nouvelle image. L'utilisation de cette technique permet d'augmenter l'accès du réseau à des informations de contexte supplémentaires, même en dehors de leur contexte normal, ce qui permet au modèle d'apprendre à identifier les objets à une échelle plus petite. De plus, l'utilisation de cette technique permet de réduire la taille des lots de données nécessaires

pour l'entraînement du modèle. Une explication plus détaillée de l'architecture de YOLOv5 est donnée en Annexe 1.

Les étapes de l'implémentation de ce détecteur d'objets sont illustrées dans la Figure 3.7, où le seuil de confiance est une valeur seuil qui permet de filtrer les détections avec une probabilité inférieure à cette valeur, la taille de l'image en pixels sur laquelle le modèle YOLO effectue des prédictions d'objets, la taille de lot est le nombre d'images qui sont traitées simultanément par le modèle. Les données d'entrées sont les images venant des caméras montées sur les voitures, tandis que les sorties représentent les boîtes englobantes des objets détectés.

3.2.3 Annotation des données

Pour entraîner un modèle de détection d'objets, on utilise des images qui ont été annotées en définissant les régions contenant les objets à détecter, et en leur attribuant les étiquettes appropriées. Cela consiste à identifier les objets dans les images et les marquer pour l'entraînement du modèle, qui utilise ensuite ces données annotées pour détecter les objets spécifiés dans les images.

Les fameux détecteurs tels que YOLOv5 ont été entraînés sur de grandes bases de données annotées comme MS-COCO pour tester leur performance. Cependant, dans le cas d'une application spécifique comme la nôtre, qui vise à détecter des anomalies sur la route comme les nids de poule, il n'est pas toujours possible de trouver une grosse base de données annotées disponible pour entraîner le modèle. Pour cela, nous avons annoté notre propre base de données en utilisant le logiciel open source CVAT.

Il est à noter que l'entraînement et la surveillance de l'évolution de performance du réseau ne font pas partie de ce projet.

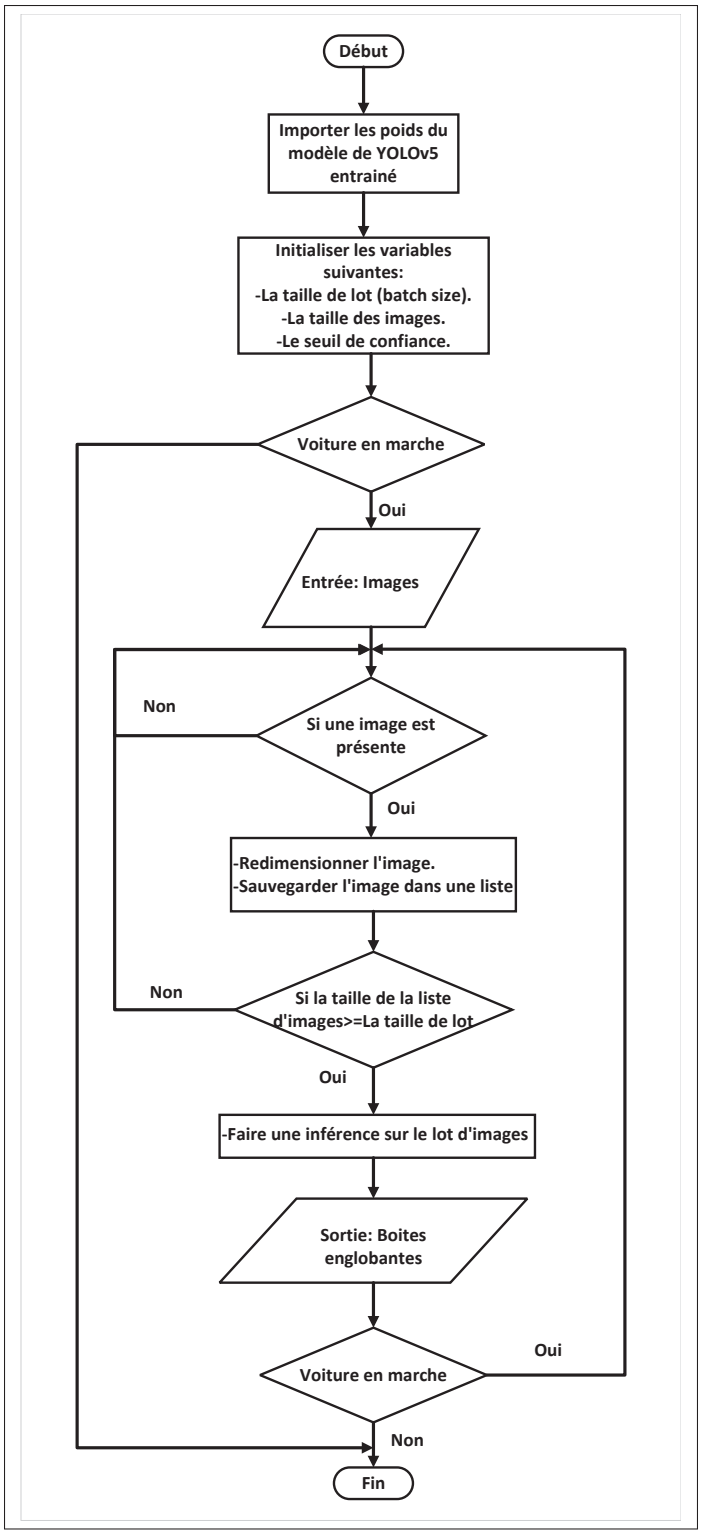

FIGURE 3.7 Organigrame d'implémentation de YOLOv5

3.3 Suivi des objets détectés

Norfair est une bibliothèque Python qui permet de suivre les objets en temps réel et qui peut être personnalisée. Elle vise à ajouter des fonctionnalités de suivi à tout modèle de détection d'objets en utilisant peu de code. Pour utiliser Norfair dans notre cas, il est nécessaire de fournir en entrée les détections effectuées par YOLOv5

Le suiveur de Norfair est basé sur le filtre de Kalman linéaire. En effet, il estime la position future de chaque objet dans la photo en fonction de ses positions antérieures. Il procède ensuite à une correspondance entre ces positions estimées et les points détectés par le détecteur (YOLOv5) en utilisant la mesure de distance. Il existe des fonctions de distance prédéfinies intégrées dans Norfair, toutefois les utilisateurs peuvent également définir des fonctions de distance personnalisées. Cette flexibilité permet de créer des traceurs d'objets adaptés aux besoins spécifiques.

L'implémentation de ce suiveur est illustrée dans la Figure 3.8. En effet, l'instance Tracker de cette bibliothèque est initié avec un certain nombre de paramètres :

- hit inertia min : Chaque objet suivi possède un compteur (C) qui enregistre le nombre de fois où il a été associé à une détection. Lorsqu'il est associé à une détection, ce compteur augmente, et s'il ne l'est pas, il diminue. Si l'objet ne trouve pas de correspondance pour un certain nombre d'images consécutives et que son compteur tombe en dessous d'une valeur donnée, il est détruit.
- hit inertia max : Cet argument définit la limite maximale de compteur (C), donc combien de temps un objet peut exister sans être associé à une détection pour maintenir des réponses rapides à la disparition d'objets suivis de longue durée.
- initialization delay : Ce paramètre détermine la taille minimale que doit atteindre le compteur de détections d'un objet pour être considéré comme initialisé et renvoyé à l'utilisateur en tant qu'objet suivi. Il est important que cette valeur soit inférieure à la limite maximale de hit inertia max pour éviter que l'objet ne soit jamais initialisé. Si la valeur est définie à 0, les

objets seront renvoyés à l'utilisateur dès leur première détection, mais cela peut entraîner l'apparition et la disparition immédiate d'objets.

- distance function : La fonction de distance est utilisée pour calculer la distance entre les objets nouvellement détectés et ceux qui sont actuellement suivis. Dans notre cas, la distance euclidienne a été utilisée.
- past detections length : Nombre de détections à conserver en mémoire. Dans le cas où il y a plus de détections, l'algorithme maintient ce nombre fixe de détections en les répartissant uniformément tout au long de la durée d'apparition de l'objet et donc en supprimant les détections les plus anciennes. Dans notre cas nous avons fixé 16 détections.
- distance threshold : Le seuil de distance maximale est défini pour déterminer jusqu'où une correspondance peut être établie. Si une détection ou un objet suivi est plus éloigné que ce seuil, il ne peut pas être pris en compte par le suiveur.

L'instance du suiveur reçoit des détections sous format d'une liste des centroïdes des objets détectés. Les centroïdes représentent les centres des boites englobantes. Une fois cette instance reçoit des détections qui sont les centroïdes des objets détectés, elle met à jour l'état des objets suivis, les objets qui ont déjà été initialisés, avec les détections correspondantes et renvoie les détections restantes qui n'ont pas trouvées de correspondance. Puis, pour chaque détection restante , elle initialise un nouvel objet suivi et l'ajoute à la liste des objets suivis. Finalement, elle retourne une liste des objets suivis initialisés et ceux qui ont déjà été suivis avec leurs listes de détections.

Comme mentionné précédemment, le suiveur (tracker) enregistre une liste de détections contenant les pixels correspondant aux centroïdes des objets détectés dans une séquence d'images. Cette sauvegarde est réalisée de manière à garantir une répartition uniforme dans le temps. Cependant, des tests effectués sur l'algorithme de géolocalisation présenté dans le prochain chapitre ont révélé que la variance des pixels sauvegardés a une influence significative sur la précision de la géolocalisation (voir Figure 3.9). Bien que, dans certains cas, une variance plus petite puisse entraîner moins d'erreurs, il existe d'autres facteurs qui peuvent également influencer le résultat. Par exemple, dans un cas où la variance est petite, il se peut que les mesures GPS soient plus

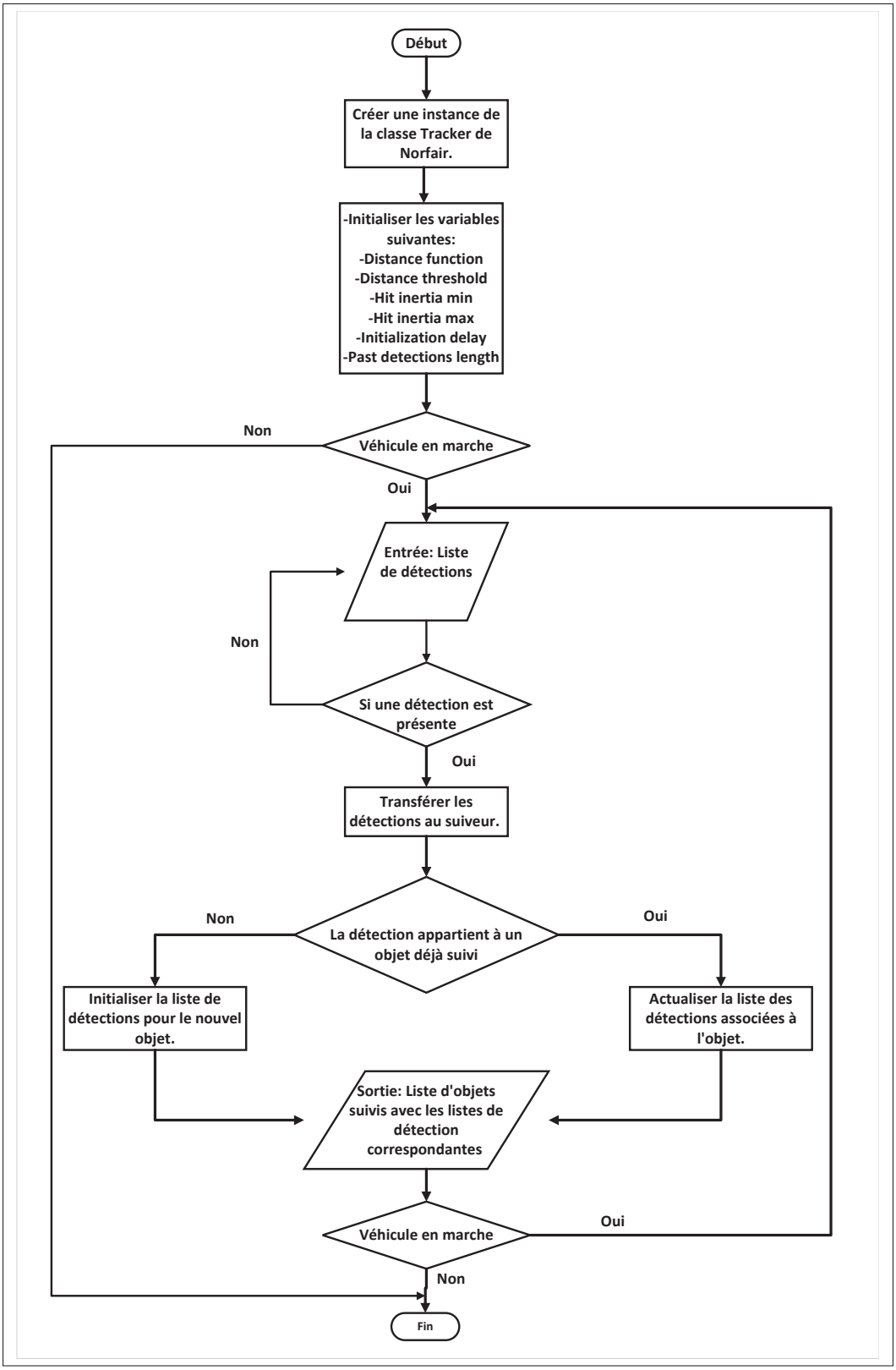

FIGURE 3.8 Organigrame d'implémentation du suiveur

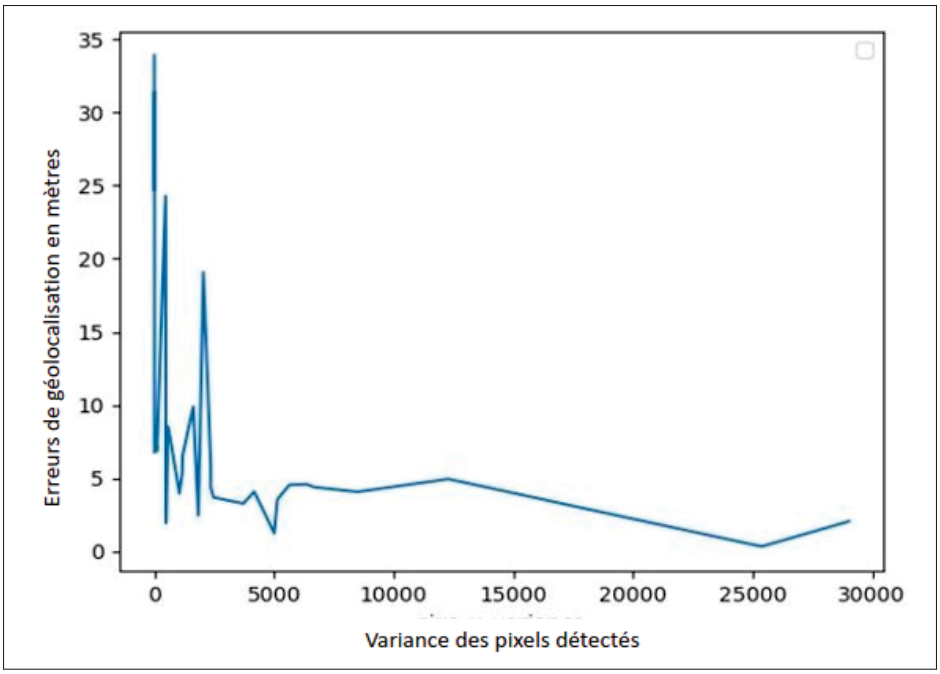

FIGURE 3.9 L'erreur de géolocalisation en fonction de la variance des pixels détectés

précises ou que la détection des objets soit meilleure, ce qui peut réduire l'erreur. Par conséquent, il est préférable de conserver les détections en fonction de leur répartition spatiale plutôt que de leur uniformité temporelle. Autrement dit, la liste de détection prend les 16 premières détections au début, cela étant la taille fixée pour la liste de détections pour chaque objet. À l'arrivée de la 17e détection, l'une des détections pourrait être remplacée pour assurer le plus de déplacement possible de l'objet sur la séquence d'images. Pour choisir la détection qui sera remplacée, il est préférable d'intégrer la nouvelle détection de manière à garantir une combinaison avec la plus grande variance de la liste de détections. Cette approche permet d'avoir un éventail plus large de pixels observés, maximisant ainsi le déplacement du centroïde détecté dans l'image et améliorant la précision de la géolocalisation. Cette modification a permis de réduire l'erreur moyenne sur l'ensemble des exemples de test.

La sélection des données se fait à l'aide de l'algorithme 3.1. Ce dernier commence par initialiser une variable "varianceIndex" à zéro et "toUpdate" à "False". Il vérifie ensuite si la dernière

détection est déjà présente dans la liste de détections passées. Si tel est le cas, l'algorithme se termine sans effectuer de mise à jour. Sinon, il ajoute la dernière détection à la liste si la taille maximale de cette dernière n'est pas atteinte. Autrement, si la liste est déjà pleine, l'algorithme calcule la variance de la liste de normes qui contient les normes euclidiennes des pixels de la liste de détections passées. Puis, à chaque itération, l'algorithme remplace un élément de la liste des normes par la norme euclidienne de la dernière détection. Par la suite, l'algorithme vérifie si cette variance est supérieure à la variance actuelle de la liste de normes. Dans le cas où il y a une augmentation de la variance, l'algorithme met à jour "varianceIndex" et "toUpdate" en conséquence. Après cela, l'algorithme supprime la détection de la liste de détections passées qui a le plus de variance si elle est remplacée par la nouvelle détection, puis ajoute la dernière détection à la liste de détections passées. Finalement, l'algorithme retourne la liste de détections passées mise à jour.

La moyenne des normes de la liste de détections passées :

$$\bar{d} = \frac{\sum_{i=1}^{n} normes_i}{n} \tag{3.4}$$

où n est le nombre de détections passées et $normes_i$ est la distance entre la i-ème détection passée et la dernière détection. La variance des normes de la liste de détections passées :

$$variance = \frac{\sum_{i=1}^{n} \left(normes_i - \bar{d}\right)^2}{n-1} \tag{3.5}$$

Cette formule calcule la variance de la distribution de normes de la liste de détections passées. La norme d'un pixel (x, y) est donnée par la formule :

$$|(x, y)| = \sqrt{x^2 + y^2} \tag{3.6}$$

Algorithme 3.1 Sélection de détection

Input : List of detections for a tracked object : `PastDetections`, Last detection of the object : `LastDetection`
Output : Updated list of detections

varianceIndex ← 0 ; *toUpdate* ← False ;
if *`LastDetection` ∈ `PastDetections`* **then**
  `Return`
end if
if *length(`PastDetections`) < past detections length* **then**
  `Add LastDetection` to `PastDetections` ;
end if
if *length(`PastDetections`) = past detections length* **then**
  norms ← [*norm*(pixel) for pixel in `PastDetections`] ;
  lastDetectionNorm ← *norm*(`LastDetection`) ;
  meanDistance ← $\frac{\sum_{i=1}^{n} normes_i}{n}$, where n is the number of past detections ;
  variance ← $\frac{\sum_{i=1}^{n}(normes_i - \bar{d})^2}{n-1}$, where $\bar{d}$ is the mean distance of the past detections ;
  for $i \leftarrow 1$ *to length(`PastDetections`)* **do**
    tempNorms ← list(norms) ;
    tempNorms[i] ← lastDetectionNorm ;
    if *variance(tempNorms) > variance* **then**
      toUpdate ← True ;
      variance ← variance(*tempNorms*) ;
      varianceIndex ← i ;
    end if
  end for
  if *toUpdate = True* **then**
    `del PastDetections[`*varianceIndex*`]` ;
    `Add LastDetection` to `PastDetections` ;
  end if
end if

3.3.1 Exemples d'utilisation des algorithmes

Les figures 3.10 et 3.11 présentent un exemple de détection d'un nid de poule et d'un lampadaire respectivement effectué par YOLOv5 pour les classes correspondantes. Cependant, il est possible que des faux positifs soient détectés (voir Figure 3.12). La Figure 3.13 illustre un exemple de suivi d'un feu rouge à l'aide du suiveur Norfair, où le point bleu indique que cette détection correspond à un objet que nous suivons dans différentes positions de la voiture.

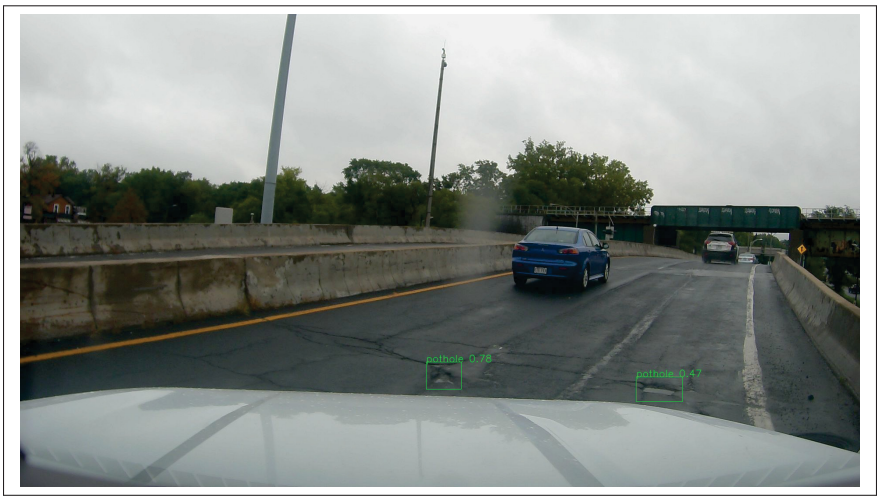

FIGURE 3.10 Détection d'un nid de poule par YOLOv5

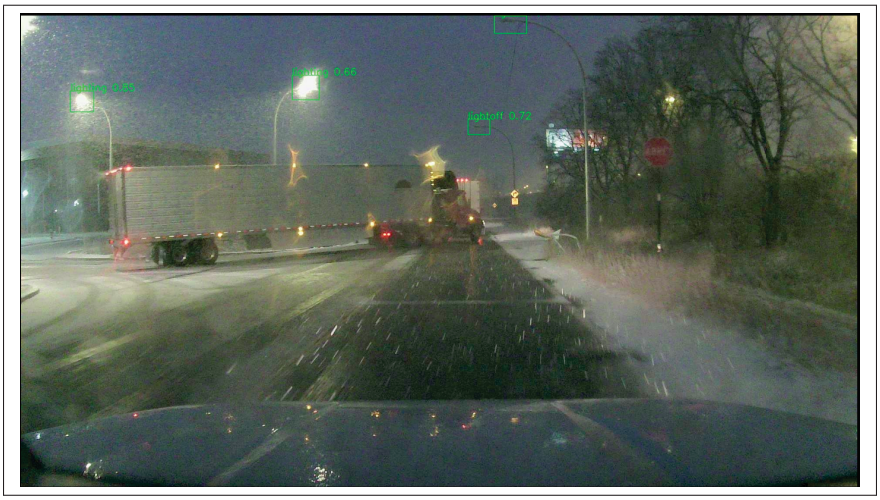

FIGURE 3.11 Détection des lampadaires par YOLOv5

FIGURE 3.12 Détection d'un débris comme nid de poule (FP)

FIGURE 3.13 Suivi d'un feu rouge avec Norfair

3.4 Conclusion

Dans ce chapitre, nous avons examiné les techniques utilisées dans le cadre de ce projet pour détecter et suivre un objet d'intérêt. Dans la prochaine étape, nous utiliserons les sorties de ces algorithmes pour géolocaliser les objets. Dans le prochain chapitre, nous examinerons les techniques utilisées pour trouver la position GPS des objets détectés et suivis dans une séquence d'images.

CHAPITRE 4

ESTIMATION DE LA POSITION GÉOGRAPHIQUE

Après avoir détecté et suivi un objet sur une séquence d'images, la prochaine étape est de déterminer sa position GPS. Il existe différentes méthodes pour effectuer cette tâche, qui peuvent varier en termes de précision et de complexité. Dans notre cas, nous avons choisi une technique basée sur les lignes d'orientation (Zhang *et al.*, 2018b). L'un des avantages de cette technique est qu'elle ne demande pas beaucoup de capacité de calcul, et qu'elle est facile à comprendre et à maintenir, ce qui la rend adaptée pour notre application.

Pour commencer, nous allons voir les données d'entrée nécessaires pour cette étape. Ces données incluent des informations sur les images, telles que les coordonnées de l'objet dans chaque image, ainsi que des données supplémentaires, comme les informations de navigation GPS du capteur utilisé pour prendre les images.

Une fois les données d'entrée compilées, nous calculons les lignes d'orientation. Ces lignes sont basées sur les angles formés entre la caméra et l'objet dans différentes images. Ensuite, ces lignes d'orientation peuvent être utilisées pour déterminer la position GPS de l'objet.

Avant d'entamer les calculs, deux configurations nécessaires ont été faites : la calibration de la caméra et la configuration du capteur GPS.

4.1 Calibration de la caméra

La calibration de la caméra revient à estimer les paramètres internes de celle-ci, autrement dit, c'est le processus de détermination des matrices intrinsèque et extrinsèque qui définissent la relation entre les coordonnées spatiales d'un point dans le monde réel et sa projection sur le plan image. L'algorithme présenté dans l'article (Zhang, 2000) a été utilisé pour déterminer les paramètres de la matrice intrinsèque. Ces derniers ne dépendent pas de la position ou de l'orientation de la caméra, ils peuvent donc être calibrés une fois et être réutilisés, alors que les paramètres extrinsèques de la caméra dépendent de la position et de l'orientation de la

caméra par rapport à l'environnement, ils peuvent donc varier considérablement avec chaque déplacement ou manipulation de la caméra.

4.1.1 Modèle de la caméra

En considérant le modèle de la caméra sténopé qui représente un modèle de formation d'image linéaire (voir Figure 4.1) où la formule qui lie un point réel en 3D (X_{monde}) avec sa projection sur le plan d'image (x_{image}) est la suivante (X_{monde} et x_{image} sont représentés sous forme de vecteurs homogènes 3D et 2D respectivement) :

$$\underbrace{s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{sx_{image}} = \underbrace{\begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}}_{X_{camra}} \tag{4.1}$$

$$\underbrace{\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}}_{X_{camra}} = \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}}_{[\,R\ \ t\,]} \underbrace{\begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix}}_{X_{monde}} \tag{4.2}$$

$$sx_{image} = \underbrace{K\begin{bmatrix} R & t \end{bmatrix}}_{P} X_{monde} \tag{4.3}$$

où s est un facteur d'échelle, la matrice de projection P est le résultat de la multiplication de la matrice intrinsèque K et la matrice extrinsèque. La matrice intrinsèque K décrit les caractéristiques internes de la caméra et elle est utilisée pour projeter les points 3D donnés dans le système de coordonnées de la caméra (X_{camra}) en coordonnées de pixels 2D (x_{image}) comme le décrit l'équation (4.1). La matrice extrinsèque décrit la position et l'orientation de la caméra dans l'espace 3D et elle est utilisée pour projeter les points 3D donnés dans le système

de coordonnées du monde (X_{monde}) sur le référentiel de la caméra (X_{camra}) comme le décrit l'équation (4.2).

f_x et f_y sont les distances focales horizontales et verticales respectivement, C_x et C_y sont les coordonnées du point principal. R est une matrice de rotation et t est le vecteur de translation. Ils sont utilisés pour transformer les coordonnées d'un point exprimé dans le repère des coordonnées réelles en coordonnées exprimées dans le repère de la caméra. Ces matrices sont généralement appelées les paramètres extrinsèques de la caméra.

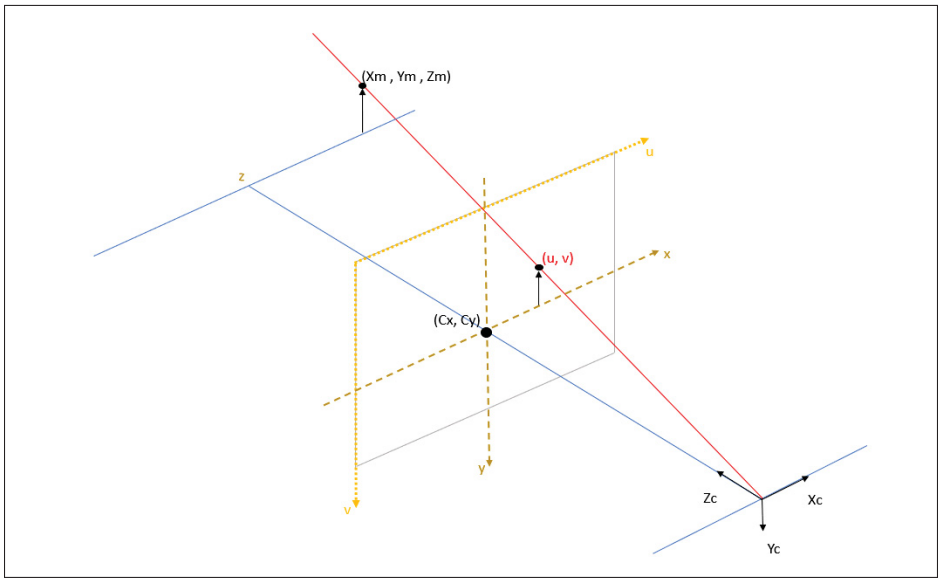

FIGURE 4.1 Modèle de la caméra sténopé

Il est important de noter que la transformation de X_{monde} en x_{image} via la matrice de projection P n'est pas nécessairement réversible. Cette transformation permet de projeter un point en 3D dans un plan 2D, ce qui entraîne une perte d'informations. En effet, le processus de projection implique une division par la profondeur du point 3D, qui ne peut pas être facilement inversée.

Cependant, il est possible d'identifier une ligne 3D pour chaque pixel, ce qui signifie que chaque pixel correspond à une direction dans l'espace et chaque point sur cette ligne a une projection correspondant au même pixel. En se basant sur cela, il est possible de faire une reconstitution 3D en utilisant plusieurs images prises sous différents angles de vue.

4.1.2 Les distorsions dans les images

Les images prises par les caméras peuvent présenter des distorsions en raison des caractéristiques de l'objectif utilisé et des défauts de fabrication. Les distorsions les plus courantes sont la distorsion radiale, la distorsion tangentielle et la distorsion prismatique fine, qui ont un impact sur les coordonnées des points capturés.

Les erreurs de production dans la courbure radiale des éléments de l'objectif causent la distorsion radiale (voir Figure 4.2), tandis que les problèmes de décentrage des éléments de l'objectif causent à la fois la distorsion radiale et tangentielle. La distorsion prismatique fine est causée par des problèmes de production lors de l'assemblage de l'objectif et/ou de l'appareil photo, qui peut causer une légère inclinaison entre les éléments de l'objectif et/ou le capteur, causant à la fois la distorsion tangentielle et radiale (Martins *et al.*, 2020).

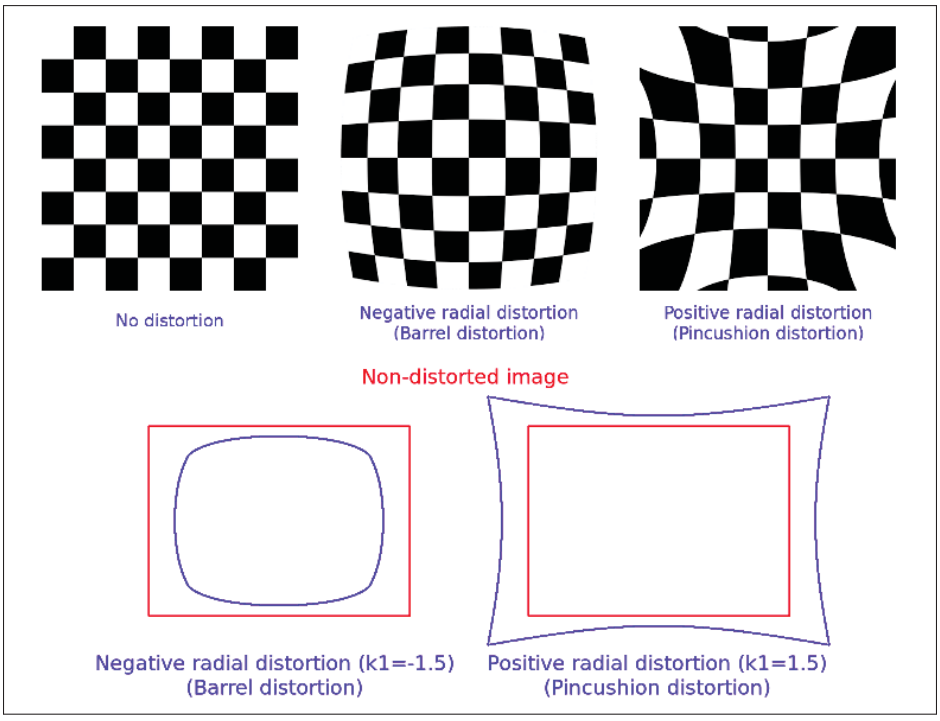

FIGURE 4.2 Distorsion radiale (OpenCV, 2009)

Les équations qui représentent ces distorsions sont les suivantes (Martins *et al.*, 2020) :

$$r^2 = x'^2 + y'^2 \qquad avec \qquad x' = X_c/Z_c \qquad et \qquad y' = Y_c/Z_c \tag{4.4}$$

$$x_d = x' \frac{1 + k1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^5} + 2p_1x'y' + p_2(r^2 + 2x'^2) + s_1r^2 + s_2r^4 \tag{4.5}$$

$$y_d = y' \frac{1 + k1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^5} + 2p_2x'y' + p_1(r^2 + 2y'^2) + s_3r^2 + s_4r^4 \tag{4.6}$$

$$\begin{bmatrix} u_d \\ v_d \\ 1 \end{bmatrix} = K \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \tag{4.7}$$

où k1, k2, k3 sont les coefficients de distorsion radiale, et p1 et p2 sont les coefficients de distorsion tangentielle (Duane, 1971). Les paramètres s1, s2, s3 et s4 modélisent la distorsion prismatique fine (Weng *et al.*, 1992b). Les paramètres k4, k5 et k6 font partie d'un autre modèle appelé modèle de division (Heller *et al.*, 2015) qui permet d'obtenir de meilleurs résultats pour des distorsions plus élevées, typiques des objectifs à angle de champ plus large. Finalement les coordonnées des points de l'image qui ont subi une distorsion sont données par l'équation (4.7).

Il est important de noter qu'il existe un effet de distorsion supplémentaire qui peut apparaître lorsque la caméra est inclinée (autour des axes x et y d'un angle τ_x et τ_y respectivement) de telle sorte qu'elle focalise un plan oblique par rapport à elle. Ces caméras sont appelées caméras Scheimpflug (Zhang & Zhou, 2015), les formules associées à ce type ont été détaillées dans (Martins *et al.*, 2020), ainsi, la Figure 4.3 montre l'effet de changement de quelques paramètres de distorsion.

4.1.3 Détermination des paramètres

Afin de déterminer tous ces paramètres et pouvoir rectifier ces distorsions, nous utilisons la méthode présentée dans (Zhang, 2000). Cette méthode nécessite uniquement un motif plan avec une géométrie connue et des points clés détectables comme un échiquier. Cette méthode a été

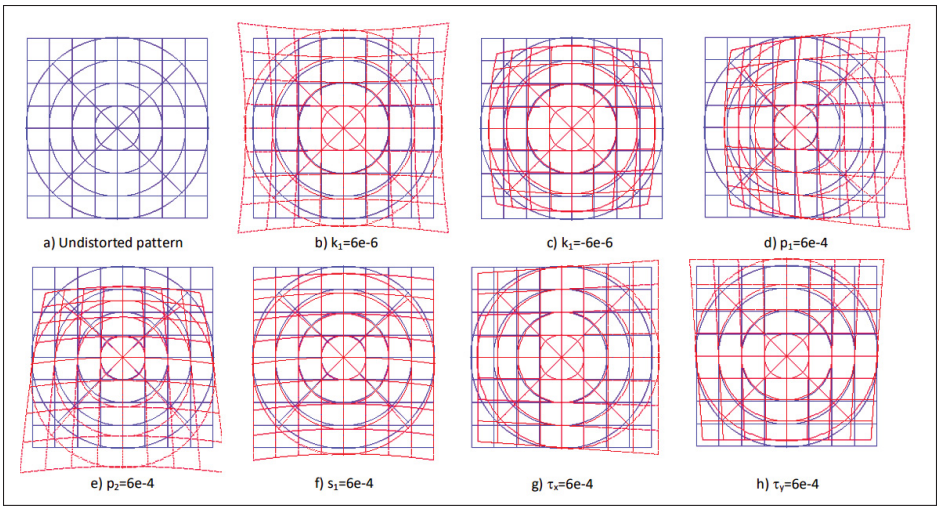

FIGURE 4.3 L'effet de variation des paramètres de distorsions sur l'image (Martins *et al.*, 2020)

intégrée dans la librairie OpenCV qui est une bibliothèque de vision par ordinateur open source, largement adoptée dans les applications de recherche et commerciales.

Cette technique de calibration se résume en 4 étapes :

- Préparer un motif plan avec une géométrie connue telle qu'un échiquier en raison de sa simplicité de manipulation par rapport aux objets 3D complexes ;
- Prendre des photos de l'échiquier à partir de différentes orientations. Cela peut être fait en changeant la direction de la caméra ou en déplaçant physiquement l'échiquier pour le photographier sous différents angles ;
- Détecter les caractéristiques de ces images.
- Estimer les paramètres intrinsèques en utilisant une solution fermée décrite dans le même article ;
- Affiner tous les paramètres en incluant les distorsions de la lentille en minimisant l'estimation de la vraisemblance maximale.

En connaissant tous ces paramètres, nous pouvons ensuite rectifier les distorsions (Weng *et al.*, 1992a), ce qui peut faciliter l'identification des lignes 3D pour chaque pixel et le rendre plus précis et fiable. En corrigeant les distorsions, les lignes droites dans l'image seront plus proches

de l'état réel, et la forme des objets sera plus proche de la forme réelle. Cela peut rendre les calculs d'angles entre un objet et la caméra plus précis, car ils sont basés sur des images plus proche à la réalité.

Cependant, il est important de noter que la rectification de la caméra n'assure pas une correspondance linéaire parfaite entre les pixels dans l'image et l'angle visuel. Il peut y avoir des erreurs de mesure en raison des erreurs de calibration.

4.2 Configuration du GPS

Dans ce projet, nous avons utilisé le GPS u-blox ZED-F9R. C'est un GPS doté de la technologie RTK (Real-Time Kinematic) pour obtenir une précision de positionnement de l'ordre du centimètre. Cette technique repose sur l'utilisation d'un récepteur additionnel (appelé récepteur de base) connecté à une station de référence GPS. Cette combinaison permet d'améliorer considérablement la précision de positionnement par rapport aux systèmes GPS standard. Cependant, pour activer toutes les fonctionnalités permettant le suivi de la voiture, il est nécessaire de configurer préalablement le module GPS.

Le GPS RTK utilise des messages RTCM (Radio Technical Commission for Maritime Services) pour recevoir des corrections de positionnement en temps réel (Landau *et al.*, 2002). Ces corrections sont envoyées par une station de référence GPS à un récepteur RTK, qui les utilise pour calculer une position plus précise. Les informations contenues dans les messages RTCM incluent des erreurs d'horloge, d'éphémérides et de différence de code. Il existe 18 bases de données RTCM au Québec, dont l'emplacement est représenté sur la Figure 4.4. Plus d'information sur ces bases peuvent être repérées dans la référence (MERN). Dans notre algorithme, nous connectons l'ordinateur avec la base la plus proche de la voiture afin d'obtenir une correction plus fiable.

Le module utilisé combine les données d'IMU (accéléromètre + gyroscope), les mesures GNSS, et un modèle dynamique dédié pour fournir un positionnement hautement précis là où le GNSS

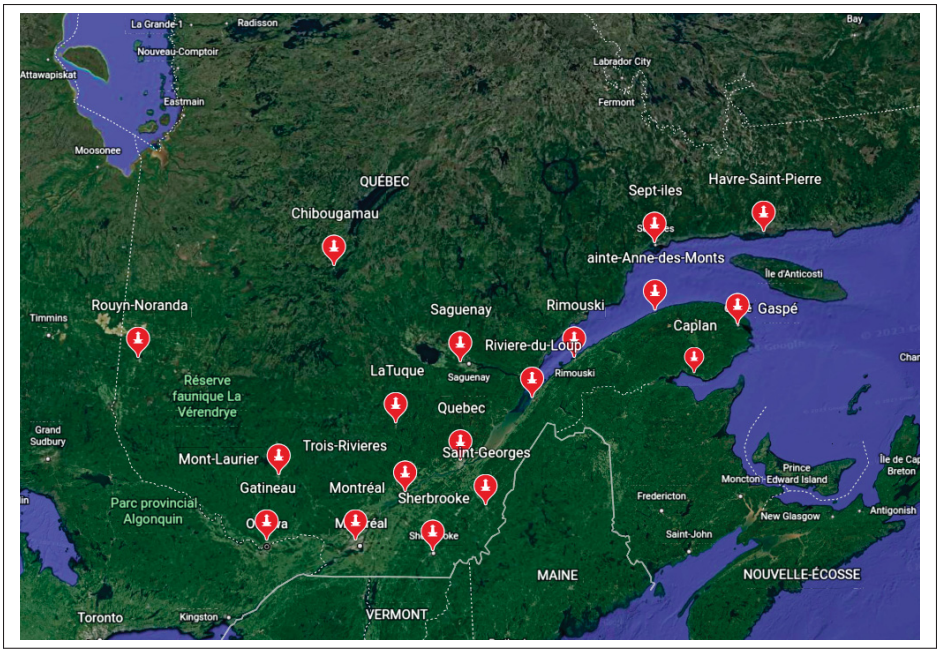

FIGURE 4.4 Bases RTK au Québec

seul échouerait (u blox). D'après la documentation du module utilisé, le GPS doit être monté sur la voiture de la façon illustrée dans la Figure 4.5 :

Le module utilisé (u-blox ZED-F9R) possède plusieurs types de positionnement :

- 2D, qui détermine la position uniquement en latitude et longitude.
- 3D, qui détermine la position en latitude, longitude et altitude.
- FLOAT RTK, qui utilise une station de base unique pour fournir des corrections au récepteur, permettant une précision inférieure à un mètre dans les directions horizontale et verticale.
- FIX RTK, qui utilise une station de base fixe pour fournir des corrections au récepteur, permettant une précision de centimètre près dans les directions horizontale et verticale.
- DR+GNSS ou "mode de fusion", dans lequel GNSS et IMU (système de mesure d'inertie) sont utilisés ensemble.
- DR ou "mode de navigation par défaut", dans lequel le module a perdu les signaux GNSS et ne se base uniquement sur l'IMU. Plus le temps passe dans ce mode, plus la position s'aggrave.

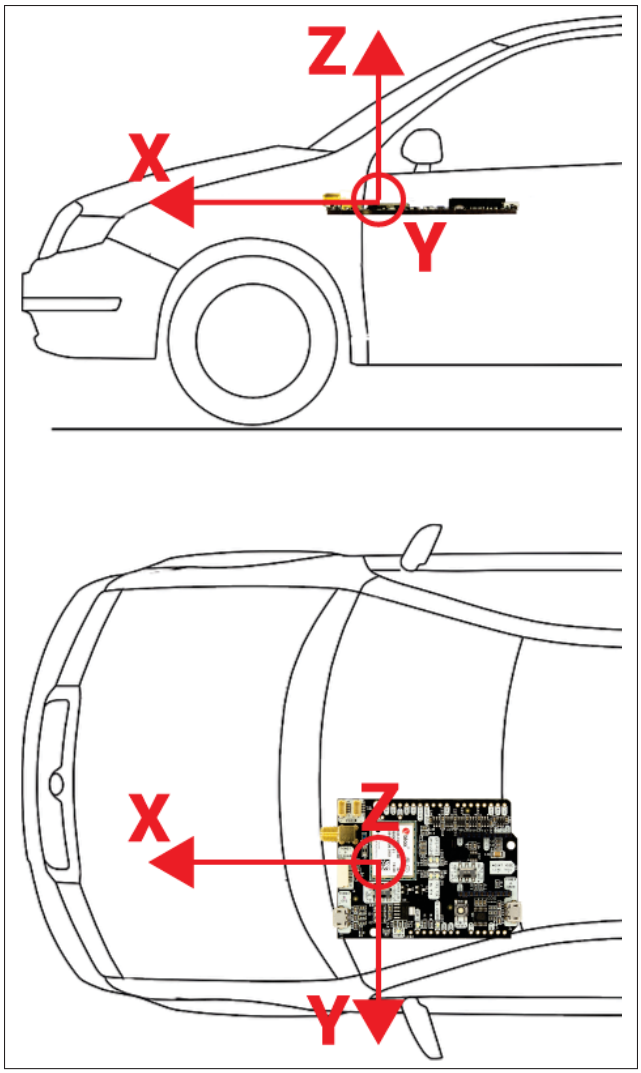

FIGURE 4.5 Montage du GPS sur la voiture (Ardusimple, 2021)

La précision de la position est déterminée par les différents types de positionnement (2D, 3D, FLOAT RTK, FIX RTK) et les modes de fusion et de navigation par défaut assurent une précision élevée même lorsque les signaux GNSS sont perturbés.

Afin d'atteindre la précision maximale du capteur, nous avons configuré le capteur de la manière suivante :

- Les systèmes de navigation par satellite (GNSS) et les fréquences utilisées sont sélectionnés. Chacun des systèmes GNSS peut être activé ou désactivé individuellement. Un système

GNSS est considéré comme activé lorsque la clé d'activation est activée et qu'au moins une des fréquences de ce système est également activée. Dans notre cas, les systèmes GPS, GLONASS, GALILEO et BeiDou (BDS) ont été activés.

- Le module possède trois modes pour gérer les signaux à faible intensité : désactivé, automatique et configuré. Le mode automatique permet au récepteur de détecter et de compenser automatiquement les signaux faibles. Le mode configuré nécessite que l'utilisateur entre une valeur maximale de C/N0 observée dans un environnement à ciel dégagé pour ajuster la compensation. Dans notre cas nous avons choisi le mode automatique.
- Le modèle dynamique automobile a été utilisé pour notre application, il doit être sélectionné en mettant ce paramètre à la valeur 04 (u blox, 2021).
- Le temps entre chaque mesure effectuée par le système de positionnement par satellite (GNSS). Dans notre cas, ce temps a été réglé à 100 ms, cela signifie que le taux de mesure est 10 Hz.
- Le paramètre "RATE-NAV" est utilisé pour déterminer le taux de solutions de navigation en comptant le nombre de mesures effectuées. Dans notre cas, "RATE-NAV" est 10 Hz
- L'alignement automatique de l'IMU dans le système a été activé. Cela permet au système de déterminer automatiquement les angles de décalage entre les différents axes de l'IMU.
- Le paramètre "NAV-PRIO" détermine la fréquence à laquelle le récepteur émet des messages de navigation avec des niveaux de priorité différents. Si ce paramètre n'est pas nul, le récepteur émet des messages prioritaires contenant des données de navigation avec un taux élevé et une faible latence, ainsi que des messages non prioritaires contenant des données de navigation auxiliaires avec un taux faible et une latence plus élevée. Si ce paramètre est nul, tous les messages de navigation ont la même priorité. Dans notre cas, "NAV-PRIO" est 10 Hz.
- La fusion de données venant du GNSS et de l'IMU a été activée. Cela permet d'améliorer les performances du système en environnements où les signaux GNSS peuvent être perturbés ou absents.

4.3 L'acquisition de données

Les données d'entrée de l'algorithme développé sont les suivants :

- Positions GPS (Latitude, Longitude) de la voiture avec leur horodatage.
- L'orientation de la voiture chaque position GPS avec son horodatage, cette mesure est aussi envoyée par le capteur GPS.
- Les détections d'objet sont les résultats produits par un suiveur (tracker) d'objet après avoir suivi un objet à travers une série d'images. Ces détections sont regroupées dans une liste qui contient les informations sur la position de l'objet (centroïde), la taille de la boîte englobante de l'objet et l'horodatage de chaque détection.
- Les paramètres internes de la caméra utilisée dans la prise des photos.

La détection de la position GPS et la détection de l'objet ne sont pas synchronisées. L'exemple de la Figure 4.6 représente un exemple sur la répartition des données acquises dans le temps, néanmoins, le temps entre deux détections GPS reste très faible (0,1 seconde).

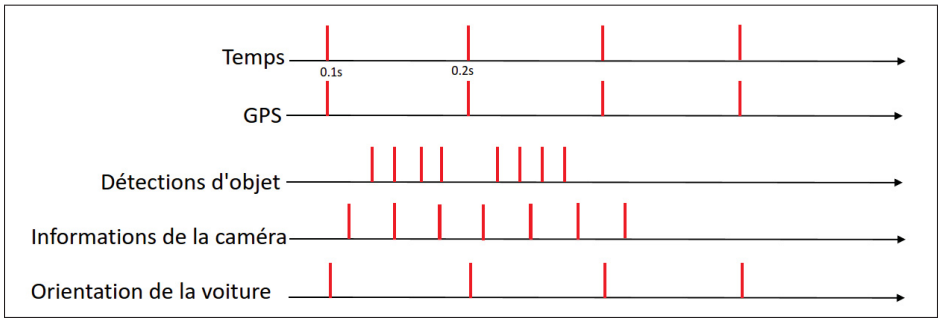

FIGURE 4.6 Chronogramme de la répartition des coordonnées GPS et la détection d'objet dans le temps

4.4 Projection des coordonnées GPS

Les données de positionnement des véhicules reçues sont sous forme de coordonnées WGS84 (IOGP, 2019), qui indiquent la position géographique en termes de magnitudes angulaires se référant au nord ou au sud (Latitude) et à l'est ou à l'ouest (Longitude). Pour pouvoir utiliser

ces données avec des techniques de géométrie cartésienne, il est nécessaire de les convertir en coordonnées locales.

Dans ce projet, nous avons pour objectif de calculer la position des objets pour les afficher sur une carte géographique générée par un service web. Ce service utilise un système de projection Mercator sphérique appelé EPSG :3857, qui est utilisé pour représenter la terre sur un plan. Cependant, cette projection peut entraîner des erreurs. En effet, la projection de Mercator est une projection conforme, ce qui signifie que les angles entre les lignes sont préservés, mais les distances et les surfaces sont déformées. Les distances ne sont correctes qu'à l'équateur et deviennent de plus en plus déformées à mesure que l'on s'éloigne de l'équateur (International Association of Oil & Gas Producers, 2018).

Dans cette transformation, les coordonnées locales (E, N) sont calculées comme suit (IOGP, 2019) :

$$E = FE + \alpha(\lambda - \lambda_0) \tag{4.8}$$

$$N = FN + \alpha ln(tan(\frac{\pi}{4} + \frac{\phi}{2})) \tag{4.9}$$

N et E signifient respectivement "Northing" (latitude projetée en mètres) et "Easting" (longitude projetée en mètres). α est la longueur du demi-grand axe de l'ellipsoïde de référence, c'est-à-dire le rayon de l'équateur, λ est la longitude, $\lambda 0$ est la longitude d'origine naturelle, FN et FE sont la fausse abscisse et la fausse ordonnée, ce sont des valeurs linéaires appliquées à l'origine des coordonnées pour s'assurer que toutes les valeurs x et y sont positives, ϕ est la latitude.

Pour faire la transformée inverse, c'est-à-dire, des coordonnées locales vers les coordonnées GPS, les calculs sont faits comme suit (IOGP, 2019) :

$$D = \frac{(FN - N)}{\alpha} \tag{4.10}$$

$$\phi = \frac{\pi}{2} - 2atan(e^D) \tag{4.11}$$

$$\lambda = \lambda_0 + \frac{(E - FE)}{\alpha} \tag{4.12}$$

4.5 Interpolation des coordonnées locales

Après avoir converti les positions GPS en coordonnées locales, il est nécessaire de retrouver les coordonnées de la voiture lors des détections d'objets. Cela est nécessaire car le capteur GPS envoie la position de la voiture toutes les 0,1 secondes, ce qui ne correspond pas nécessairement à la position de chaque détection.

La Figure 4.7 illustre les positions manquantes. Dans cet exemple, le but est de géolocaliser l'objet (en vert), cet objet a été détecté et identifié dans les positions en rouge et dans un intervalle de temps qui tombe entre les deux détections GPS (en bleu).

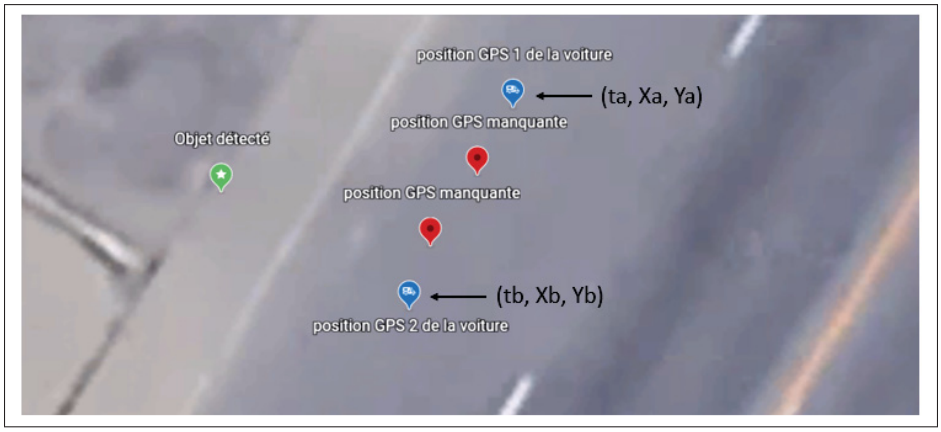

FIGURE 4.7 Illustration des données GPS

Pour trouver les coordonnées locales des points manquants, nous pouvions faire une interpolation en se basant sur la position de la voiture et sa vitesse comme suit :

En utilisant la notation suivante :

- t : le temps du détection de l'objet.
- ta, tb : le temps de la détection GPS 1 et 2 respectivement.
- Xa, Ya, Xb, Yb :les coordonnées locales du point GPS 1 et 2 respectivement.
- Va, Vb : vitesses aux points 1 et 2 respectivement.

$$X(t) = X_a + V_a * (tb - ta) \tag{4.13}$$

$$Y(t) = Y_a + V_a * (tb - ta) \tag{4.14}$$

Les équations (4.15) et (4.16) sont valables dans le cas où la vitesse est constante, sinon il est nécessaire d'utiliser une méthode d'interpolation qui prend en compte la variation de vitesse. L'une des méthodes couramment utilisée est l'interpolation par spline cubique. Cependant, étant donné que notre fréquence d'échantillonnage est de 10 Hz et que les mesures de vitesse ne sont pas synchronisées avec les mesures GPS, nous devrons également interpoler les données de vitesse. Nous avons décidé de ne faire l'interpolation qu'avec les mesures de position. Pour cela, nous avons utilisé une interpolation linéaire des coordonnées X et Y par rapport au temps. Une interpolation linéaire des deux coordonnés (X,Y) par rapport au temps a été faite comme suit :

$$X(t) = \frac{X_a - X_b}{t_a - t_b}t + \frac{t_a X_b - t_b X_a}{t_a - t_b} \tag{4.15}$$

$$Y(t) = \frac{Y_a - Y_b}{t_a - t_b}t + \frac{t_a Y_b - t_b Y_a}{t_a - t_b} \tag{4.16}$$

D'autre part, z est la hauteur de la caméra par rapport au sol qui est toujours constante, les points sont sous la forme (x,y,z), cette hauteur sera utilisée pour estimer la hauteur de l'objet par rapport au sol. La Figure 4.8 montre un exemple de résultat de l'interpolation faite sur un passage de la voiture.

4.6 Mesure des lignes d'orientation voiture-objet

Après avoir estimé la position de la voiture aux moments des détections et compilé une liste des centres des objets sur chaque image de la séquence, nous avons utilisé une technique basée sur les lignes d'orientation décrite dans la référence (Reed *et al.*, 2008) pour déterminer la position géographique de ces objets. Cette méthode permet de calculer le vecteur de direction de la caméra vers l'objet en utilisant la position du centre de l'objet sur l'image. Ensuite, les orientations de la caméra par rapport à la voiture et de la voiture par rapport au monde réel

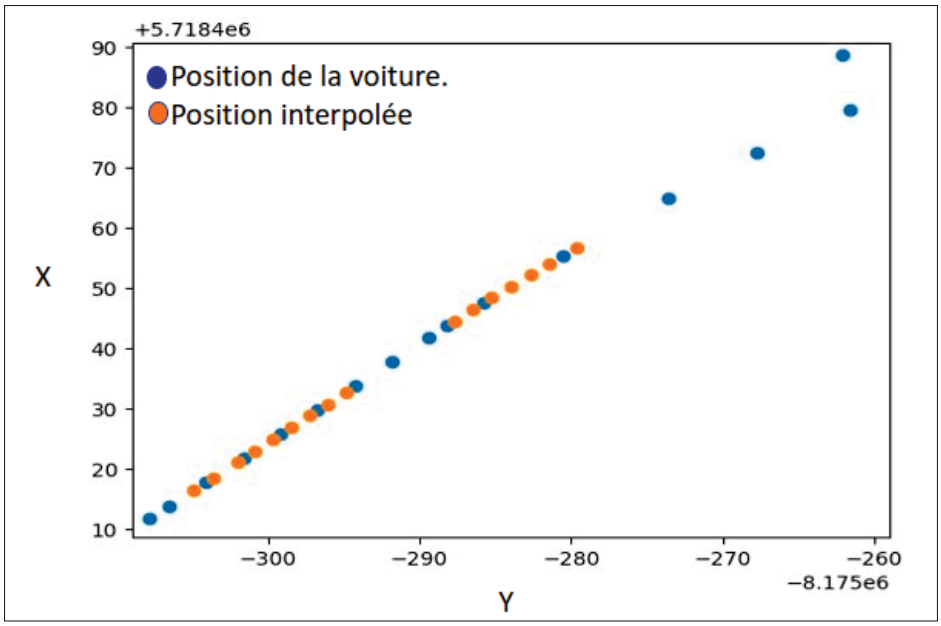

FIGURE 4.8 Exemple de l'interpolation des positions aux moments de détections

sont utilisées pour calculer les paramètres de la ligne passant par l'objet et la voiture dans le référentiel du monde réel. Le but final est d'identifier une ligne 3D entre l'objet et la position de la voiture dans le monde réel pour chaque image, ce qui permet de déterminer la position géographique de l'objet (voir Figure 4.9).

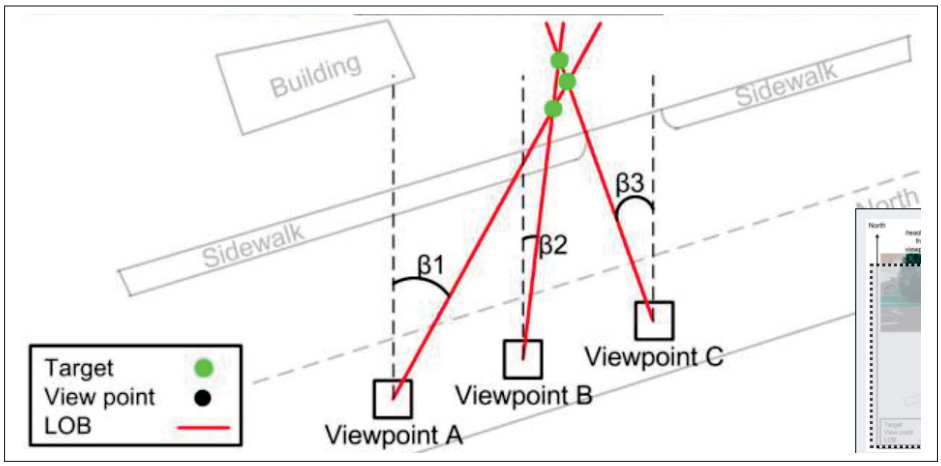

FIGURE 4.9 Localisation par lignes d'orientation (Zhang et al., 2018)

4.6.1 Vecteur de direction caméra-objet

Pour mesurer l'orientation de l'objet par rapport à la voiture, il est nécessaire de connaître son orientation par rapport à la caméra. Les paramètres internes de la caméra sont connus, ce qui permet de calculer le vecteur entre l'image et l'objet en utilisant l'image et le centroïde de l'objet. Avant de procéder à ce calcul, les images ont été rectifiées, c'est-à-dire que les distorsions ont été corrigées et donc les centroïdes des objets détectés ont été rectifiés ainsi que le centre optique (C_x,C_y). Ce calcul se fait comme suit :

- C_x, C_y sont les coordonnées du point principal.
- x_{obj}, y_{obj} sont les coordonnés du pixel du centre de l'objet dans l'image.
- f_x, f_y sont les distances focales
- La matrice K est la matrice des paramètres intrinsèques.

Nous avons l'équation qui lie les coordonnés de l'objet dans le repère de la caméra et pixel correspondant sur l'image :

$$x_{image} = KX_{camera} \tag{4.17}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{4.18}$$

et donc nous pouvons calculer le vecteur de direction de l'objet par rapport à limage comme suit :

$$V_{objet}^{image} = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} (u - C_x)/f_x \\ (v - C_y)/f_y \\ 1 \end{bmatrix} \tag{4.19}$$

4.6.2 Mesure de l'orientation de la caméra par rapport à la voiture

La position de la caméra par rapport au référentiel de la voiture n'est pas alignée, ce qui nécessite la mesure de trois angles d'Euler : le lacet (rotation autour de l'axe z), le tangage (rotation autour

de l'axe y) et le roulis (rotation autour de l'axe x) (Yaw, Pitch, Roll) comme illustré dans la Figure 4.10. De plus, il existe un décalage entre l'emplacement de la caméra et celui du capteur GPS sur la voiture (X_{offset}, Y_{offset},Z_{offset}) qui a été mesuré manuellement. Cependant, il est difficile de mesurer les angles de la même façon que le décalage.

Dans cette étude, nous avons considéré que le Roll, Pitch et Yaw de la caméra étaient tous égaux à zéro. En effet, il est difficile de mesurer manuellement avec précision de petits angles, et les tests d'une technique basée sur le calcul de point de fuite (Lee *et al.*, 2020) n'ont pas été concluants car ils nécessitent de s'assurer que la voiture est parfaitement alignée avec les lignes de la route pour obtenir une estimation précise.

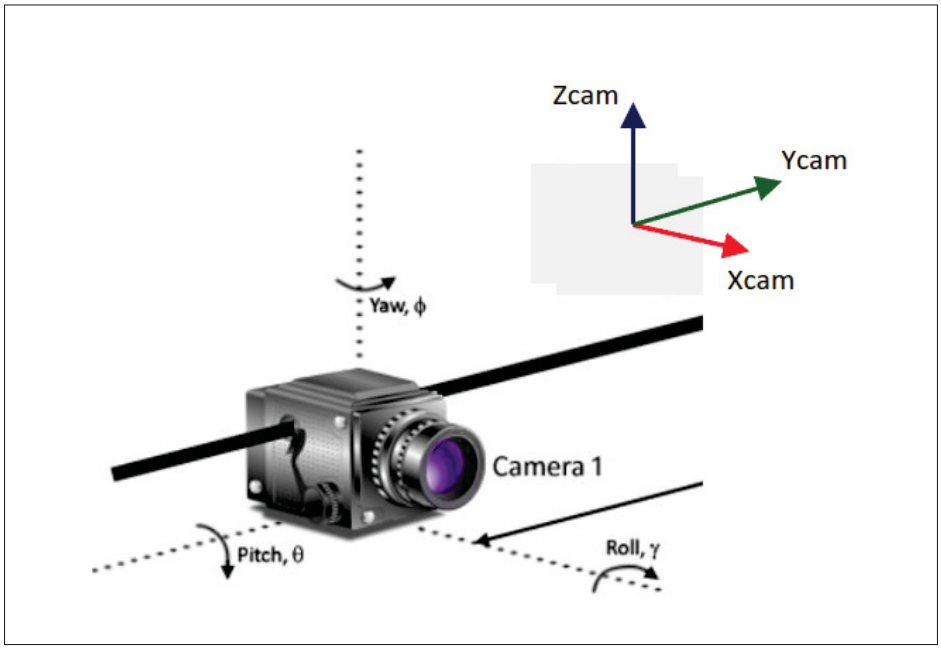

FIGURE 4.10 Lacet, Tangage, Roulis. Adapté de (M.Santoro, et al. 2012)

4.6.3 Mesure de l'orientation de la voiture par rapport au mode réel

L'orientation de la voiture par rapport au mode réel vient du GPS, qui la transmet sous forme de quaternion. Pour convertir ces données en angles d'Euler, nous utilisons les équations suivantes :

$$yaw_{voiture} = atan2(2(q_w q_z + q_x q_y), 1 - 2(q_y^2 + q_z^2)) \tag{4.20}$$

$$pitch_{voiture} = asin(2(q_w q_y - q_z q_x)) \tag{4.21}$$

$$roll_{voiture} = atan2(2(q_w q_x + q_y q_z), 1 - 2(q_x^2 + q_y^2)) \tag{4.22}$$

où q_w, q_x, q_y et q_z sont les coefficients du quaternion.

La matrice de rotation qui permet de projeter un vecteur V qui est projeté dans le repère de la voiture sur le repère du monde réel est :

$$R_{voiture}^{monde} = R_{yaw_{voiture}} R_{pitch_{voiture}} R_{roll_{voiture}} \tag{4.23}$$

Après avoir effectué les calculs des matrices de transformation, nous pouvons projeter les vecteurs caméra-objet dans le référentiel du monde réel (Figure 4.11) pour obtenir les lignes de direction. Cela simplifiera le calcul de la position géographique ultérieure.

$$V_{objet}^{monde} = R_{voiture}^{monde} . (R_{camera}^{voiture} . (R_{image}^{camera} . V_{objet}^{image}) + T_{camra}^{voiture}) \tag{4.24}$$

$$V_{objet}^{camera} = R_{image}^{camera} . V_{objet}^{image} \tag{4.25}$$

4.7 Calcul du point d'intersection des lignes d'orientation

Une fois les lignes générées et projetées dans le bon référentiel, voir Figure 4.12, nous allons passer au calcul du point d'intersection qui représente les coordonnées géographiques de l'objet en coordonnées locales, sauf que les lignes ne coïncident pas nécessairement au même point en

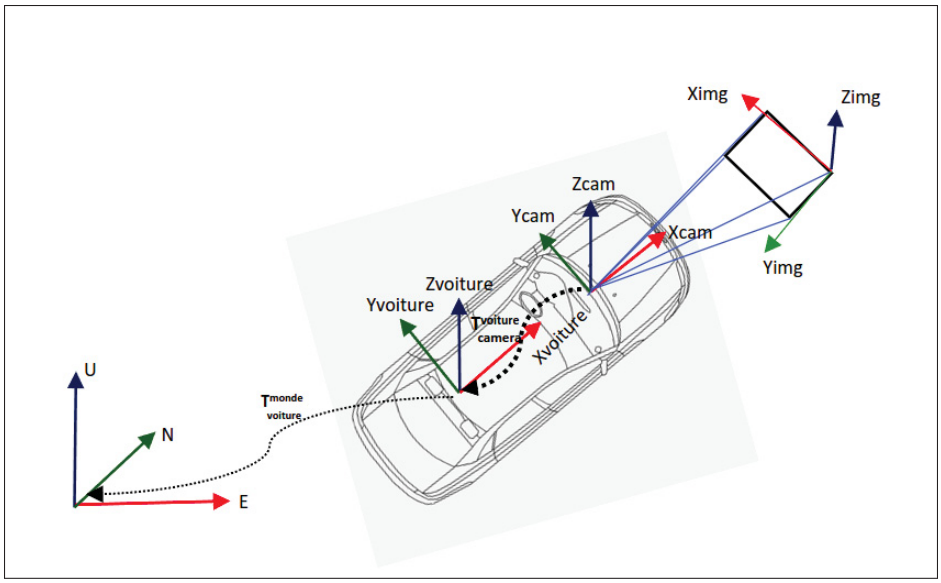

FIGURE 4.11 Changement de référentiels adaptée de (Soto, 2021)

raison des erreurs de mesures. Afin d'obtenir le point d'intersection le plus proche de toutes les lignes, une méthode des moindres carrés a été utilisée.

Le calcul du point d'intersection a été fait avec la méthode démontrée dans (Traa, 2013), cette méthode se fait en plusieurs étapes qui seront démontrées dans cette section.

4.7.1 Définition des paramètres

Les lignes en général peuvent être définies par deux paramètres, un vecteur de direction de la ligne et un point qui lui appartient. Dans notre cas, les lignes sont tridimensionnelles. Les notations suivantes sont utilisées :

- n : vecteur de la direction de la ligne.
- a : point qui appartient à la ligne.
- p : un point dans l'espace.

$$n = \begin{bmatrix} n1 & n2 & n3 \end{bmatrix}^T \; et \; \left\| nn^T \right\|_2 = 1 \tag{4.26}$$

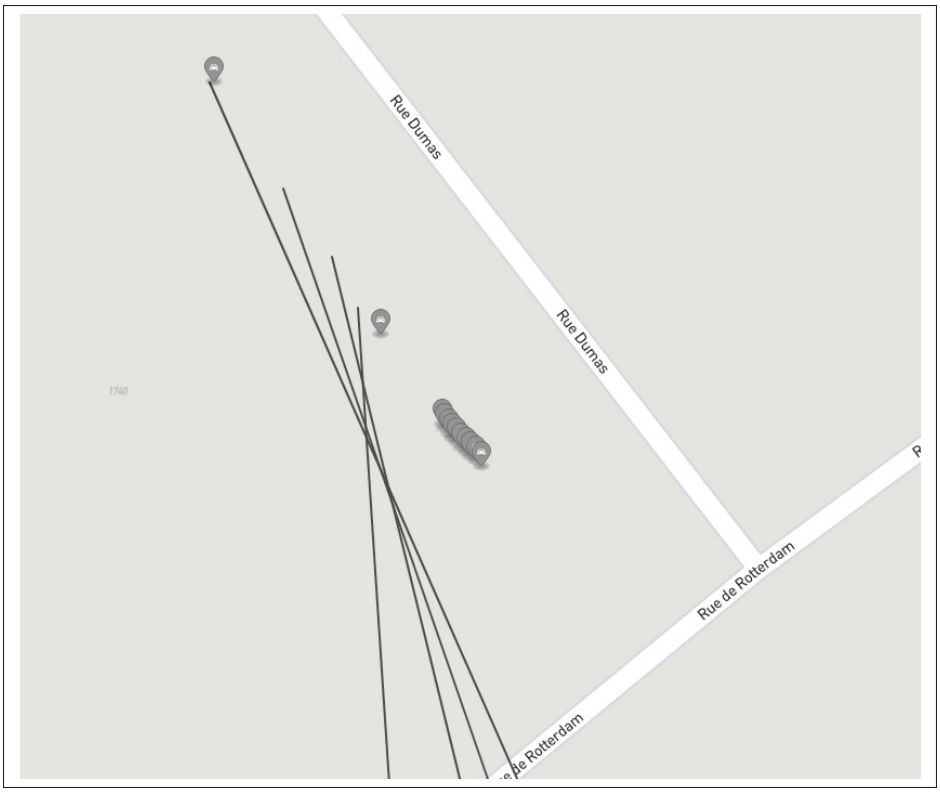

FIGURE 4.12 Les lignes d'orientation générées

$$a = \begin{bmatrix} a1 & a2 & a3 \end{bmatrix}^T, p = \begin{bmatrix} p1 & p2 & p3 \end{bmatrix}^T \tag{4.27}$$

4.7.2 Distance entre un point et une ligne

La Figure 4.13 montre la relation vectorielle de la distance perpendiculaire entre un point et une droite.

La distance perpendiculaire est définie par les équations vectorielles suivantes :

$$D(p,(a,n)) = \left\|(a-p) - ((a-p)^T n)n)\right\|_2^2 \tag{4.28}$$

$$= \left\|(a-p) - nn^T(a-p))\right\|_2^2 \tag{4.29}$$

$$= \left\|(I-nn^T)(a-p))\right\|_2^2 \tag{4.30}$$

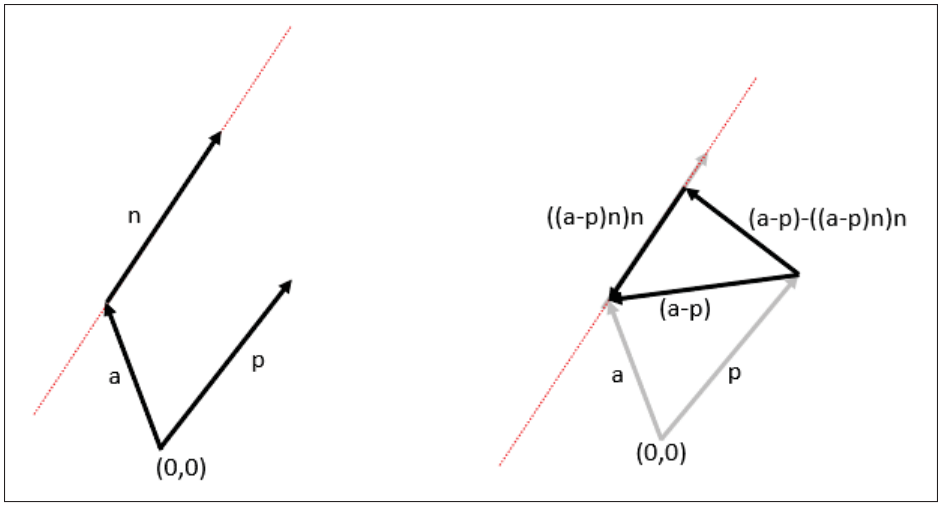

FIGURE 4.13 Distance perpendiculaire entre un point et une ligne

$$= (a-p)^T(I-nn^T)(I-nn^T)^T(a-p) \tag{4.31}$$

$$(I-nn^T)(I-nn^T)^T = I-2nn^T+nn^Tnn^T = I-nn^T \tag{4.32}$$

$$D(p,(a,n)) = (a-p)^T(I-nn^T)^T(a-p) \tag{4.33}$$

4.7.3 Solution des moindres carrés à l'intersection des lignes

Pour calculer le point le plus proche de l'intersection, la distance entre ce point et toutes les lignes K doit être minimale, c'est-à-dire que p doit être calculé de manière à ce que la dérivée de la distance par rapport à p soit nulle . Le calcul se fait comme suit :

$$D(p,(a,n)) = \sum_{i=1}^{K}(a_i-p)^T(I-n_in_i^T)^T(a_i-p) \tag{4.34}$$

$$\frac{\partial D(p,(a,n))}{\partial p} = \sum_{i=1}^{K}-2(I-n_in_i^T)(a_i-p) = 0 \tag{4.35}$$

Ce qui nous amène à l'équation suivante :

$$\sum_{i=1}^{K}(I - n_i n_i^T)p = \sum_{i=1}^{K}(I - n_i n_i^T)a_i \tag{4.36}$$

$$R = \sum_{i=1}^{K}(I - n_i n_i^T), q = \sum_{i=1}^{K}(I - n_i n_i^T)a_i \tag{4.37}$$

Nous pouvons résoudre le système Rp=q soit directement, soit appliquer la pseudoinverse de Moore-Penrose (Courrieu, 2008). La solution du système est le point d'intersection recherché (x, y, hauteur de l'objet).

Les figures montrent les résultats de la méthode de calcul utilisée. La figure 4.14 et la figure 4.15 présentent les résultats de calcul. La figure 4.16 montre la projection de ce point sur la carte. La figure 4.17 montre un exemple en 3D où les points rouges représentent la voiture, les points blues représentent la caméra, le point vert représente le point d'intersection calculé et le point noir représente la position réelle de l'objet. Notant que pour ces exemples les mesures GPS des positions de la voiture étaient précises.

Une fois le point calculé, il ne reste plus qu'à faire la projection sur la carte en utilisant la transformée inverse de EPSG :3857 (IOGP, 2019).

La Figure 4.18 présente un exemple de géolocalisation de trois feux de circulation qui ont été détectés et suivis pendant une séquence d'images. Dans le cas présenté, la précision du GPS était 50 cm près, et l'erreur de la géolocalisation était de 2 mètres.

4.8 Projection d'une position GPS connue sur le plan d'image

Pour estimer la position d'un objet sur une image à partir de sa position GPS connue, nous utilisons une projection sur le plan d'image. Nous commençons par déterminer l'orientation de l'objet par rapport à la voiture en utilisant les positions GPS connues de l'objet et de la voiture. Nous transformons ensuite cette orientation dans le référentiel de la caméra, puis déterminons le

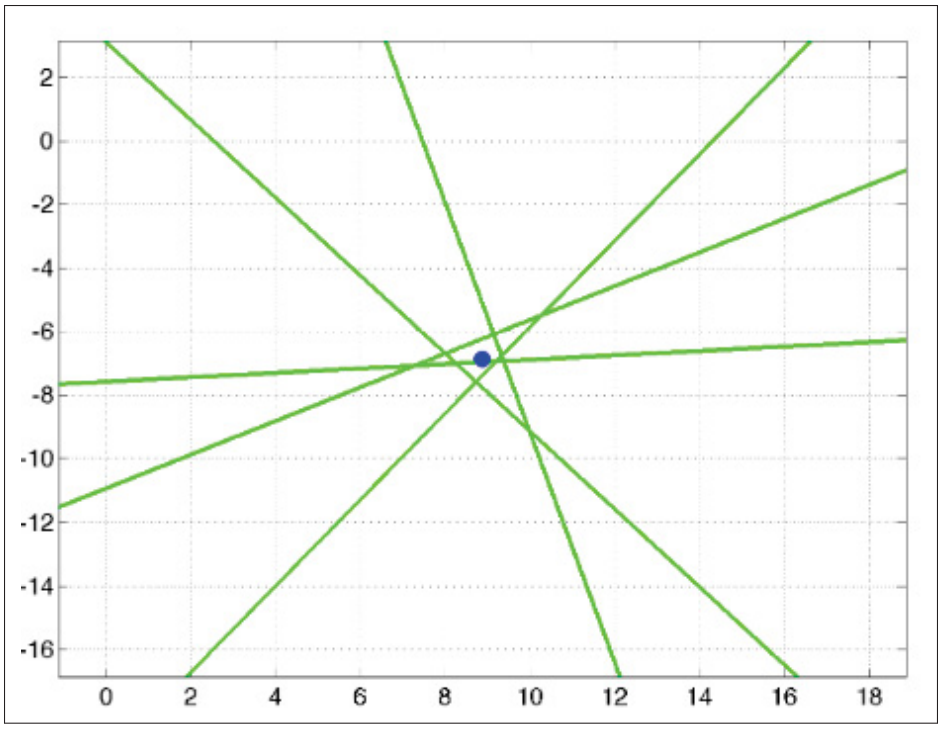

FIGURE 4.14 Exemple trivial du calcul du point d'intersection (Traa, 2013)

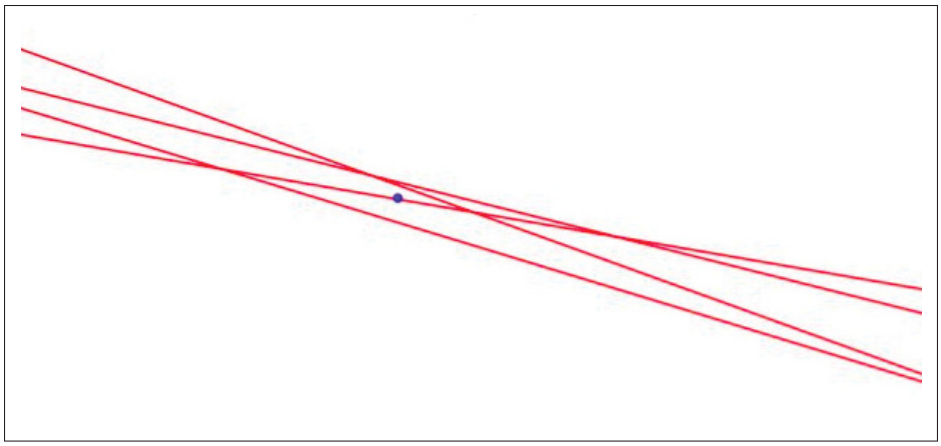

FIGURE 4.15 Calcul de l'intersection d'un ensemble de données

vecteur de direction de l'objet vers l'image. Enfin, nous projetons ce vecteur sur le plan d'image

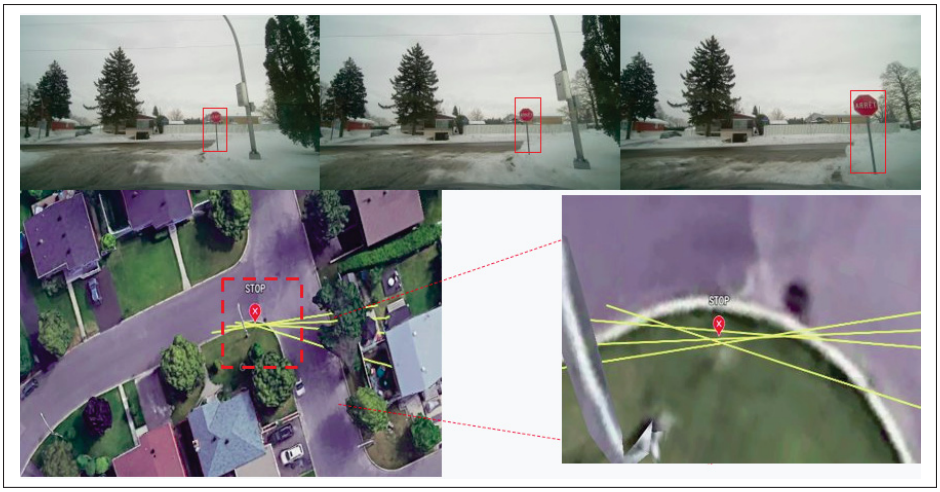

FIGURE 4.16 Projection du point trouvé sur la carte

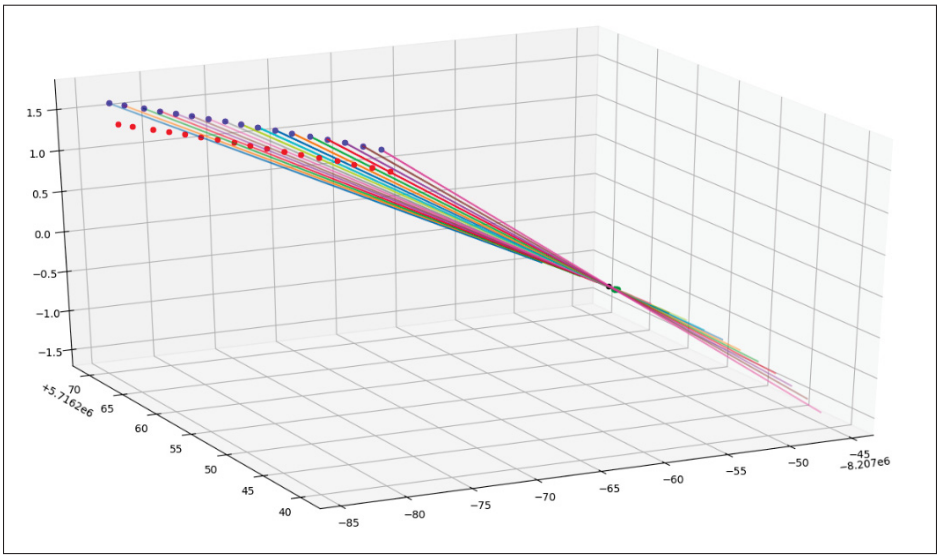

FIGURE 4.17 Illustration du point d'intersection en 3D

pour estimer la position du pixel correspondant à l'objet sur l'image. La projection a été faite comme suit :

- V_{objet}^{monde} =Position GPS de l'objet
- $T_{voiture}^{monde}$ =Position GPS de la voiture

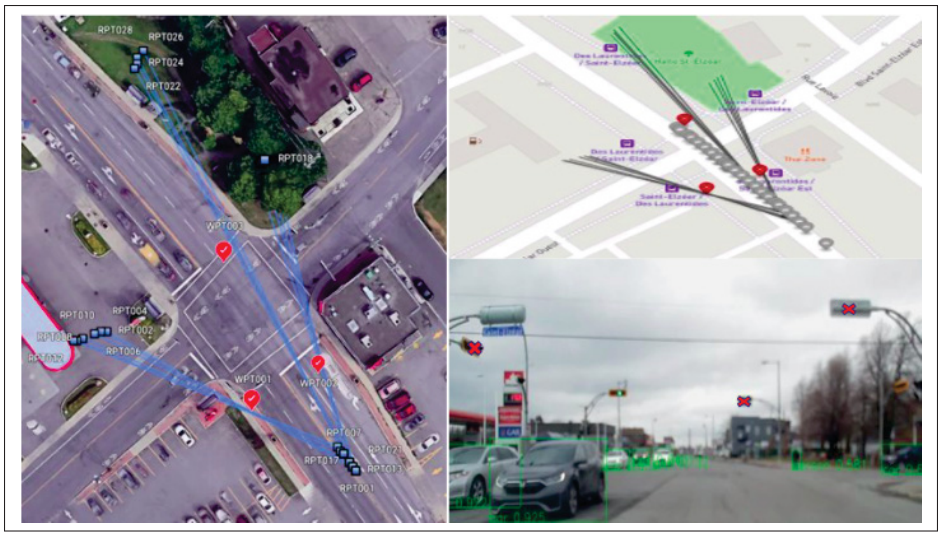

FIGURE 4.18 Exemple de géolocalisation des feux de circulation

- L'inverse de la matrice de rotation est : $R_a^{b^{-1}} = R_a^{b^T} = R_b^a$, ce que nous permet d'inverser les matrices de rotation trouvées dans la section précédente.

Pour trouver le vecteur $V_{objet}^{image} = [x, y, z]^T$, le calcul suivant a été fait :

$$T_{camra}^{voiture} = \begin{bmatrix} X_{offset} & Y_{offset} & Z_{offset} \end{bmatrix}^T \tag{4.38}$$

$$V_{objet}^{voiture} = R_{monde}^{voiture} V_{objet}^{monde} - R_{monde}^{voiture} T_{voiture}^{monde} \tag{4.39}$$

$$V_{objet}^{camra} = R_{voiture}^{camra} V_{objet}^{voiture} - R_{voiture}^{camra} T_{camra}^{voiture} \tag{4.40}$$

$$V_{objet}^{image} = R_{camra}^{image} V_{objet}^{camra} \tag{4.41}$$

Une fois que le vecteur V_{objet}^{image} est trouvé, le calcul du pixel correspondant de l'objet sur l'image est effectué comme suit :

$$u = f_x * x/z + c_x \tag{4.42}$$

$$v = f_y * y/z + c_y \tag{4.43}$$

La Figure 4.19 montre un exemple d'application de cette projection. Dans cette exemple la position GPS de la voiture était précise ainsi que la position de lampadaire pointé.

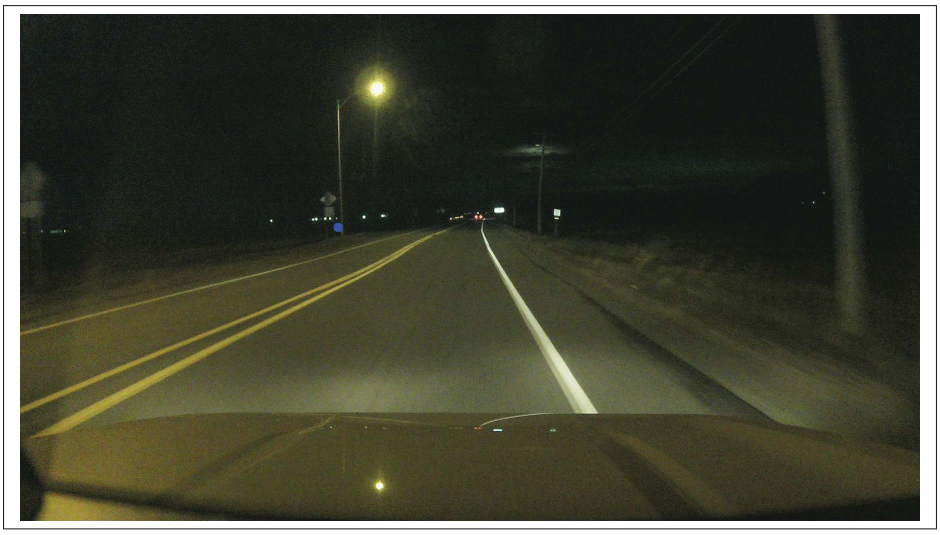

FIGURE 4.19 Exemple de la projection

4.9 Géolocalisation face à une insuffisance de détection

Dans des situations de détection de nids de poule, il peut être difficile d'obtenir une estimation précise de la position GPS en utilisant le calcul d'intersection en raison de la difficulté pour la caméra de les détecter de loin ou d'occultations par d'autres véhicules. Pour résoudre ce problème, on utilise un algorithme géométrique qui permet d'estimer la position GPS d'un objet à partir d'une seule détection en connaissant la hauteur de l'objet.

Cet algorithme est basé sur le calcul de l'intersection d'une ligne avec un plan (Sunday, 2021). En effet, une ligne L en 3D peut soit être parallèle à un plan ou bien se croiser avec ce dernier en un seul point. Nous définissons la ligne L à l'aide de l'équation paramétrique :

$$P(s) = P_0 + s(P_1 - P_0) = P_0 + su \tag{4.44}$$

Tandis que le plan est défini par un point V0 sur celui-ci et un vecteur normal n = (a, b, c). Nous pouvons déterminer si la ligne et le plan sont parallèles ou bien s'ils se croisent en un seul point. Si n*u = 0, cela signifie que le vecteur de direction de la ligne u est perpendiculaire au vecteur normal du plan n, et donc que la ligne L et le plan F sont parallèles. Dans ce cas, soit ils ne se croisent jamais, soit la ligne L se trouve complètement dans le plan F.

Si la ligne et le plan ne sont pas parallèles, ils se croisent en un unique point $P(s_I)$ (voir Figure 4.20). Ce point est calculé en utilisant une méthode similaire à celle utilisée pour trouver l'intersection de deux lignes dans un espace 2D :

$$s_I = \frac{-n.w}{n.u} = \frac{n.(V_0 - P_0)}{n.(P_1 - P_0)} \tag{4.45}$$

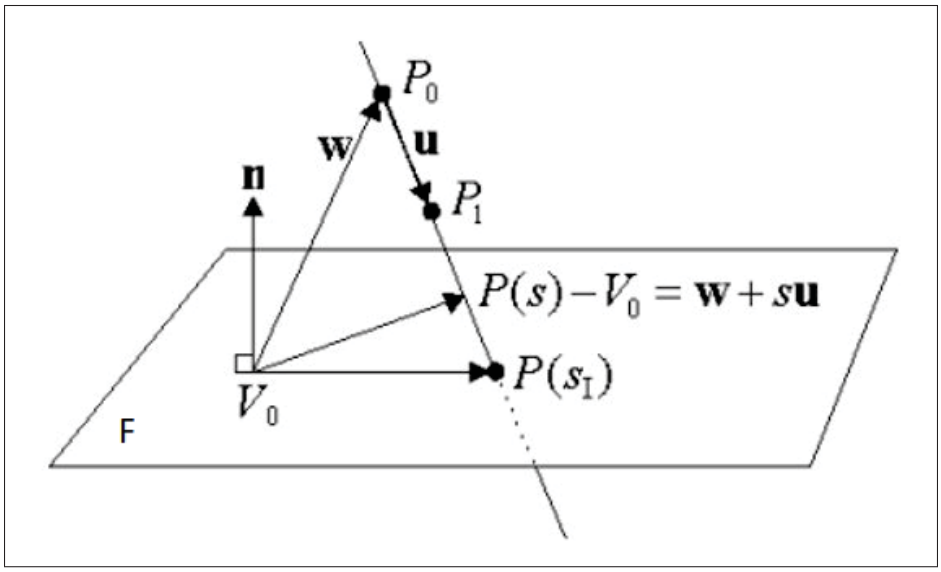

FIGURE 4.20 Illustration de l'intersection ligne-plan adaptée de (Sunday, 2021)

Le vecteur de direction u de la ligne et sa transformation vers le référentiel du monde réel est calculé de la même manière que la section 1.7, P_0 est la position de la caméra en coordonnées local, le vecteur normal n est z=1 et V_0 est [0, 0, Hauteur de l'objet]. Une fois le point d'intersection

trouvé en coordonnées locales, il ne reste plus qu'à faire la projection sur la carte en utilisant la transformée inverse de EPSG :3857 (IOGP, 2019).

La Figure 4.21 montre un exemple de calcul de l'intersection. Dans cet exemple, la position GPS de la voiture était précise et le pixel correspondant au centre de la base du lampadaire a été entré manuellement pour un test de géolocalisation de la base du lampadaire. Cela est possible car nous disposons de la position GPS du lampadaire. En revanche, il est difficile de valider la position GPS trouvée pour un nid de poule, car nous ne disposons pas de leur position réelle et ne pouvons donc pas les repérer sur Google Maps pour les tester. Le point bleu représente la caméra, le point rouge représente la voiture et le point noir représente la position réelle du lampadaire.

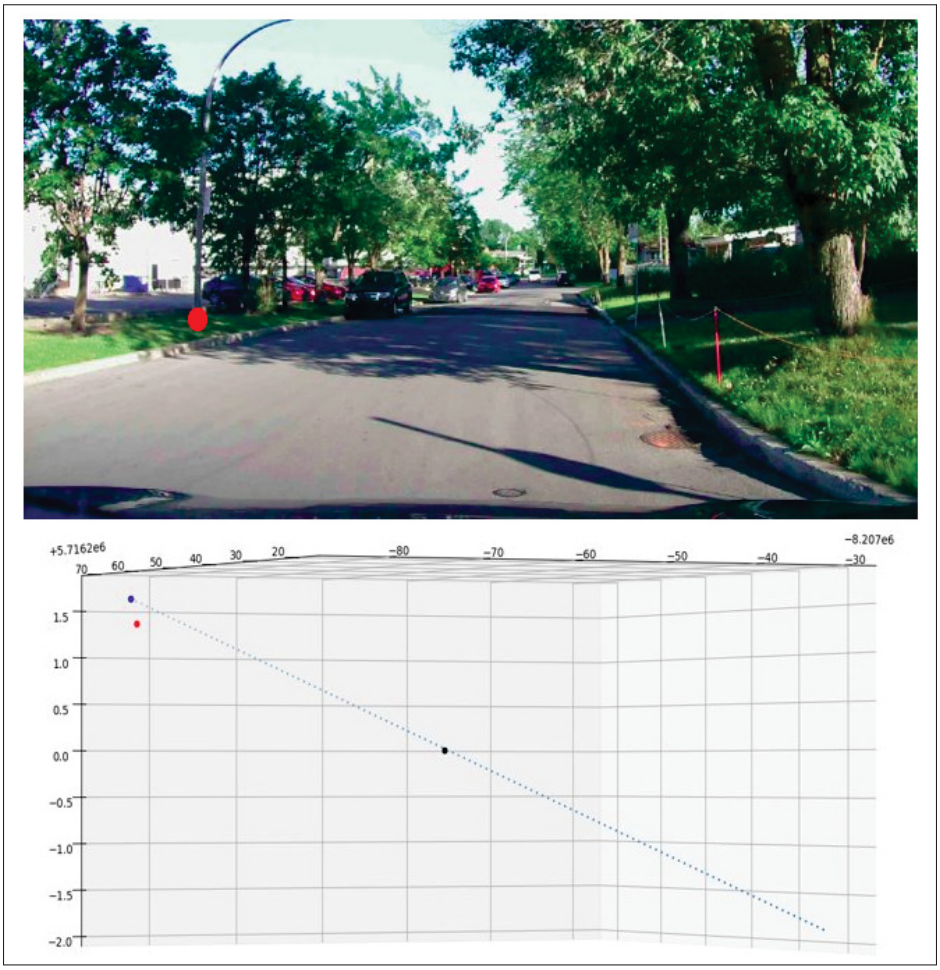

FIGURE 4.21 Exemple de calcul de l'intersection ligne-plan

4.10 Conclusion

Dans ce chapitre, nous avons exploré les algorithmes développés pour géolocaliser un objet détecté et projeter un objet dont la position GPS est connue sur le plan d'image. Au chapitre suivant, nous examinerons les résultats expérimentaux.

CHAPITRE 5

RÉSULTATS EXPÉRIMENTAUX

5.1 Introduction

Dans ce chapitre, nous allons voir les résultats de validation et d'expérimentation des algorithmes présentés dans les sections précédentes. D'abord nous allons voir les spécifications qui ont été implémentées dans le système, ensuite, nous allons voir les résultats de validation et d'expérimentation de l'algorithme de l'estimation de la position GPS, nous discuterons les limitations trouvées par la suite.

Dans un second temps, nous allons voir les résultats expérimentaux de la projection sur le plan d'image d'un objet dont la position GPS est connue. À la fin nous allons voir les résultats de validation de l'algorithme de l'estimation de la position avec une seule détection.

5.2 L'implémentation de l'algorithme de la géolocalisation

Le système proposé est divisé en trois parties :

- Détecteur d'objet.
- Suiveur d'objet.
- Estimateur de la position GPS d'un objet.

Chaque partie de ce système représente un noeud sur Robot Operating system (ROS) (Quigley *et al.*, 2009). ROS est un système d'exploitation libre et ouvert conçu spécifiquement pour les robots. Il propose une infrastructure de communication pour les différents éléments d'un robot. Il fournit une infrastructure de réseau qui permet la communication entre différents nœuds (processus). Ces nœuds peuvent échanger des messages, des commandes et des données (voir Figure 5.1).

Dans notre système, le premier noeud reçoit les images, il détecte les objets présents sur chaque image. Il envoie les détections par la suite au noeud du suiveur (tracker). Ce dernier associe

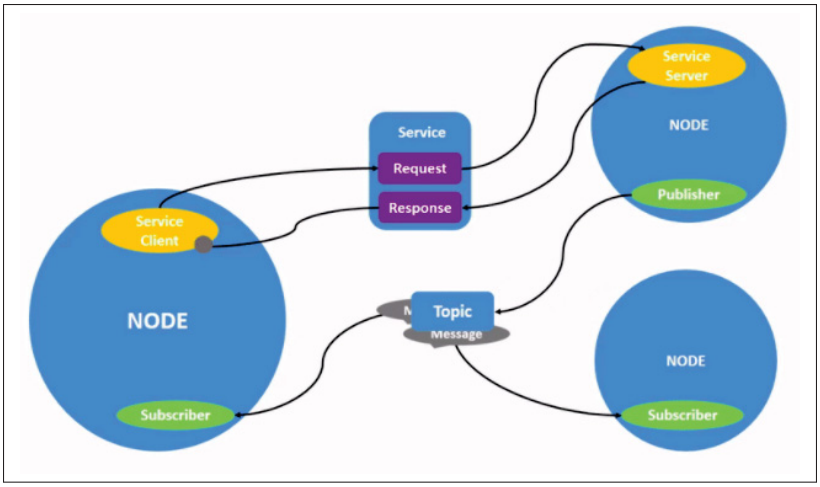

FIGURE 5.1 Illustration de Robot Operating System (Robotics, 2021)

les détections reçues à chaque objet suivi. Une fois un objet est suivi sur une séquence, sa liste détection sera envoyée finalement au noeud de l'estimation de la position, qui reçoit aussi la position GPS et l'orientation de la voiture ainsi que les paramètres de la caméra. Ce dernier noeud calcule la position GPS de l'objet. Les initialisations des paramètres du système sont dans le tableau 5.1, les paramètres présentés dans ce tableau ont été utilisés dans le cas de la géolocalisation des feux de circulation. Les décalages et les angles de la caméra ont été mesurés manuellement.

TABLEAU 5.1 Notre initialisation des paramètres

Détecteur	**Valeur**	**Suiveur**	**Valeur**	**Estimateur**	**Valeur**
batch size	5 frames	Distance function	Euclidean	gps sensor height	1.36 m
inference size	960 pixel	Distance threshold	150 m	camera roll	0
confidence threshold	0.5	Hit inertia min	10	camera pitch	2.45
model	yolov5m6	Hit inertia max	30	camera yaw	0
/	/	Initialization delay	4 frames	cam gps offset x	-0.22 m
/	/	Past detections length	16	cam gps offset y	-0.73 m
/	/	/	/	cam gps offset z	0.27 m

5.3 Validation de l'algorithme de la géolocalisation

L'algorithme de géolocalisation a été testé sur une base de données limitée comprenant 18 feux de circulation. Pour chaque feu, une liste de détections correspondantes a été obtenue en le suivant à travers une séquence d'images. Pour la validation, nous avons vérifié que nous disposions d'au moins 8 détections pour chaque objet, que la vitesse de la voiture était moyenne (40 - 50 Km/h) et que la précision du GPS était à un mètre près. Il est à noter que les positions réelles de ces feux de circulation ont été prises sur Google maps.

La vérification de l'algorithme sur ces feux de circulation a donné une précision moyenne de 4,5 mètres d'erreur pour l'ensemble des données. La Figure 5.2 illustre un exemple de géolocalisation de 3 feux de circulation, où les intersections des lignes d'orientation calculées pour chacun des feux sont représentées par des marqueurs rouges. Dans ce cas, l'erreur moyenne est de 3,2 mètres. Visuellement, nous voyons que le le marqueur indique la bonne zone, mais avec un petit décalage avec la position de l'objet prise de Google maps.

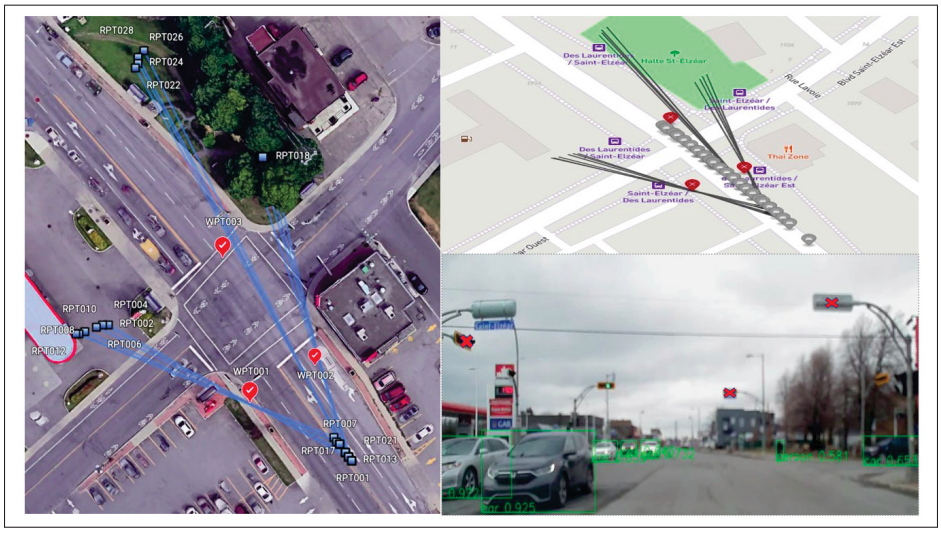

FIGURE 5.2 Exemple de la géolocalisation des feux de circulation

5.4 Résultats expérimentaux de l'algorithme de la géolocalisation

Le système a été implanté dans les voitures du MTQ afin de détecter et localiser différents types d'anomalies routières. Notre analyse des résultats se concentrera sur la localisation des lampadaires défectueux, car nous disposons d'une base de données contenant les positions GPS réelles de ces derniers, ce qui nous permettra de comparer les positions trouvées par notre algorithme. Un lampadaire défectueux peut être détecté plusieurs fois par différentes voitures à différentes dates et heures. À chaque fois qu'un lampadaire est détecté, sa position GPS est estimée. Dans ce cas, nous constatons que les positions estimées sont dispersées autour du lampadaire détecté (voir Figure 5.3).

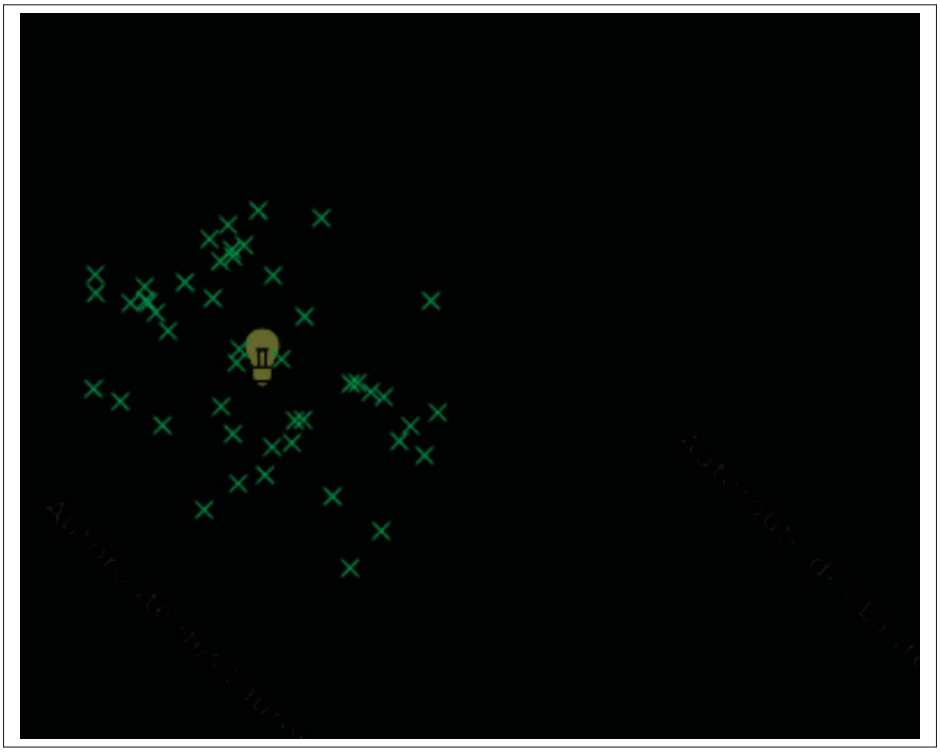

FIGURE 5.3 Illustration de l'estimation de la position GPS du lampadaire

Dans la figure, la lampe jaune représente la position réelle de la base du lampadaire et les croix vertes représentent les positions estimées des deux lampes défectueuses (voir les 2 rectangles

rouges sur la Figure 5.4). Pour les types de lampadaires où la lampe n'est pas montée directement au-dessus de la base, nous nous attendons à ce que la différence entre la position GPS réelle de la base du lampadaire et la position estimée soit d'environ 1,5 mètre de décalage.

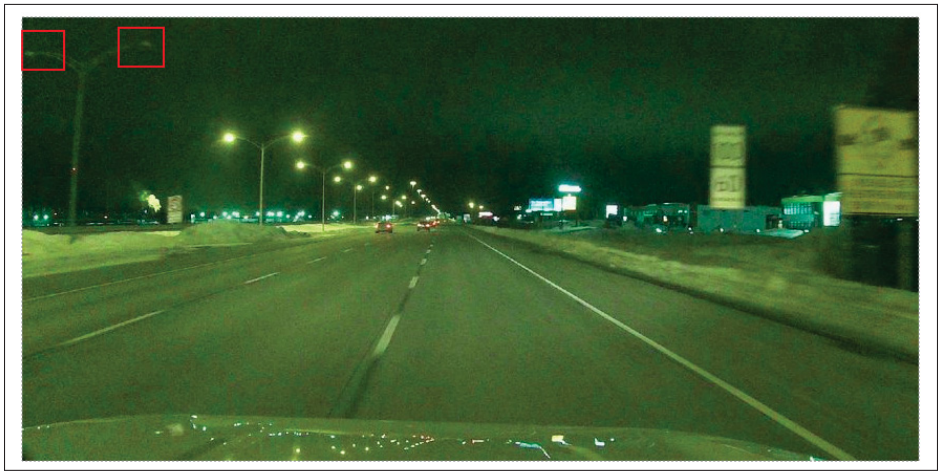

FIGURE 5.4 Les deux lampes défectueuses

Pour l'exemple présenté sur la Figure 5.3, l'erreur de chaque estimation de la position GPS du lampadaire varie entre 2 et 15 mètres. Afin d'analyser la distribution des estimations autour des lampadaires et avoir une estimation de notre exactitude et précision, nous allons analyser la situation suivante : nous avons 5 lampadaires consécutifs où les lampadaires 1, 3 et 5 sont défectueux, tandis que les lampadaires 2 et 4 sont en bon état (voir Figure 5.5).

Sur l'ensemble de ces lampadaires, nous avons 155 positions estimées. La Figure 5.6 montre la distribution des ces points (points noirs) autour des lampadaires où les cercles vert, bleu et rouge représentent les cercles autour des lampadaires de rayon 15, 20, 35 mètres respectivement, les marqueurs en rouge sont les positions réelles des lampadaires.

Dans ce cas, nous constatons que 75% des points correspondant à chaque lampadaire se trouvent dans le cercle de 15 mètres, 88% se trouvent dans le cercle de 20 mètres et 97% se trouvent dans le cercle de 35 mètres. Cependant, à une distance de 35 mètres, il est possible que le point soit plus proche du lampadaire suivant que du lampadaire en question, d'où le risque d'associer l'estimation au mauvais objet.

FIGURE 5.5 Exemple de cas de lampadaires analysés

FIGURE 5.6 Distribution des points du cas étudié

Ainsi :

- Le rayon du cercle qui couvre 50% des points est 10.5 mètres.

- Le rayon du cercle qui couvre 75% des points est 15 mètres.
- Le rayon du cercle qui couvre 95% des points est 32.5 mètres.

Les résultats collectés au cours des deux derniers mois pour l'estimation de la position des lampadaires défectueux ne prennent en considération que les estimations qui se trouvent dans le cercle de 20 mètres d'un lampadaire. Autrement dit, si une estimation tombe dans le cercle de 20 mètres d'un lampadaire, cette estimation sera associée à ce lampadaire. Cependant, les données d'entrée de notre algorithme ne sont pas sauvegardées, ce qui rend impossible l'analyse de l'influence de certains paramètres, tels que le nombre de détections sortant du seuil qui a été attribué à l'estimateur. Donc, il n'est pas possible de regarder de plus près toutes les raisons pour lesquelles nous avons obtenu de grandes erreurs dans certains cas.

En revanche, nous avons obtenu 22 200 estimations associées à des lampadaires sur l'ensemble des résultats. Pour l'analyse de ces derniers, nous avons retenu uniquement les lampadaires qui ont eu plus de 10 estimations au cours de la période des deux derniers mois, c'est qui revient à 9000 positions estimées.

La Figure 5.7 montre les erreurs trouvées pour l'ensemble des 510 lampadaires. Les trois graphes représentent la distance entre la position GPS réelle du lampadaire et la position trouvée. Le graphe bleu correspond à la distance entre la position réelle et la position estimée la plus éloignée appartenant au lampadaire, le graphe rouge représente la distance entre la position réelle et la valeur estimée la plus proche, et le graphe vert est la moyenne des estimations sur l'ensemble des lampadaires. Il est à noter que la position réelle fournie correspond à la position du poteau, mais les estimations ont évalué la position des lampes défectueuses. Ce décalage n'a pas été pris en compte car il est différent d'un type de lampadaire à un autre. L'erreur moyenne trouvée sur l'ensemble était de 11.5 mètres. Nous avons calculé le centre de nuage de points autour des lampadaires (comme l'exemple présenté dans la Figure 5.3) et l'avons comparé avec la position réelle. La Figure 5.8 montre le graphe qui représente la distance entre la position réelle et le centre de nuage autour de chaque poteau.

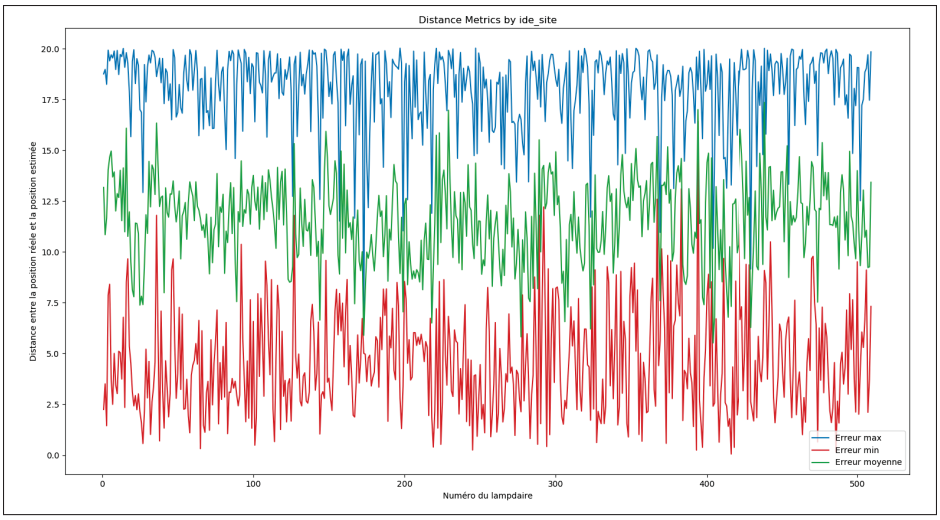

FIGURE 5.7 Graphe de l'erreur d'estimation de la position

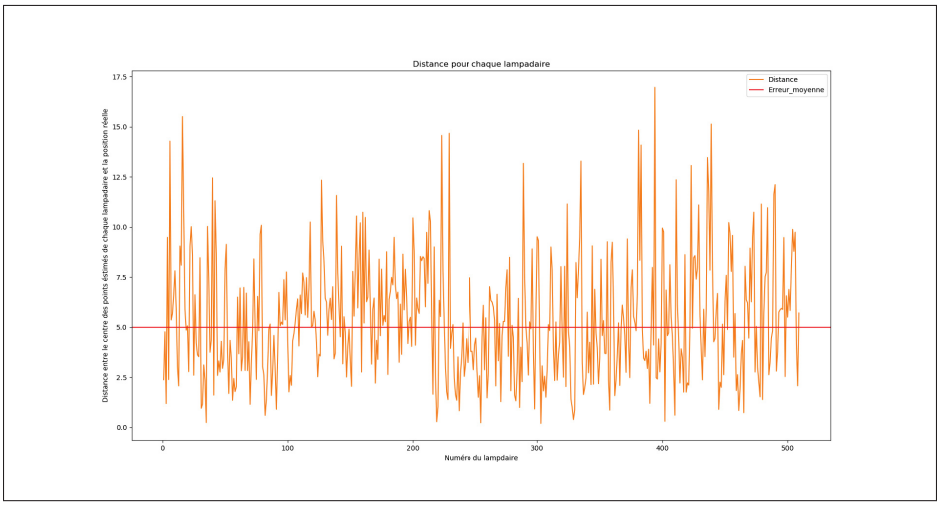

FIGURE 5.8 Distance entre le centre des points estimés de chaque lampadaire et la position réelle

L'erreur entre les centres des nuages autour de chaque lampadaire et leur position réelle est de 5 mètres. Cela peut s'expliquer par le fait qu'il y a plusieurs voitures qui passent avec des calibrations de caméra différentes et parfois même avec une calibration de GPS différente (si ce GPS n'était pas encore configuré avec notre configuration), ce qui entraîne une précision de mesure différente. Nous constatons que nos mesures sont globalement proches de la réalité, mais

elles sont imprécises et instables, car elles varient considérablement d'une mesure à l'autre, même si en moyenne, elles sont relativement correctes.

5.5 Discussion des résultats

Les tests de notre algorithme ont permis de souligner quelques limites éventuelles. Bien que notre algorithme ait montré une certaine exactitude dans la géolocalisation des lampadaires défectueux, il existe certaines limites et problèmes qui causent les imprécisions trouvées et qui méritent d'être abordés.

La première limitation est que l'algorithme ne donne pas une estimation valide lorsque le véhicule est stationnaire. En effet, dans ce cas, le résultat du suiveur (tracker) sera une liste de détections qui pointent presque vers le même emplacement sur l'image. Cela peut conduire à générer des lignes d'orientation qui se croisent au niveau de la position du véhicule, car il est stationnaire et représente donc le point commun entre toutes les lignes. De plus, lorsque le véhicule reste stationnaire pendant plusieurs détections consécutives d'un objet, cela peut grandement influencer le calcul de l'intersection entre les points.

L'exemple de la figure 5.9 montre une situation où la voiture était stationnaire au début du suivi d'un feu de circulation, puis d'autres détections ont été effectuées lors de son déplacement. Les points rouges représentent la voiture, les points blues représentent la caméra. Pour cet exemple, l'erreur est de 31 mètres. Le point trouvé est très proche de la voiture, car c'est le point où le plus grand nombre de lignes se croisent. Dans la figure, les deux figures montrent le même cas, mais avec un angle de vu diffèrent.

Un deuxième facteur qui peut fausser l'estimation est le changement de la forme de la boîte englobante. Cela peut arriver lorsque l'objet est détecté sous différents angles, étant donné que les objets sont en 3D, mais aussi lorsque l'objet arrive aux bords de champ de vision de la caméra. Dans ce dernier cas, seule une partie de l'objet sera détectée, ce qui implique que le centre de l'objet ne sera pas le même pour toutes les images de la séquence. L'exemple du changement de la forme d'un feu de circulation est présenté dans la Figure 5.10. Le feu de circulation était

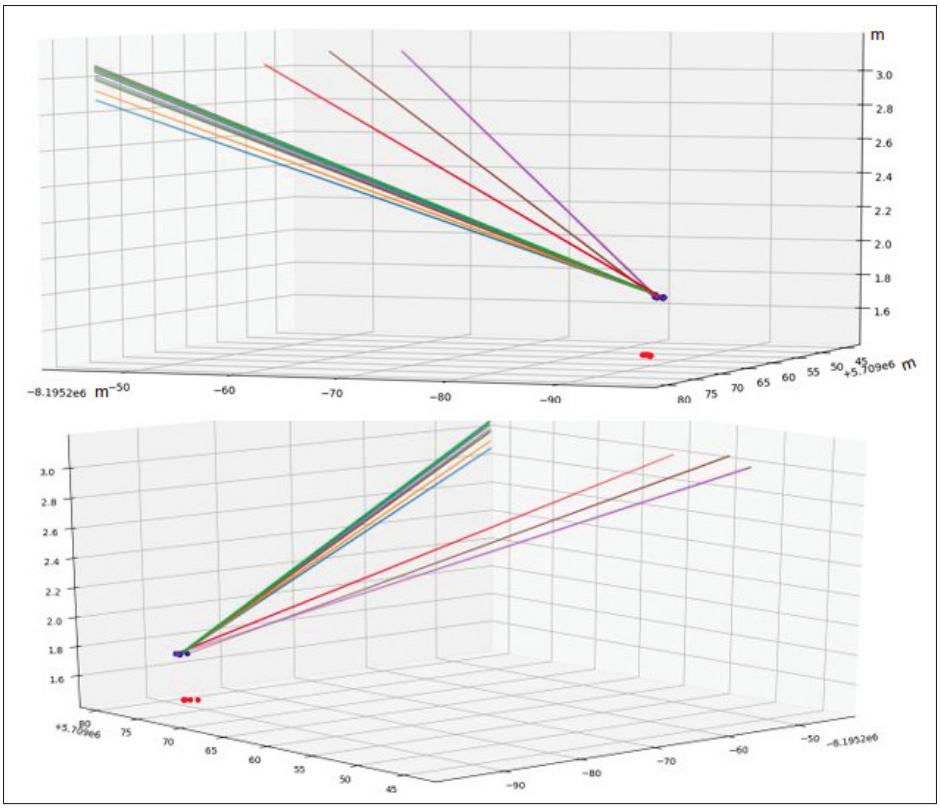

FIGURE 5.9 Exemple de la géolocalisation par un véhicule stationnaire de deux point de vue différents

FIGURE 5.10 Exemple du changement de la forme de la boite englobante

entièrement détecté au début de la séquence, mais plus il sort du champ de vision, plus le centre de l'objet (représenté par le point rouge) pointe vers un autre emplacement de l'objet.

Les résultats peuvent être influencés par la calibration des angles de la caméra. La Figure 5.11 illustre le changement d'erreur en fonction du changement du pitch pour le même ensemble de feux de circulation. Une mauvaise mesure de ces paramètres de la caméra (pitch, roll, yaw) peut augmenter significativement l'erreur, comme le montre le graphique où un changement de 2 degrés du pitch peut induire une erreur de 1,2 mètres.

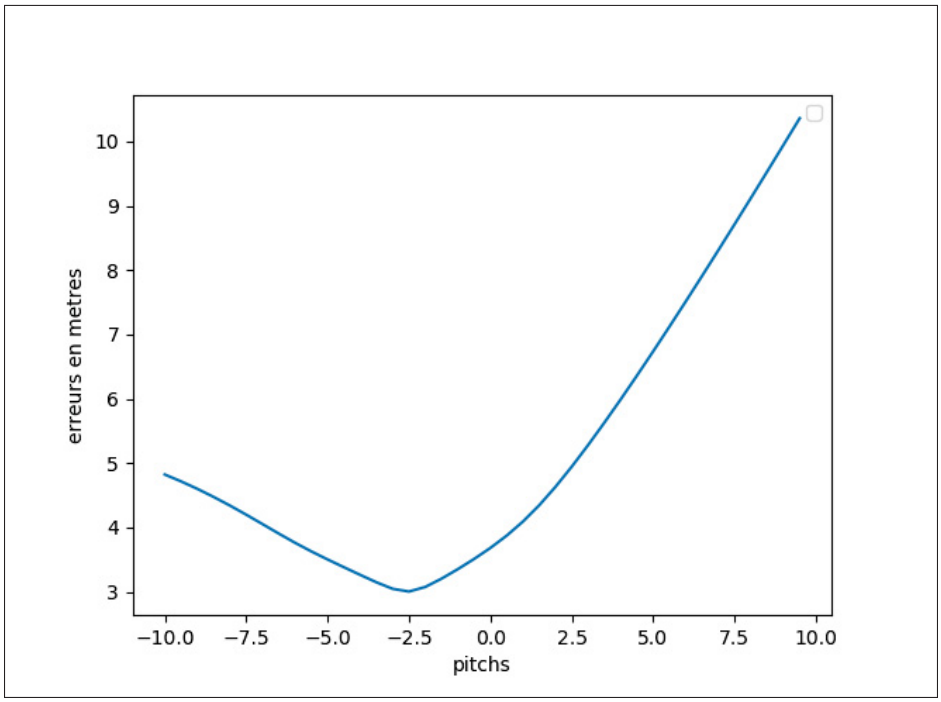

FIGURE 5.11 La variation d'erreur de la géolocalisation en fonction du pitch

Un autre facteur qui peut entraîner des erreurs est une fréquence GPS trop basse. Cela signifie que l'estimation des positions et des orientations du véhicule au moment des détections par une interpolation linéaire entre deux mesures GPS séparées d'une seconde peut donner des estimations fausses lorsque le véhicule circule à grande vitesse. Cela peut arriver si la configuration du GPS n'a pas été effectuée avec succès, car à la base la configuration permet au GPS de fonctionner à 10 Hz.

Étant donné que la position GPS, l'orientation de la voiture et les détections sont les entrées de cet algorithme, une mesure erronée de l'un de ces facteurs fausse les résultats du calcul. La Figure 5.12 présente un exemple où une erreur de mesure de l'orientation était présente, indiquée par les lignes vertes. Nous pouvons observer que l'orientation change considérablement d'une position à l'autre, bien que la voiture se déplace en ligne droite. Dans ce cas, la transformation du vecteur du référentiel de la voiture vers le référentiel du monde réel sera erronée, car l'angle mesuré est faux, ce qui implique un mauvais résultat de positionnement.

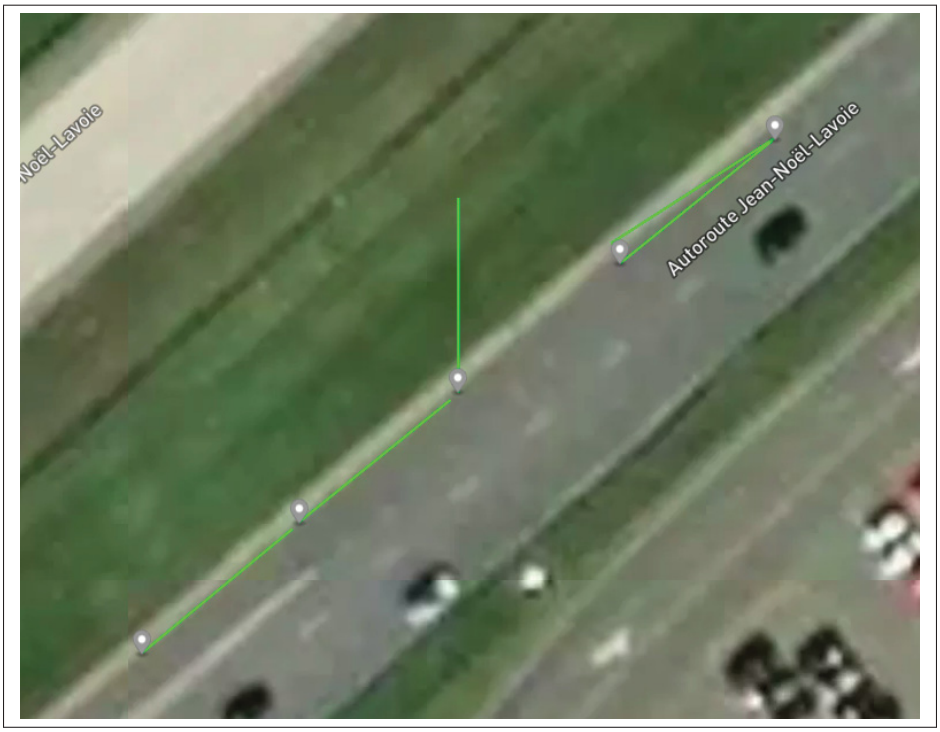

FIGURE 5.12 Exemple de la fausse mesure de l'orientation

Dernièrement, dans certains cas, de fausses mesures ont été trouvées dans la base de données des lampadaires utilisée, ce qui peut conduire à une association incorrecte des estimations à des positions erronées (voir Figure 5.13).

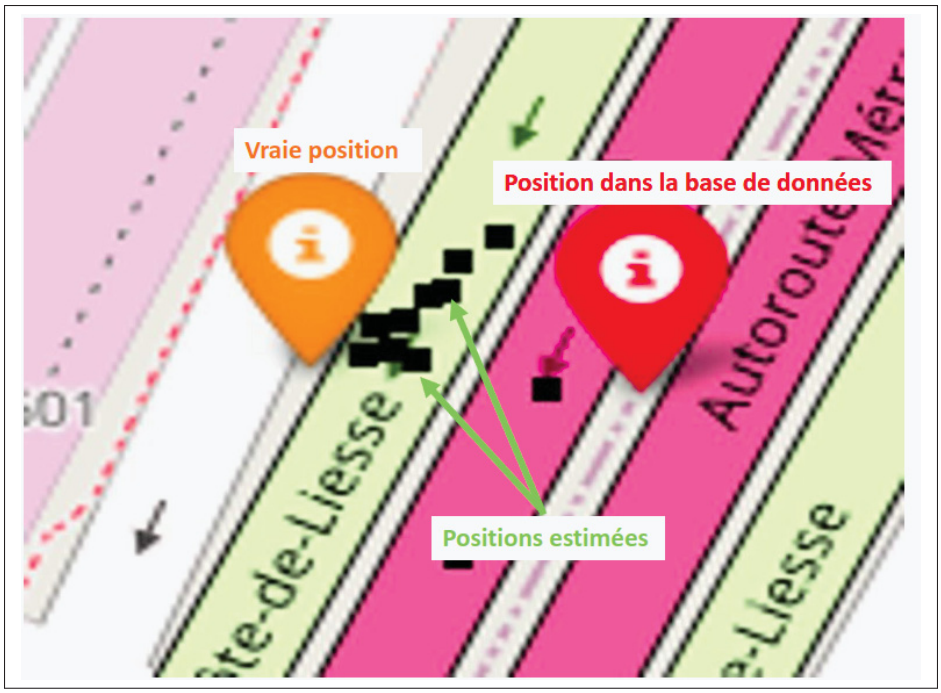

FIGURE 5.13 Exemple d'une fausse mesure dans la base de donnée fournie

5.6 Validation de l'algorithme de la projection

Pour valider l'algorithme de projection d'un objet dont nous possédons la position GPS sur le plan d'image, nous avons suivi la procédure suivante : Nous avons collecté 250 vidéos enregistrées entre les mois de janvier et février. Pour chaque vidéo, nous avons récupéré les données de position GPS et d'orientation de la voiture. Nous avons ensuite projeté les lampadaires se trouvant à proximité de la voiture sur le plan d'image. Cette étape a été répétée pour chaque vidéo.

Ensuite, pour chaque cas, nous avons sauvegardé les images avec une distance de 20, 40 et 70 mètres entre le lampadaire et la voiture. Nous avons ensuite comparé les coordonnées des pixels réels correspondant à l'objet avec les coordonnées des pixels projetés par l'algorithme. L'erreur est calculée comme suit :

$$Erreur = (x_{reel} - x_{projection}, y_{reel} - y_{projection})$$

TABLEAU 5.2 Erreur de projection

Distance voiture-objet	erreur
20 mètres	(82,21) pixels
40 mètres	(80,20) pixels
70 mètres	(88,18) pixels

Nous constatons que l'erreur pour les différentes distances est presque similaire. La Figure 5.14 montre un exemple de projection du même objet pris à différentes distances, où le point bleu est le résultat de l'algorithme et le point rouge est la position réelle.

FIGURE 5.14 Exemple de la projection sur le plan d'image

Cet algorithme peut être affecté par les mêmes facteurs que l'algorithme de géolocalisation, étant donné que ses entrées sont les positions GPS et les orientations de la voiture, ainsi que la base de données d'un objet d'intérêt. Par conséquent, une fausse mesure de la position GPS réelle peut induire des erreurs de projection.

5.7 Validation de l'algorithme de la géolocalisation face à une insuffisance de détection

Cet algorithme est principalement conçu pour géolocaliser les nids de poule. Cependant, la validation de ses résultats est un peu difficile car la position exacte des nids de poule n'est pas connue, ce qui rend impossible une comparaison avec des mesures réelles. Pour valider l'efficacité de l'algorithme, nous avons mis en place la procédure suivante : nous avons sélectionné cinq cas où les mêmes nids de poule ont été détectés plusieurs fois sur une séquence d'images.

Pour chaque nid de poule, nous avons vérifié si la position GPS estimée était la même pour chaque détection dans la séquence d'images afin d'analyser la reproductibilité de l'algorithme. La Figure 5.15 montre les 5 nids de poule étudiés, où les marqueurs rouges représentent le centre de toutes les positions estimées pour chaque nid de poule, et les cercles bleus représentent les cercles qui couvrent tous les points pour chaque nid de poule. En moyenne, les détections sont réparties dans des cercles d'un rayon de 3,5 mètres.

Le graphe de la Figure 5.16 montre l'écart type des positions estimées par rapport au centre des cercles. Cependant, il est important de noter que le nombre d'échantillons est limité, ce qui peut affecter la représentativité des résultats. De plus, il convient de souligner que l'algorithme n'a pas encore été implémenté sur les voitures, d'où la rareté des données disponibles pour cette étude.

Néanmoins, l'écart type moyen sur l'ensemble est de 1,2 mètre, ce qui est relativement faible par rapport à la plage des distances entre le centre des positions estimées et chaque estimation du même nid de poule. Cela peut être considéré comme un bon indicateur de reproductibilité.

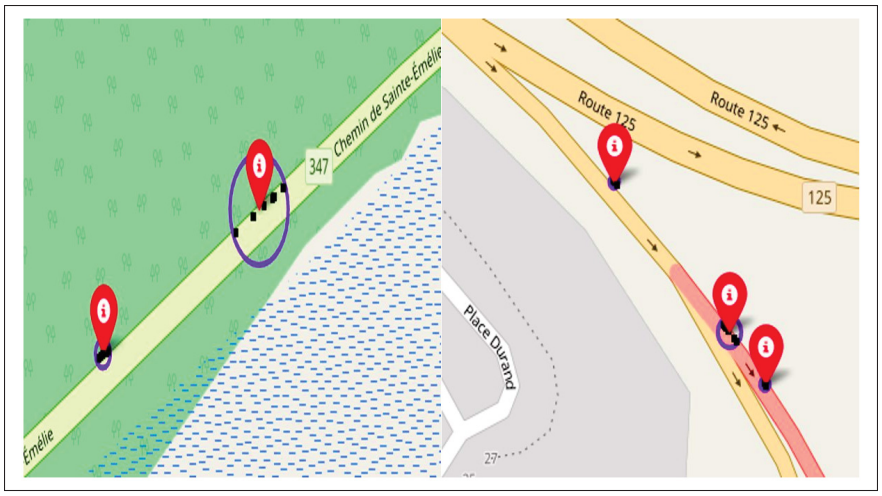

FIGURE 5.15 La projection de la position des nids de poule étudiés

Cependant, il est nécessaire de recueillir davantage de données pour parvenir à des conclusions plus robustes.

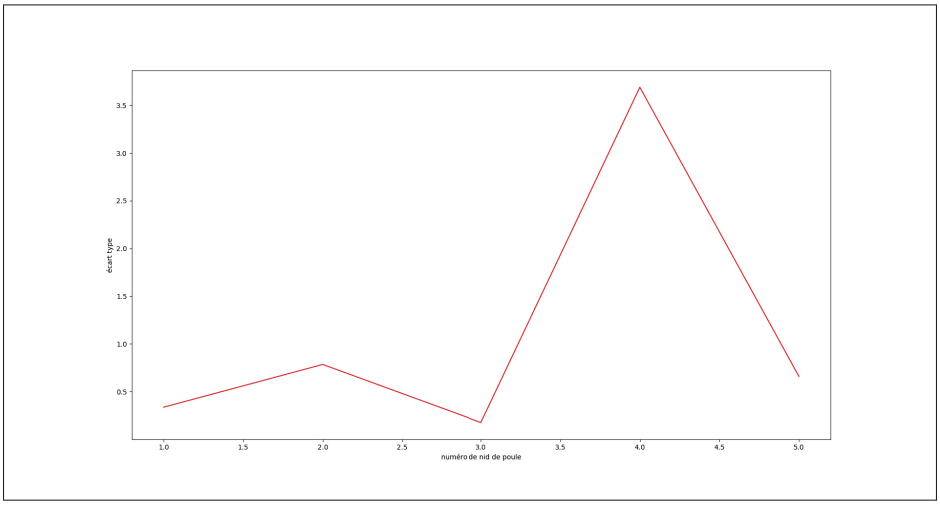

FIGURE 5.16 Écart type par rapport au centre des estimation

5.8 Conclusion

Dans ce chapitre, nous avons examiné les résultats expérimentaux de l'algorithme de géolocalisation des objets détectés en validant son fonctionnement sur la localisation des feux de circulation. Ensuite, nous avons étudié le cas de la géolocalisation des lampadaires défectueux. Nous avons validé le fonctionnement de l'algorithme de projection sur le plan d'image en utilisant une base de données de 250 vidéos, avec une base de données contenant les positions GPS des lampadaires de la MTQ. Enfin, nous avons essayé d'analyser la reproductibilité de l'algorithme de géolocalisation en cas de détection insuffisante, en utilisant les échantillons disponibles.

CONCLUSION ET RECOMMANDATIONS

Dans ce mémoire, nous avons abordé la problématique de la géolocalisation des objets détectés par une caméra monoculaire montée sur un véhicule, afin d'automatiser le processus de la localisation GPS des anomalies routières telles que les lampadaires défectueux et les nids de poule, tout en respectant les limites des capacités de calcul de notre système.

L'objectif principal de ce projet était de cartographier les routes de la province du Québec en trouvant les positions GPS des objets d'intérêt, afin de créer une base de données qui pourrait être exploitée pour diverses applications, telles que l'automatisation du processus de repérage des anomalies routières, la fourniture d'informations aux véhicules autonomes pour leur montrer où se trouvent les anomalies, telles que les nids de poule ou les feux de circulation, ou encore pour les chercheurs qui travaillent sur ce sujet. Le système repose sur l'utilisation d'une caméra monoculaire, qui est moins coûteuse que d'autres technologies, telles que le LIDAR.

Pour atteindre notre objectif, nous avons combiné trois algorithmes : un détecteur d'objets basé sur les concepts de la vision par ordinateur afin de localiser l'objet sur l'image, un suiveur (tracker) pour retracer le même objet sur une séquence d'images prises sous différents angles, et un algorithme basé sur les lignes d'orientation afin de déterminer les coordonnées 3D d'un objet dans le monde réel. Nous avons choisi cet algorithme en particulier pour garantir une exécution rapide qui ne nécessite pas une grande capacité de calcul.

L'algorithme développé a été testé en utilisant une base de données fournie par le MTQ contenant les positions GPS réelles des lampadaires. Nous avons comparé les estimations de l'algorithme et avons constaté que 88% des positions estimées se trouvaient dans un rayon de 20 mètres pour l'échantillon étudié. Toutefois, nous avons également observé que nos estimations étaient imprécises et instables dans certains cas. Ensuite, nous avons donc testé l'algorithme de projection sur 250 cas pour valider son fonctionnement où l'erreur était de (80,20) pixels. Finalement, nous avons abordé la reproductibilité de l'algorithme de géolocalisation face à une

insuffisance de détection en prenant des exemples de nids de poule. Cependant, nous n'avons pas encore validé son utilisation sur une grande base de données car il n'a pas été mis en production pour le moment.

Malgré les inexactitudes constatées dans certains cas, les résultats globaux ont été très encourageants. Des améliorations sont encore possibles pour renforcer la performance de l'algorithme, notamment en calibrant la pose de la caméra à l'aide de concepts tels que l'odométrie visuelle. Il est également possible d'améliorer la configuration GPS en ajustant les filtres proposés par ce dernier pour réduire les perturbations et améliorer la précision des mesures de position. Il est également nécessaire d'apporter des améliorations au niveau du suiveur (tracker) en ce qui concerne la sélection des données, afin d'éviter les problèmes résultant des variations de forme de la boîte englobante.

De plus, d'autres approches peuvent être considérées pour les futures recherches, notamment SLAM-ORB et Structure From Motion. En effet, ces approches ont montré de bonnes performances en ce qui concerne la reconstruction des scènes en 3D. Ainsi, il est possible de les combiner avec les positions GPS de la voiture pour retrouver les positions des objets. Il serait également intéressant de travailler sur la réduction du temps d'exécution en utilisant une méthode plus rapide pour la détection des points clés correspondant à chaque objet en les combinant avec des méthodes de détection plus rapide que SIFT et ORB.

ANNEXE I

ARCHITECTURE DE YOLOV5

1. Réseaux de neurones convolutifs

Les réseaux de neurones convolutifs sont une architecture en apprentissage profond, qui est très puissante dans la reconnaissance de caractéristiques des objets comme les contours, les textures et les couleurs présentent sur une image. Ils sont basés sur le glissement des filtres sur une image donnée afin d'extraire des caractéristiques. Leur principe de fonctionnement se résume comme suit (voir Figure I-1) :

- Pré-traitement des images en entrée afin de les adapter au modèle utilisé à travers différentes opérations comme le redimensionnement ou la normalisation .
- Détection des caractéristiques : cela consiste à déterminer les caractéristiques qui définissent un objet en faisant l'opération de convolution entre l'image et un ensemble de filtres. Ces filtres sont variés et chacun permet de détecter une caractéristique différente comme la détection des coins.
- Chaque filtre (aussi appelé noyau) se déplace sur l'image de gauche à droite et de haut en bas, en effectuant un produit de convolution à chaque étape. Le résultat de cette opération est une carte d'activation qui donne la réponse du filtre à chaque position de l'image (voir Figure I-2). L'ensemble des cartes d'activation produites par différents filtres appliqués à l'image d'entrée appelée la carte de caractéristiques.
- Après avoir passé tous les filtres sur l'image, une couche de non-linéarité ou une fonction d'activation est utilisée. Cette fonction permet d'enlever la linéarité et d'introduire des relations non linéaires, ce qui peut améliorer la capacité du réseau à apprendre et à généraliser. La fonction d'activation la plus utilisée est RELU qui consiste simplement à remplacer tous les négatifs par 0.

$$Relu(x) = max(0, x) \tag{A I-1}$$

- Ensuite, des couches de pooling sont utilisées entre les couches de convolution. Elles servent à réduire la dimension spatiale de l'entrée. Elles fonctionnent en sélectionnant les plus fortes

activations dans une région donnée et en les agrégeant en une seule valeur. Cela peut être fait de différentes manières, comme la moyenne ou le maximum (voir Figure I-3). La couche de pooling est généralement utilisée pour réduire le sur-apprentissage et accélérer le temps de calcul.

- Finalement, une couche entièrement connectée est utilisée pour effectuer la classification finale. Cette couche prend en entrée un vecteur de caractéristiques extraites par les couches de convolution et de pooling précédentes, et produit en sortie une prédiction de classe, en donnant une probabilité de correspondance pour chaque classe en utilisant une fonction comme softmax (équation (1.2)).

$$\sigma(\vec{z})_i = \frac{exp(z_i)}{\sum_{j=0}^{K} exp(z_j)} \qquad \text{(A I-2)}$$

Où z est un vecteur contenant les différentes entrées et K est le nombre des classes.

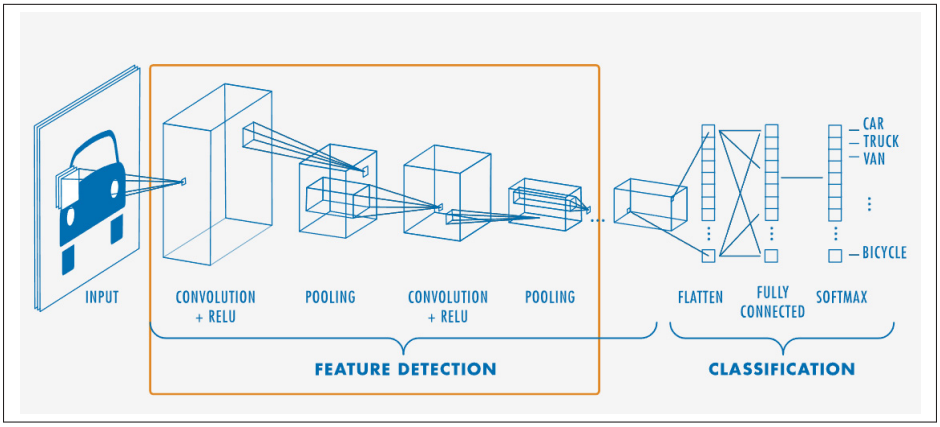

FIGURE-A I-1 Illustration de l'architecture des CNN (Matlab, 2022)

2. Architecture de YOLOv5

La Figure I-4 et I-5 résument les composants principaux de chaque partie de YOLOv5. L'architecture de YOLO se compose généralement de trois parties :

- Backbone : un réseau neuronal convolutionnel qui regroupe et forme des cartes de caractéristiques à partir des images d'entrée

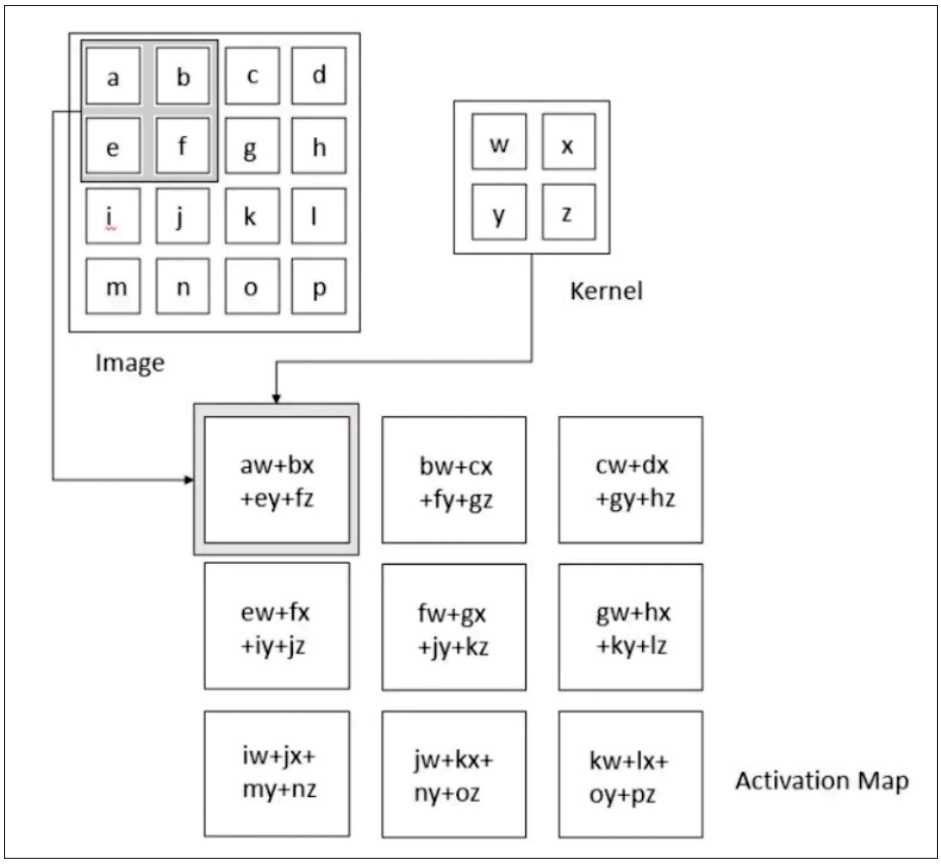

FIGURE-A I-2 Illustration de l'opération de la convolution
(Heaton, 2018)

- Neck : un ensemble de couches qui relient et combinent ces caractéristiques pour les transmettre à la prédiction.
- La tête : la dernière partie du modèle qui utilise les caractéristiques collectées par le Neck pour effectuer les étapes de prédiction de boîtes et de classes

2.1 La couche Focus

La couche Focus a pour but de simplifier les couches, diminuer le nombre de paramètres du modèle, réduire les FLOPS (floating-point operations per second), limiter l'utilisation de la mémoire CUDA (Architecture de dispositif de calcul unifié), améliorer les performances en termes de vitesse d'exécution et cela avec un impact minimal sur le mAP (précision moyenne).

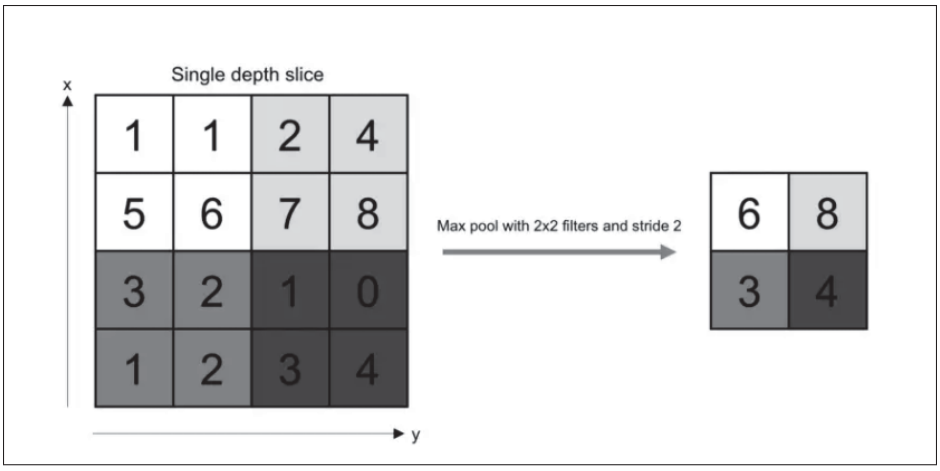

FIGURE-A I-3 Illustration de pooling (Albawi *et al.*, 2017)

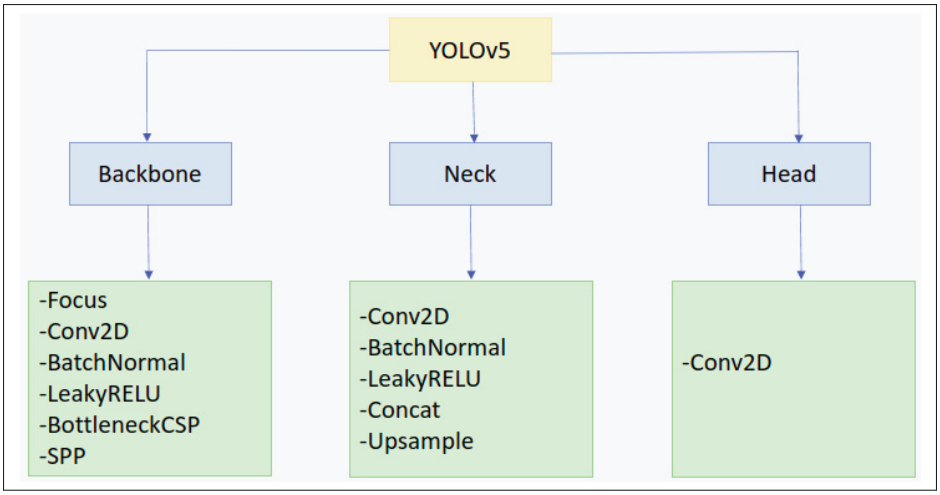

FIGURE-A I-4 Vue d'ensemble sur les composants de YOLOv5

2.1.1 La couche Batch Norm

La normalisation de lot (Batch Norm) est une couche intermédiaire qui est intégrée entre deux couches cachées. Elle a pour rôle de prendre les sorties de la couche précédente, de les normaliser pour ensuite les transmettre comme entrée pour la couche suivante.

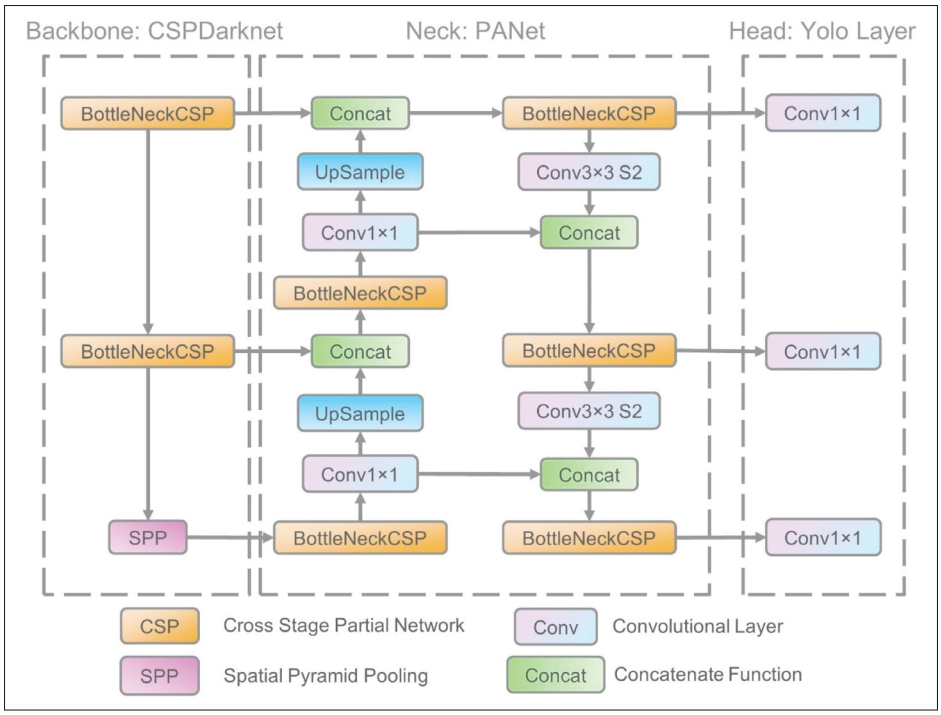

FIGURE-A I-5 Vue d'ensemble de l'architecture de YOLOv5 (Xu *et al.*, 2021)

La normalisation par lots est une technique qui permet d'améliorer la performance et la stabilité d'un modèle en réduisant les variations des données d'entrée dans chaque lot de données utilisé pour l'entraînement. Cela permet notamment de rendre le modèle moins sensible aux réglages des hyperparamètres, de réduire les problèmes de décalage covariant interne et de diminuer l'importance de l'initialisation des poids.

2.1.2 LeakyRELU

La fonction Leaky ReLU est la fonction d'activation utilisée dans ce réseau. Elle est similaire à la fonction ReLU standard, mais elle ne retourne pas zéro quand le signal d'entrée est négatif. Cela évite le problème des neurones "morts" dans les réseaux de neurones en permettant aux neurones de continuer à apprendre même lorsque l'entrée est négative.

$$f(x) = \begin{cases} x & \text{si } x > 0, \\ 0.01x & \text{sinon} \end{cases} \tag{A I-3}$$

2.2 Cross-Stage-Partial-Connections (CSP)

YOLOv5 utilise une architecture basée sur CSP-Darknet53 (Wang *et al.*, 2020). Cette architecture est une version de Darknet53 qui a été modifiée en y intégrant la technique CSP pour améliorer les performances du réseau.

CSPDarknet53 est utilisé pour la détection d'objets. Il utilise une technique appelée CSPNet qui consiste à diviser la carte de caractéristiques de la couche de base en deux sections, puis les fusionne en utilisant une hiérarchie transversale de niveau (voir Figure I-6). Cette méthode permet au réseau d'avoir plus de flux de gradient et plus de rapidité de calcul. La technique CSPNet divise la carte de caractéristiques en deux parties, l'une desquelles est traitée par un bloc dense et une couche de transition, l'autre est intégrée à la carte de caractéristiques de l'étape suivante.

YOLOv5 a apporté deux améliorations importantes à la partie intermédiaire du modèle, appelée "neck". La première est l'utilisation d'une variante de Spatial Pyramid Pooling (SPP) qui permet une meilleure utilisation des informations spatiales. La seconde est la modification de Path Aggregation Network (PANet) (Wang *et al.*, 2019) en intégrant BottleNeckCSP dans son architecture. PANet est principalement utilisé pour améliorer le processus de segmentation d'instances en conservant les informations spatiales. Dans YOLOv5 PANet adopte une nouvelle structure de réseau de pyramide de caractéristiques (FPN) avec un chemin descendant, ce qui permet d'améliorer la propagation des caractéristiques de bas niveau.

2.3 Spatial Pyramid Pooling (SPP)

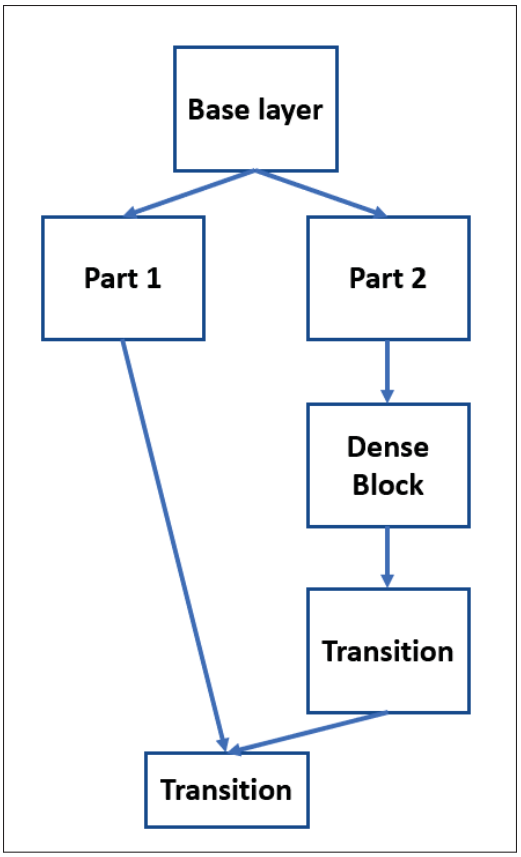

FIGURE-A I-6 L'architecture de partition de CSPDarknet53 (Wang *et al.*, 2020)

La Spatial Pyramid Pooling (SPP) (He *et al.*, 2015) est une technique utilisée pour gérer les images de différentes échelles en divisant l'image en une pyramide de régions à différentes échelles. Les caractéristiques obtenues sont ensuite concaténées et envoyées dans une couche entièrement connectée pour générer la sortie finale. Cela permet au réseau de gérer les variations d'échelle des objets dans l'image. La couche SPP est généralement placée après les couches de convolution et avant les couches entièrement connectées dans l'architecture du réseau.

Le pooling pyramidal spatial maintient les informations spatiales dans des boîtes spatiales locales. Le nombre de boîtes et leur taille sont fixes. Dans chaque boîte spatiale, les résultats de convolution avec chaque filtre sont regroupés. Dans la Figure I-7, un pooling à trois niveaux est effectué.

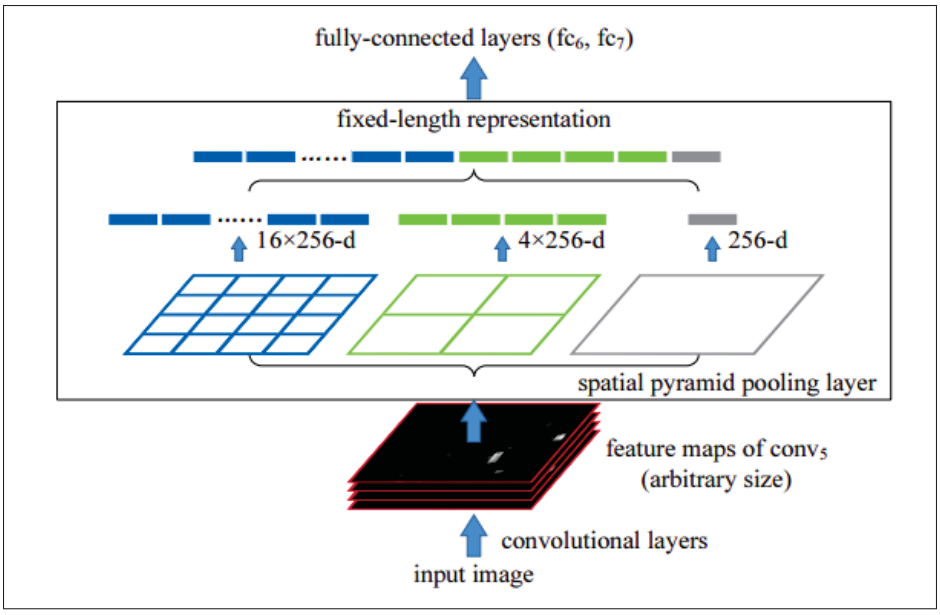

FIGURE-A I-7 Spatial Pyramid Pooling (He *et al.*, 2015)

En ce qui concerne la tête, YOLOv5 utilise trois couches de convolution qui permettent de prédire l'emplacement des boîtes englobantes (x, y, hauteur, largeur) autour des objets détectés, ainsi que les scores de confiance associés à chaque détection et les classes d'objets auxquelles ils appartiennent.

2.4 Fonction d'erreur

La fonction de perte utilisée dans YOLOv5 est composée de plusieurs termes qui permettent de prendre en compte différents aspects de la détection d'objets.

- BCE (Entropie croisée binaire) est utilisée pour calculer la perte des classes, qui mesure la dissimilarité entre les probabilités de classe prédites et les étiquettes de classe réelles. Il permet de mesurer l'erreur de classification.

$$Loss_{class} = -\sum_{j=0}^{K} y_i * log(p_i) + (1 - y_i) * log(1 - p_i) \qquad \text{(A I-4)}$$

où y_i est l'étiquette de classe réelle pour la classe i et p_i est la probabilité prédite pour la classe i.

- La perte d'objet est également calculée en utilisant BCE, qui mesure la dissimilarité entre les scores d'objet prédits et les étiquettes d'objet réelles, Il permet de mesurer l'erreur de la détection.

$$Loss_{obj} = -\sum_{j=0}^{K} o_i * log(p_i) + (1 - o_i) * log(1 - o_i) \quad \text{(A I-5)}$$

où o_i est l'étiquette d'objet réelle pour l'objet i et p_i est le score d'objet prédit pour l'objet i.

- CIoU (Intersection complète sur l'Union) est utilisé pour calculer la perte de localisation. Il mesure la dissimilarité entre les boîtes englobantes prédites et les boîtes englobantes réelles en prenant en compte à la fois la précision de la localisation (par l'intersection) et la sélectivité (par l'union). Il permet de mesurer l'erreur de localisation.

$$L_{loc} = 1 - IoU + (cx_p - cx_t)^2 + (cy_p - cy_t)^2 + (w_p - w_t)^2 + (h_p - h_t)^2 \quad \text{(A I-6)}$$

où cx_p, cy_p,w_p, h_p sont les coordonnées du centre, la largeur et la hauteur de la boîte englobante prédite respectivement. cx_t, cy_t, w_t, h_t sont les coordonnées du centre, la largeur et la hauteur de la vraie boîte englobante respectivement. IoU est l'intersection sur union entre la boîte englobante prédite et la boîte englobante vraie.

La formule finale de la fonction de perte est obtenue en combinant ces différents termes avec des poids appropriés, pour obtenir une mesure globale de la qualité des résultats obtenus par le modèle.

$$Loss = \lambda_1 * Loss_{class} + \lambda_2 * Loss_{obj} + \lambda_3 * Loss_{loc} \quad \text{(A I-7)}$$

BIBLIOGRAPHIE

Afshine Amidi, S. A. (2021). CS 230 - Deep Learning. Repéré le 2022-12-15 à https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks.

Albawi, S., Mohammed, T. A. & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*, pp. 1-6. doi : 10.1109/ICEngTechnol.2017.8308186.

Aldegheri, S., Bombieri, N., Bloisi, D. D. & Farinelli, A. (2019). Data Flow ORB-SLAM for Real-time Performance on Embedded GPU Boards. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5370-5375. doi : 10.1109/IROS40897.2019.8967814.

Alori, J., Descoins, A., Rıos, B. & Castro, A. tryolabs/norfair : v0. 3.1. Repéré le 2022-02-02 à https://zenodo.org/record/7504727#.ZAZWE3bMJD8.

Ardusimple. (2021). rtk-ins-simplertk2b-f9r-hookup-guide. Repéré le 2022-12-15 à https://www.ardusimple.com/rtk-ins-simplertk2b-f9r-hookup-guide/.

Arulampalam, M. S., Maskell, S., Gordon, N. & Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on signal processing*, 50(2), 174–188.

Bay, H., Ess, A., Tuytelaars, T. & Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer vision and image understanding*, 110(3), 346–359.

Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M. (2020). Yolov4 : Optimal speed and accuracy of object detection. *arXiv preprint arXiv :2004.10934*.

Bustos, A. P., Chin, T.-J., Eriksson, A. & Reid, I. (2019). Visual SLAM : Why bundle adjust ? *2019 international conference on robotics and automation (ICRA)*, pp. 2385–2391.

Calonder, M., Lepetit, V., Strecha, C. & Fua, P. (2010). *Brief : Binary robust independent elementary features*. European conference on computer vision.

Campbell, S., O'Mahony, N., Krpalcova, L., Riordan, D., Walsh, J., Murphy, A. & Ryan, C. (2018). Sensor technology in autonomous vehicles : A review. *2018 29th Irish Signals and Systems Conference (ISSC)*, pp. 1–4.

Choi, S., Park, J. & Yu, W. (2013). Resolving scale ambiguity for monocular visual odometry. *2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 604-608. doi : 10.1109/URAI.2013.6677403.

Comaniciu, D., Ramesh, V. & Meer, P. (2000). *Real-time tracking of non-rigid objects using mean shift*. Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662).

Courrieu, P. (2008). Fast computation of Moore-Penrose inverse matrices. *arXiv preprint arXiv :0804.4809*.

Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1, 886-893 vol. 1. doi : 10.1109/CVPR.2005.177.

Deilamsalehy, H. & Havens, T. C. (2016). *Sensor fused three-dimensional localization using IMU, camera and LiDAR*. 2016 IEEE SENSORS.

Du, L., Zhang, R. & Wang, X. *(2020). Overview of two-stage object detection algorithms*. Journal of Physics : Conference Series.

Duane, C. B. (1971). Close-range camera calibration. *Photogramm. Eng*, 37(8), 855–866.

Gao, F., Deng, F., Li, L., Zhang, L., Zhu, J. & Yu, C. (2021). MGG : Monocular Global Geolocation for Outdoor Long-Range Targets. *IEEE Transactions on Image Processing*, 30, 6349–6363.

Gao, H., Cheng, B., Wang, J., Li, K., Zhao, J. & Li, D. (2018). Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment. *IEEE Transactions on Industrial Informatics*, 14(9), 4224–4231.

Ge, Z., Liu, S., Wang, F., Li, Z. & Sun, J. (2021). Yolox : Exceeding yolo series in 2021. *arXiv preprint arXiv :2107.08430*.

Gené-Mola, J., Sanz-Cortiella, R., Rosell-Polo, J. R., Morros, J.-R., Ruiz-Hidalgo, J., Vilaplana, V. & Gregorio, E. (2020a). Fruit detection and 3D location using instance segmentation neural networks and structure-from-motion photogrammetry. *Computers and Electronics in Agriculture*, 169, 105165.

Gené-Mola, J., Sanz-Cortiella, R., Rosell-Polo, J. R., Morros, J.-R., Ruiz-Hidalgo, J., Vilaplana, V. & Gregorio, E. (2020b). Fuji-SfM dataset : a collection of annotated images and point clouds for Fuji apple detection and location using structure-from-motion photogrammetry. *Data in brief*, 30, 105591.

Gevorgyan, Z. (2022). SIoU Loss : More Powerful Learning for Bounding Box Regression. *arXiv preprint arXiv :2205.12740*.

Girshick, R. (2015). Fast r-cnn. *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448.

Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587.

Godard, C., Mac Aodha, O. & Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 270–279.

He, K., Zhang, X., Ren, S. & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9), 1904–1916.

He, K., Gkioxari, G., Dollár, P. & Girshick, R. (2017). Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980-2988. doi : 10.1109/ICCV.2017.322.

Heaton, J. (2018). Ian goodfellow, yoshua bengio, and aaron courville : Deep learning. Springer.

Heller, J., Henrion, D. & Pajdla, T. (2015). Stable radial distortion calibration by polynomial matrix inequalities programming. *Computer Vision–ACCV 2014 : 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part I 12*, pp. 307–321.

International Association of Oil & Gas Producers. (2018). Geomatics Guidance Note 23 – Web Mercator. (373-23).

IOGP. (2019). *Coordinate Conversions and Transformations including Formulas*. Repéré à https://www.iogp.org/wp-content/uploads/2019/09/373-07-02.pdf.

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., NanoCode012, Kwon, Y., Michael, K., TaoXie, Fang, J., imyhxy, Lorna, Yifu), Wong, C., V, A., Montes, D., Wang, Z., Fati, C., Nadar, J., Laughing, UnglvKitDe, Sonck, V., tkianai, yxNONG, Skalski, P., Hogan, A., Nair, D., Strobel, M. & Jain, M. (2021). ultralytics/yolov5 : v7.0 - YOLOv5 SOTA Realtime Instance Segmentation. Repéré le 2023-02-02 à https://doi.org/10.5281/zenodo.7347926.

Jocher, G., Chaurasia, A. & Qiu, J. (2022). YOLO by Ultralytics. Repéré le 2023-02-02 à https://github.com/ultralytics/yolov5.

Juan, L. & Gwun, O. (2009). A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4), 143–152.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

Kandimalla, V. V. (2021). DEEP LEARNING APPROACHES TO CLASSIFY AND TRACK AT-RISK FISH SPECIES.

Karami, E., Prasad, S. & Shehata, M. (2017). Image matching using SIFT, SURF, BRIEF and ORB : performance comparison for distorted images. *arXiv preprint arXiv :1710.02726*.

Kim, H., Liu, B., Goh, C. Y., Lee, S. & Myung, H. (2017). Robust vehicle localization using entropy-weighted particle filter-based data fusion of vertical and road intensity information for a large scale urban area. *IEEE Robotics and Automation Letters*, 2(3), 1518–1524.

Kim, S., Kim, H., Yoo, W. & Huh, K. (2015). Sensor fusion algorithm design in detecting vehicles using laser scanner and stereo vision. *IEEE Transactions on Intelligent Transportation Systems*, 17(4), 1072–1084.

Kozonek, N., Zeller, N., Bock, H. & Pfeifle, M. (2019). ON THE FUSION OF CAMERA AND LIDAR FOR 3D OBJECT DETECTION AND CLASSIFICATION. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.

Krylov, V. A. & Dahyot, R. (2018). Object geolocation using mrf based multi-sensor fusion. *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 2745–2749.

Kuutti, S., Fallah, S., Katsaros, K., Dianati, M., Mccullough, F. & Mouzakitis, A. (2018). A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications. *IEEE Internet of Things Journal*, 5(2), 829–846.

Landau, H., Vollath, U., Chen, X. et al. (2002). Virtual reference station systems. *Journal of global positioning systems*, 1(2), 137–143.

Lee, J.-K., Baik, Y.-K., Cho, H. & Yoo, S. (2020). Online Extrinsic Camera Calibration for Temporally Consistent IPM Using Lane Boundary Observations with a Lane Width Prior. *arXiv preprint arXiv :2008.03722*.

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W. et al. (2022). YOLOv6 : A single-stage object detection framework for industrial applications. *arXiv preprint arXiv :2209.02976*.

Li, C., Li, L., Geng, Y., Jiang, H., Cheng, M., Zhang, B., Ke, Z., Xu, X. & Chu, X. (2023). YOLOv6 v3. 0 : A Full-Scale Reloading. *arXiv preprint arXiv :2301.05586*.

Li, R., Pang, M., Zhao, C., Zhou, G. & Fang, L. (2016). Monocular long-term target following on uavs. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 29–37.

Li, S. Z. (2009). *Markov random field modeling in image analysis*. Springer Science & Business Media.

Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A. & Hengel, A. V. D. (2013). A survey of appearance models in visual object tracking. *ACM transactions on Intelligent Systems and Technology (TIST)*, 4(4), 1–48.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 91–110.

Martins, P. F., Costelha, H., Bento, L. C. & Neves, C. (2020). Monocular Camera Calibration for Autonomous Driving — a comparative study. *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 306-311. doi : 10.1109/ICARSC49921.2020.9096104.

MERN. Réseau géodésique du Québec. Repéré le 2023-02-02 à https://mern.gouv.qc.ca/repertoire-geographique/reseau-geodesique-donnees-gnss/.

Minaeian, S., Liu, J. & Son, Y.-J. (2018). Effective and Efficient Detection of Moving Targets From a UAV's Camera. *IEEE Transactions on Intelligent Transportation Systems*, 19(2), 497-506. doi : 10.1109/TITS.2017.2782790.

Mur-Artal, R., Montiel, J. M. M. & Tardos, J. D. (2015). ORB-SLAM : a versatile and accurate monocular SLAM system. *IEEE transactions on robotics*, 31(5), 1147–1163.

OpenCV. (2009). Camera Calibration and 3D Reconstruction. Repéré le 2023-02-15 à https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. & Lerer, A. (2017). Automatic differentiation in pytorch.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y. et al. (2009). ROS : an open-source Robot Operating System. *ICRA workshop on open source software*, 3(3.2), 5.

Redmon, J. & Farhadi, A. (2017). YOLO9000 : better, faster, stronger. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271.

Redmon, J. & Farhadi, A. (2018). Yolov3 : An incremental improvement. *arXiv preprint arXiv :1804.02767*.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016). *You only look once : Unified, real-time object detection*. Proceedings of the IEEE conference on computer vision and pattern recognition.

Reed, J. D., da Silva, C. R. & Buehrer, R. M. (2008). Multiple-source localization using line-of-bearing measurements : Approaches to the data association problem. *MILCOM 2008-2008 IEEE Military Communications Conference*, pp. 1–7.

Ren, S., He, K., Girshick, R. & Sun, J. (2015). Faster r-cnn : Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Robotics, O. (2021). ROS 2 documentation. Repéré à https://docs.ros.org/en/foxy/Tutorials/index.html.

Rublee, E., Rabaud, V., Konolige, K. & Bradski, G. (2011). ORB : An efficient alternative to SIFT or SURF. *2011 International conference on computer vision*, pp. 2564–2571.

Santoro, M., AlRegib, G. & Altunbasak, Y. (2012). Misalignment correction for depth estimation using stereoscopic 3-D cameras. *2012 IEEE 14th International Workshop on Multimedia Signal Processing (MMSP)*, pp. 19–24.

Schonberger, J. L. & Frahm, J.-M. (2016). Structure-from-motion revisited. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4104–4113.

Shetty, A. & Gao, G. X. (2017). *Covariance estimation for gps-lidar sensor fusion for uavs*. Proceedings of the 30th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2017).

Soto, R. (2021). Car At Getdrawings. Repéré le 2022-12-15 à https://www.kindpng.com/imgv/owwRb_car-at-getdrawings-top-view-of-cars-clipart/.

Sun, S., Sarukkai, R., Kwok, J. & Shet, V. (2018). Accurate deep direct geo-localization from ground imagery and phone-grade gps. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1016–1023.

Sunday, D. (2021). *Practical Geometry Algorithms with C++ Code*. Daniel Sunday.

Sung, M., Yu, S.-C. & Girdhar, Y. (2017). Vision based real-time fish detection using convolutional neural network. *OCEANS 2017 - Aberdeen*, pp. 1-6. doi : 10.1109/OCEANSE.2017.8084889.

Traa, J. (2013). *Least-Squares Intersection of Lines*. Repéré à https://silo.tips/download/least-squares-intersection-of-lines.

u blox. ZED-F9R-03B. Repéré le 2022-09-15 à https://content.u-blox.com/sites/default/files/documents/ZED-F9R-03B_DataSheet_UBX-22024085.pdf.

u blox. (2021). ZED-F9R-02B : u-blox F9 high precision sensor fusion GNSS receiver. (UBX-21017486). Repéré à https://content.u-blox.com/sites/default/files/ZED-F9R-02B_DataSheet_UBX-21017486.pdf. Revision R01.

Uijlings, J. R., Van De Sande, K. E., Gevers, T. & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2), 154–171.

Wang, C.-Y., Liao, H.-Y. M., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W. & Yeh, I.-H. (2020). CSPNet : A new backbone that can enhance learning capability of CNN. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 390–391.

Wang, C.-Y., Bochkovskiy, A. & Liao, H.-Y. M. (2022). YOLOv7 : Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv :2207.02696*.

Wang, K., Liew, J. H., Zou, Y., Zhou, D. & Feng, J. (2019). Panet : Few-shot image semantic segmentation with prototype alignment. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9197–9206.

Weng, J., Cohen, P., Herniou, M. et al. (1992a). Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on pattern analysis and machine intelligence*, 14(10), 965–980.

Weng, J., Cohen, P., Herniou, M. et al. (1992b). Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on pattern analysis and machine intelligence*, 14(10), 965–980.

Wojke, N., Bewley, A. & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. *2017 IEEE international conference on image processing (ICIP)*, pp. 3645–3649.

Wu, C. (2013). Towards Linear-Time Incremental Structure from Motion. *2013 International Conference on 3D Vision - 3DV 2013*, pp. 127-134. doi : 10.1109/3DV.2013.25.

Xu, R., Lin, H., Lu, K., Cao, L. & Liu, Y. (2021). A forest fire detection system based on ensemble learning. *Forests*, 12(2), 217.

Zhang, H., Yi, R., Chen, J., Sun, Z. & Zhang, H. (2022). Multi-object Tracking for Autonomous Driving with a Monocular Camera Subject to Missing Data. *2022 8th International Conference on Control, Automation and Robotics (ICCAR)*, pp. 224-229. doi : 10.1109/ICCAR55106.2022.9782627.

Zhang, H., Cisse, M., Dauphin, Y. N. & Lopez-Paz, D. (2017). mixup : Beyond empirical risk minimization. *arXiv preprint arXiv :1710.09412*.

Zhang, W., Witharana, C., Li, W., Zhang, C., Li, X. & Parent, J. (2018a). Using deep learning to identify utility poles with crossarms and estimate their locations from google street view images. *Sensors*, 18(8), 2484.

Zhang, W., Witharana, C., Li, W., Zhang, C., Li, X. & Parent, J. (2018b). Using deep learning to identify utility poles with crossarms and estimate their locations from google street view images. *Sensors*, 18(8), 2484.

Zhang, X. & Zhou, T. (2015). Generic Scheimpflug camera model and its calibration. *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2264–2270.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11), 1330–1334.

Zhao, X., Pu, F., Wang, Z., Chen, H. & Xu, Z. (2019). Detection, tracking, and geolocation of moving vehicle from uav using monocular camera. *IEEE Access*, 7, 101160–101170.

Zhao, X., Gao, Z., Zhang, Y. & Chen, B. M. (2020). A Target Tracking and Positioning Framework for Video Satellites Based on SLAM. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1887–1894.

Zou, Z., Shi, Z., Guo, Y. & Ye, J. (2019). Object detection in 20 years : A survey. *arXiv preprint arXiv :1905.05055*.