Prédiction et évitement d'obstacles basés deep learning : Application à la mobilité ferroviaire et routière.

Antoine Mauri

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité AUTOMATIQUE, SIGNAL, PRODUCTIQUE, ROBOTIQUE

Préparée au sein de l'Université de Rouen Normandie

Prédiction et évitement d'obstacles basés deep learning : Application à la mobilité ferroviaire et routière.

Présentée et soutenue par
ANTOINE MAURI

Thèse soutenue le 14/11/2022 devant le jury composé de		
M. PAUL CHECCHIN	PROFESSEUR DES UNIVERSITES, UNIVERSITE CLERMONT AUVERGNE CLERMONT AUVERGNE	Rapporteur du jury
M. CEDRIC DEMONCEAUX	PROFESSEUR DES UNIVERSITES, UNIVERSITE DE BOURGOGNE	Rapporteur du jury
MME SAMIA AINOUZ	PROFESSEUR DES UNIVERSITES, Institut National des Sciences Appliquées Rouen Normandie	Membre du jury
M. REDOUANE KHEMMAR	, ESIGELEC	Membre du jury
M. MURIEL PRESSIGOUT	MAITRE DE CONFERENCES, INST NAT SC APPLIQ RENNES	Membre du jury
M. RÉMI BOUTTEAU	PROFESSEUR DES UNIVERSITES, Université de Rouen Normandie	Directeur de thèse

Thèse dirigée par RÉMI BOUTTEAU (Laboratoire d'Informatique, du Traitement de l'Information et des Systèmes)

UNIVERSITÉ DE ROUEN NORMANDIE

MIIS

litis

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité Automatique, Signal, Productique, Robotique

Préparée au sein de « SEGULA/ESIGELEC »

Prédiction et évitement d'obstacles basés deep learning. Application à la mobilité ferroviaire et routière

Présentée et soutenue par Antoine Mauri

Thèse soutenue publiquement le 14/11/2022 devant le jury composé de		
M. Paul CHECCHIN	Professeur des Universités Université Clermont Auvergne (Institut Pascal)	Rapporteur
M. Cédric DEMONCEAUX	Professeur des Universités Université de Bourgogne (ImVia)	Rapporteur
Mme. Samia AINOUZ	Professeur des Universités INSA Rouen Normandie (LITIS)	Examinateur
M. Muriel PRESSIGOUT	Maître de conférences INSA Rennes (IETR)	Examinateur
M. Rémi BOUTTEAU	Professeur des Universités Université de Rouen Normandie (LITIS)	Directeur de thèse
M. Radouane KHEMMAR	Enseignant Chercheur ESIGELEC (IRSEEM)	Co-encadrant de thèse

Thèse dirigée par Rémi BOUTTEAU, (Laboratoire d'Informatique, du Traitement de l'Information et des Systèmes)

UNIVERSITÉ DE ROUEN NORMANDIE

MIIS

SEGULA TECHNOLOGIES

ESIGELEC irseem

Remerciements

Tout d'abord, je tiens à remercier mon directeur de thèse, Rémi Boutteau (LITIS) et mon encadrant Redouanne Khemmar (IRSEEM) sans qui cette thèse n'aurait pu être menée à bien. Je tiens à les remercier pour leur aide, leurs conseils, leur temps et leur patience durant ces trois années de thèse. Je tiens également à remercier Madjid Haddad (SEGULA Technologies) pour m'avoir accompagné et avoir rendu cette thèse possible. Un merci spécial à tous mes collègues de l'IRSEEM qui m'ont accompagné durant ma thèse, en particulier Isabelle Riguidel pour son aide précieuse et nos conversations passionnantes. Merci aux collègues du pôle SIRD, notamment Vincent, Jérémy, Marc, Anthony et Christophe sans qui mon travail aurait été impossible. Merci aux stagiaires qui m'ont épaulé durant mon travail et en particulier à Camille, Messmer et François pour leur bonne humeur et les bons moments que nous avons partagés. Merci à Antoine Caillot, doctorant, pour nos moments de complicité durant nos thèses respectives. Merci aux vétérans de la thèse, Romain et Louis pour leurs conseils avisés et judicieux tout au long de mes recherches. Enfin, merci à Edgard, Aristide et Pierre qui ont partagé mon bureau et qui m'ont supporté pendant ces longues années.

Je tiens à remercier ma famille d'avoir été à mes côtés tout au long de cette période et d'avoir accepté de m'héberger pendant les confinements de ces années difficiles. La piscine et les petits repas ont été très appréciés.

Cette thèse a été une expérience très enrichissante et m'a appris beaucoup de choses. Merci à tous ceux qui ont contribué à rendre cette expérience inoubliable. Du fond du cœur, merci.

Chapitre

Table des matières

Table des matières

Chapitre

Table des figures

Chapitre

Liste des tableaux

Liste des tableaux

Chapitre

Table des abréviations

ABS	Anti Block System
ADAS	Advanced Driver Assistance Systems
AP	Average Precision
BMP	Bad Matching Pixels
BTS	Big To Small
CNN	Convolutionnal Neural Network
Conv	Convolution
CRIANN	Centre Régional Informatique et d'Applications Numériques de Normandie
CS	Center Score
DDR4	Double Data Rate 4th generation
Deconv	Déconvolution
DS	Dimension Score
EKF	Extended Kalman Filter
ESORAD	Esigelec engineering high school and Segula technologies ROad and RAilway Dataset
Flops	Floating-point operations per second
FOG	Fiber Optic Gyro
GPS	Global Positioning System
GPU	Graphics Processing Unit
GTAV	Grand Theft Auto V
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
IA	Intelligence Artificielle
ID	Identité
IMU	Inertial Measurement Unit
IOU	Intersection Over Union
LiDAR	Light Detection and Ranging
LSTM	Long Short Term Memory
mAP	mean Average Precision
MD2	MonoDepth2
MOT	Multi-Object Tracking
MOTA	Multiple Object Tracking Accuracy
NDC	Normalized Device Coordinate
NMS	Non-Maximum Suppression
OS	Orientation Score
R	Rappel
RADAR	RAdio Detection And Ranging
RAM	Random Access Memory
RCNN	Region based Convolutional Neural Networks
RE	Relative Error
RELU	REctified Linear Unit
RGB	Red Green Blue
RHL	Reverse Huber Loss

RMSE	Root Mean Squared Error
RoI	Region of Interest
RVB	Rouge Vert Bleu
SAE	Society of Automative Engineers
SILU	Sigmoid Linear Unit
SORT	Simple Online and Realtime Tracking
SPP	Spatial Pyramidal Pooling
SRE	Squared Relative Error
SSD	Single Stage Detector
ToF	Time of Flight
VRAM	Video Random Access Memory
WLS	Weighted Least Squares
YOLO	You Only Look Once

Chapitre 1

Introduction Générale

1.1 Introduction à la navigation autonome

1.1.1 Évolution de la mobilité

La société humaine telle que nous la connaissons aujourd'hui est rendue possible par la mobilité moderne. Le transport des personnes et des marchandises est rapide et relativement fiable. En France, selon l'INSEE, le transport routier de marchandises a atteint 317,3 milliards de tonnes-kilomètres en 2018. L'avènement des entreprises de commerce en ligne, Amazon en tête, met en évidence la dépendance de notre société au transport de biens et de marchandises. On estime également que la quantité de marchandises transportées aura tendance à augmenter de manière exponentielle dans un avenir proche. Cette mobilité, grâce à la mondialisation des marchés, permet à de nombreux secteurs de prospérer et les chaînes d'approvisionnement comprennent souvent des produits de différents pays. La mobilité est donc au centre du développement économique. Le prix du pétrole (et donc de la mobilité) est directement lié à la croissance économique d'un pays.

Tout au long de l'histoire de la société humaine, la mobilité a été le principal moteur de la prospérité. L'invention de la roue et du transport maritime a permis le transport sur de longues distances et a permis aux villes proches des voies navigables de commercer et de prospérer, comme Paris avec la Seine, Londres avec la Tamise, ou la ville de Venise directement reliée à la mer Méditerranée. Les premières grandes puissances occidentales, comme l'Empire romain, ont pu s'étendre sur de vastes territoires (Afrique du Nord, Égypte, Turquie, France, Espagne, etc.) grâce au transport maritime rendu possible par la mer Méditerranée. Les nombreuses infrastructures telles que les routes pavées ont également facilité le contrôle des territoires à l'intérieur des terres, facilitant le transport des personnes et des marchandises pour le commerce mais aussi des personnes pour assurer le contrôle de ces vastes territoires.

La mobilité connaît d'importantes évolutions au cours de l'histoire, en 1804 l'invention de la locomotive à vapeur et des chemins de fer améliore drastiquement les transports. Après le premier voyage avec des passagers en 1825, cette invention va permettre de relier des territoires très éloignés et de réduire considérablement le temps du trajet mais aussi d'augmenter la quantité de marchandises pouvant être transportées par rapport aux anciens moyens de transport par chevaux. Symbole emblématique de la conquête de l'Ouest américain, notamment à travers de nombreuses œuvres cinématographiques et littéraires, la locomotive à vapeur permet de rejoindre l'océan Pacifique depuis la côte Est des États-Unis en seulement 8 jours contre plusieurs mois en 1869 avec le Central Pacific.

L'arrivée de l'automobile a également changé la mobilité au début du 20ème siècle. Celle-ci permet à un individu de parcourir une distance beaucoup plus grande avec une autonomie et un confort incomparables au transport par cheval qui était auparavant le moyen le plus répandu. Grâce à l'industrialisation et à l'invention des chaînes de montage par la société Ford, le coût des voitures est drastiquement réduit, ce qui entraîne la démocratisation du transport routier. Les voies de circulation s'adaptent à ce nouveau mode de transport en s'élargissant et en laissant place à des routes asphaltées au lieu de routes pavées ou de chemins de terre. Les autoroutes permettent de réduire le temps de trajet entre les principaux axes de communication des pays. Le développement est tel que le transport de marchandises à l'intérieur des terres se fait désormais principalement par la route. Le train, quant à lui, reste privilégié pour les déplacements sur de longues distances ou pour les déplacements urbains afin d'éviter les encombrements

routiers.

Bien que ces progrès présentent de nombreux avantages, ils ont aussi leurs inconvénients:

— La sécurité: Aujourd'hui, l'une des principales causes de décès chez les jeunes est due aux accidents de la route, avec un total de 1,3 million de morts dans le monde [1]. Les collisions avec les trains ne sont pas rares non plus et causent régulièrement des décès et des retards importants.

— Dépendance énergétique: Le transport routier est largement dépendant des énergies fossiles. Le prix de ce transport varie en fonction du prix du pétrole, qui reste une ressource limitée. Cela peut entraîner une augmentation significative des prix du transport en cas de pénurie de carburant, avec un effet négatif en aval sur l'économie.

— Impacts environnementaux: Les transports sont à l'origine de la majorité des émissions de CO2 et de particules fines dans le monde et sont largement responsables du réchauffement climatique.

— Les embouteillages: La démocratisation de l'automobile ainsi que la croissance de la population ont entraîné des problèmes tels que la congestion des routes à proximité des grandes villes et des zones peuplées, ce qui augmente de manière exponentielle le temps nécessaire pour effectuer un trajet routier.

Pour faire face à ces problèmes, des solutions ont été trouvées, comme l'invention de la ceinture de sécurité à trois points en 1955, qui a permis de réduire considérablement la létalité des collisions sur la route pour le conducteur et les passagers du véhicule. La conception des véhicules vise désormais à minimiser le risque de blessure en cas de collision. L'introduction de l'électronique et des ordinateurs a également permis le développement de systèmes de freinage plus efficaces tels que l'ABS. Les véhicules à propulsion électrique ont également été introduits pour résoudre les problèmes liés à la consommation de carburant. Enfin, nous avons pu assister à la création des premiers systèmes d'aide à la conduite (ADAS) tels que le régulateur de vitesse, Correcteur Électronique de Trajectoire, etc.

1.1.2 Motivation de l'automatisation

Une tendance générale à l'automatisation accrue est observée dans tous les modes de transport, des trains aux avions en passant par les automobiles. Le nombre moyen de décès par an dans les accidents liés aux transports est en baisse depuis les années 1960, grâce aux progrès technologiques et aux réglementations. Toutefois, une proportion importante des accidents est encore liée à l'erreur humaine. Les améliorations de la sécurité apportées aux voitures, tels que les airbags et les freins antiblocage (ABS), sont un exemple de la manière dont la technologie peut contribuer à réduire le taux de mortalité. En outre, il a été démontré que le freinage d'urgence automatique réduit considérablement la gravité des accidents, en particulier lorsque le véhicule roule à grande vitesse. Il est donc clair que les technologies de conduite automatisée contribueront à améliorer la sécurité routière.

Les avantages de l'automatisation sont multiples:

— Amélioration de la sécurité: Le niveau actuel de sécurité routière est insuffisant. Les accidents de la circulation constituent la cinquième cause de décès dans le monde et sont responsables de 1,3 million de morts par an. L'automatisation améliorera la sécurité routière en réduisant le facteur humain dans les accidents de la route.

— Meilleure empreinte carbone: Grâce grâce à l'optimisation de la conduite et des itinéraires pour éviter les embouteillages, la consommation de carburant peut être considérablement réduite.

— Meilleure qualité de vie: La conduite automatisée réduira la fatigue résultant de longs trajets et augmentera ainsi le confort du passager.

L'automatisation peut également contribuer à la sécurité ferroviaire en augmentant la sécurité par

la détection des objets sur la voie, la régulation de la vitesse, le freinage et l'évitement des collisions. Toutefois, il reste quelques obstacles à surmonter avant que la conduite totalement autonome ne devienne une réalité:

— L'acceptation par les conducteurs: La méfiance du public vis-à-vis des voitures autonomes reste un défi. Certains conducteurs peuvent être réticents à l'idée de céder le contrôle à une machine.

— Préoccupations d'ordre juridique: Le cadre juridique de la conduite autonome n'est toujours pas clair. La question de la responsabilité en cas d'accident d'une voiture conduite de manière autonome est une préoccupation majeure pour les constructeurs.

— Défis techniques: Il existe des limitations techniques qui empêchent l'adoption généralisée des véhicules entièrement autonomes. Il s'agit notamment de l'absence d'algorithmes de perception robustes capables de gérer des conditions météorologiques défavorables, de la nécessité de disposer de données cartographiques très fiables pour la navigation et de l'absence de protocoles de communication normalisés entre les véhicules et les autres usagers de la route.

— Problèmes de confidentialité: La collecte et le traitement des données personnelles constituent un problème majeur dans le contexte de la conduite autonome. L'utilisation de caméras et de capteurs pour surveiller le comportement du conducteur et des passagers soulève de sérieuses inquiétudes quant au respect de la vie privée.

— Risques de cybersécurité: Les véhicules autonomes s'appuient sur des réseaux informatiques pour fonctionner correctement. Leur sécurité doit être garantie contre les cyberattaques.

1.1.3 Niveaux d'automatisations

Afin de faciliter la compréhension de l'automatisation des véhicules, la SAE (Society of Automative Engineers) a défini une norme de différents niveaux d'automatisation:

— Niveau 0 (pas d'automatisation): Ce niveau correspond aux véhicules conventionnels sans aucune forme d'automatisation. Le conducteur est responsable de tous les aspects de la conduite, y compris la direction, l'accélération, le freinage, etc.

— Niveau 1 (assistance au conducteur): À ce niveau, le véhicule offre une automatisation partielle en prenant le contrôle dans certaines circonstances, par exemple lors d'un freinage violent ou lorsque le véhicule commence à déraper en raison d'une route glissante. Le conducteur doit toujours être attentif et intervenir à certains moments.

— Niveau 2 (automatisation partielle): À ce niveau, le véhicule ne prend le contrôle que dans des situations spécifiques, comme sur les autoroutes, mais le conducteur doit garder les mains sur le volant et être prêt à reprendre le contrôle chaque fois que nécessaire.

— Niveau 3 (automatisation conditionnelle): À ce niveau, le véhicule prend le contrôle total, sauf dans des cas spécifiques où il doit pouvoir communiquer avec le conducteur.

— Niveau 4 (automatisation poussée): À ce niveau, le véhicule gère tous les aspects de la conduite, notamment la direction, l'accélération, le freinage, etc. Cependant, il n'élimine pas complètement le besoin d'intervention humaine ; un conducteur doit toujours être disponible pour des situations occasionnelles.

— Niveau 5 (automatisation complète): À ce niveau, le véhicule est entièrement automatisé et exécute toutes les fonctions de conduite sans aucune intervention du conducteur.

Pour le cas de la voiture autonome, ces niveaux sont illustrés dans la Figure 1.1. À partir du niveau

d'automatisation 3, les systèmes d'automatisation de la perception de l'environnement deviennent indispensables. Des capteurs permettant de déterminer si un obstacle se présente (Caméra, LiDAR, RADAR, etc.) ainsi qu'un ordinateur de bord performant afin de traiter ces données sont requis. L'étape suivante consiste à développer des algorithmes avancés pour permettre au véhicule de réagir rapidement et en toute sécurité dans son environnement. Par exemple, le véhicule doit être capable de prédire si un piéton va traverser son chemin ou si une voiture devant doit tourner à gauche. S'il détecte qu'une collision est inévitable, le système doit décider de l'action à entreprendre pour l'éviter. Il peut soit freiner brusquement, soit faire une embardée. Enfin, la décision doit être communiquée au conducteur pour qu'il puisse réagir en conséquence.

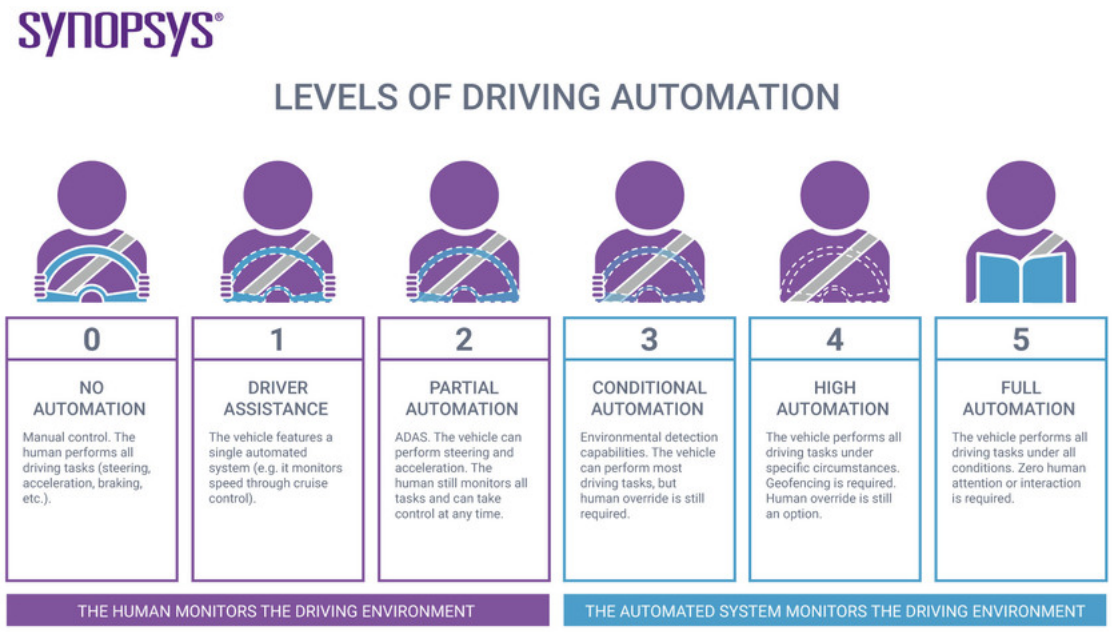

Figure 1.1: Niveaux d'automatisation de la voiture autonome [2].

1.1.4 Systèmes utilisés pour l'automatisation

L'automatisation d'un véhicule peut être séparée en trois couches:

Couche de perception

Le premier sous-système est le système de perception. Son objectif est de détecter tous les objets entourant le véhicule et d'interpréter leur signification. Le système reçoit des données de différentes sources, caméras, RADARs et LiDARs en font partie. Ces capteurs fournissent des informations sur la position des obstacles par rapport au véhicule, ainsi que sur certaines propriétés de chaque objet, comme la vitesse, la taille, etc. D'autres capteurs tels que le GPS et les unités de mesure inertielle (IMU) sont également utilisés pour déterminer la position du véhicule. Tous ces capteurs envoient des données à un ordinateur embarqué appelé ordinateur de perception qui traite les données reçues et génère une représentation de l'environnement. La Figure 1.2 présente une vue simplifiée du système de perception.

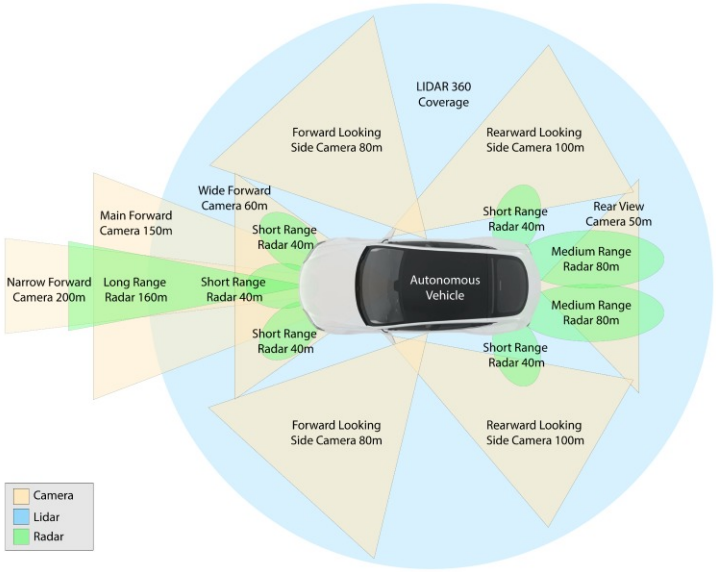

Figure 1.2: Exemple d'un système complet de perception pour la voiture autonome [3].

Couche de planification

La deuxième couche est le système de planification. Son objectif est de prendre des décisions selon les situations: arrêt, tourner à droite, à gauche, accélérer, suivre une trajectoire, etc. Cela signifie qu'il doit prendre en compte de nombreux paramètres tels que les conditions météorologiques, les feux de circulation, la topologie de la route, etc.Pour ce faire, il fait appel à la cartographie qui contiennent des informations sur la disposition des routes, des bâtiments, des intersections, etc. La carte est mise à jour en permanence par un système de cartographie qui utilise des données provenant de capteurs tels que des caméras et des LiDARs.

Couche de contrôle

La troisième couche est le système de contrôle. Son rôle est d'exécuter la trajectoire générée par le système de planification en contrôlant les actionneurs situés dans la direction, les freins et la pédale d'accélérateur.

1.1.5 Capteurs utilisés dans la couche de perception

La couche de perception contient plusieurs types de capteurs, chacun effectuant des tâches différentes. Certains capteurs sont passifs, d'autres sont actifs. Les capteurs passifs détectent simplement le monde qui nous entoure et transmettent des données brutes au calculateur. Les capteurs actifs émettent des signaux (signaux lumineux, signaux électromagnétiques, etc.) et reçoivent en retour un signal réfléchi par l'environnement qui entoure le capteur.

Caméras

Les caméras sont l'un des capteurs les plus utilisés dans les voitures autonomes. Elles sont peu coûteuses et faciles à intégrer dans les véhicules existants. Elles sont utilisées pour reconnaître les marquages de voie, les panneaux de signalisation et les piétons. Elles peuvent également être utilisées pour estimer la distance entre deux points ou pour suivre un objet en mouvement.

Les caméras sont divisées en deux catégories: monoculaire et stéréoscopique. Les caméras monoculaires n'ont qu'un seul objectif et capturent une seule image. Les caméras stéréoscopiques utilisent deux objectifs séparés horizontalement ou verticalement par une distance fixe. Elles créent deux vues légèrement décalées pouvant être combinées afin de former un modèle 3D de la scène.

RADARs

Les RADARs émettent des ondes électromagnétiques qui sont ensuite réfléchies par la cible. Les signaux de retour (écho RADAR) sont ensuite captés par le récepteur RADAR. La distance de la cible peut alors être obtenue en mesurant le temps d'aller-retour du signal. La vitesse peut être déterminée par le décalage de fréquence entre les deux signaux selon l'effet Doppler. La direction de la cible quant à elle peut être trouvée par la position angulaire de l'antenne où le signal de retour a été capté.

Un RADAR installé sur une voiture peut aider à détecter les autres véhicules ainsi que les piétons, les cyclistes, les animaux et autres obstacles. Cependant, ils peuvent être installés sur des véhicules existants avec une relative facilité.

Les RADARs peuvent être organisés en différentes classes:

— Les RADARs à courte portée: Les RADARs à courte portée utilisent la fréquence de 24 Ghz et sont utilisés pour des applications à courte portée comme la détection des angles morts, l'aide au stationnement ou la détection d'obstacles et la prévention des collisions. Ces RADARs ont besoin d'une antenne orientable avec un grand angle de balayage, créant un large champ de vision.

— RADAR à longue portée: Les RADARs à longue portée utilisant la bande des 77 Ghz (de 76 à 81 Ghz) offrent une meilleure précision et une meilleure résolution. Ils sont utilisés pour mesurer la distance et la vitesse des autres véhicules et pour détecter des objets dans un champ de vision plus large. Les applications à longue portée nécessitent des antennes directives qui offrent une meilleure résolution dans un champ de balayage plus limité. Ces systèmes offrent des portées de 80 m à 200 m ou plus.

LiDARs

Les LiDARs (light detection and ranging) émettent des faisceaux laser et mesurent le temps qu'ils mettent à se réfléchir sur les objets situés sur leur trajectoire. Cela leur permet de déterminer la distance de l'objet et sa direction. Les LiDARs sont très précis et peuvent facilement différencier différents types d'objets. Ils sont particulièrement utiles pour détecter les piétons, véhicules, etc.. Les lasers émis par le LiDAR sont du domaine infrarouge (longueurs d'onde de 250 nm à 10 µm) afin d'éviter de perturber les autres capteurs fonctionnant sur des plages de fréquences différentes. Les LiDARs les plus répendus sont les LiDARs à balayages lasers qui permettent d'obtenir la position d'une cible.

GPS

Le GPS est un système de positionnement par satellite. Ce système utilise des satellites en orbite terrestre basse pour déterminer la position de tout récepteur au sol. Un véhicule équipé d'un récepteur GPS peut déterminer sa position sur une carte en utilisant les signaux de plusieurs satellites. Les récepteurs GPS sont relativement peu coûteux et faciles à intégrer dans les véhicules existants. Cependant, ils souffrent de fréquentes pertes de signal et leur précision est insuffisante pour être utilisés seuls pour la navigation. Ils peuvent également être sensibles aux conditions météorologiques.

Unités de mesure inertielle (IMU)

Une IMU contient trois accéléromètres et trois gyroscopes qui détectent respectivement les changements de direction et de vitesse de rotation. Ils sont utilisés pour mesurer l'orientation du véhicule par rapport à un repère de référence global. Associées à un récepteur GPS, elles permettent au véhicule de suivre sa position absolue même s'il perd la réception GPS.

1.2 Contribution de la thèse

Comme indiqué dans la section précédente, pour améliorer la sécurité des transports et rendre la conduite plus autonome, les véhicules doivent avoir une meilleure perception de leur environnement. Bien que le développement de la voiture autonome fasse l'objet d'une grande attention, le transport ferroviaire suit aussi la même voie. Ce développement est motivé par la demande croissante d'automatisation dans les transports, ainsi que par la disponibilité croissante de la puissance de calcul. L'enjeu est à la fois d'améliorer le confort de conduite grâce à l'aide à la navigation et d'accroître la sécurité pendant la navigation, sachant que la majorité des accidents de la route sont dus à une erreur humaine. Cependant, comme nous l'avons vu précédemment, avant même de pouvoir concevoir un système entièrement autonome, il est nécessaire de développer des systèmes de perception fiables afin d'analyser et d'éviter tout risque d'accident. Cette perception doit garantir une bonne détection des objets, mais plus généralement une bonne interprétation des scènes. Dans la mobilité ferroviaire, la problématique est assez similaire à celle du secteur routier. Dans les deux types d'environnement, les objets d'intérêt sont les véhicules, les piétons, les bus, les cyclistes, les arbres, etc. Cependant, si le secteur routier est largement couvert par la littérature scientifique, ce n'est pas le cas du secteur ferroviaire. Ceci est en partie dû au manque de jeux de données spécifiques.

Une perception fiable doit assurer l'accomplissement d'au moins trois tâches essentielles: la détection d'objets, leur localisation et leur suivi. Une estimation précise et fiable de la profondeur pourrait accroître considérablement la sécurité en estimant la distance entre le véhicule et les objets détectés, évitant ainsi les collisions.

D'autres informations sur les objets, telles que leurs dimensions et leur orientation, sont également très utiles pour la sécurité. L'estimation de ces paramètres est donc une tâche importante qui devrait être effectuée par un système avancé d'aide à la conduite (ADAS).

La mesure de la distance aux objets repose généralement sur une gamme de capteurs complémentaires: RADAR [4], LiDAR [5], ou caméra à temps de vol (ToF) [6]. Cependant, ces capteurs présentent certaines limites car ils sont souvent coûteux et encombrants.

C'est dans ce contexte que cette thèse s'inscrit, l'objectif est de créer un système de perception fiable utilisant une seule modalité, ici les images provenant d'une caméra, afin de détecter les obstacles mettant en danger la vie des passagers. Le système doit être générique afin d'être utilisé sur tout type de caméra dans un contexte de trafic sur route mais aussi sur rail.

Dans cette thèse CIFFRE, financée par SEGULA Technologies et réalisée en collaboration avec le laboratoire IRSEEM, nous allons explorer différentes solutions basées sur l'intelligence artificielle et l'apprentissage profond pour l'évitement obstacles à partir d'images.

Les travaux ont donnés lieu à plusieurs contributions et ont été publiés dans des conférences et des revues internationales:

— Une nouvelle méthode de détection, localisation et de suivi d'objets basée sur des méthodes déjà établies et un filtre de Kalman [7]

— Un nouveau protocole d'évaluation de l'estimation de profondeur dédiée à la localisation d'objets [8, 9]

— Une étude comparative de différentes méthodes d'estimation de profondeurs basées sur des images stéréoscopiques ou monoculaires [10]

— Une première méthode temps réel pour la détection des objets en deux étages [11]

— Une seconde méthode temps réel pour la détection des objets en un seul étage unique [12]

— Un jeu de donnée hybride contenant à la fois des images réelles et virtuelles pour le domaine ferroviaire et routier [13]

1.3 Organisation du mémoire

Cette thèse est divisée en sept chapitres. Le premier chapitre introduit le sujet de la thèse ainsi que ses enjeux et présente brièvement les techniques existantes. Le chapitre 2 présente les pré-requis nécessaires à la compréhension de ce mémoire. Le chapitre 3 couvre notre première approche pour la détection, la localisation et le suivi d'objets en combinant un détecteur d'objets en temps réel nommé Yolov3 [14], un algorithme d'estimation de profondeur à partir d'images stéréo nommé MadNet [15] et un filtre de Kalman [16]. Le chapitre 4 présente notre nouveau protocole d'estimation de la profondeur, plus adapté à la localisation d'objets dans des conditions routières et ferroviaires. Dans le chapitre 5, nous décrivons la construction de nos bases de données afin de disposer de données pour le domaine ferroviaire. La détection d'objets 3D et nos contributions principales sont ensuite discutées dans le chapitre 6. Enfin dans le chapitre 7 nous tirerons les conclusions et perspectives de ce travail.

Chapitre 2

Pré-requis

Dans ce chapitre, nous allons tout d'abord présenter les prérequis nécessaires à la bonne compréhension de cette thèse. Nous introduirons les concepts de l'apprentissage profond ainsi que les ressources utilisées tout au long des travaux réalisés au cours de la thèse.

2.1 Introduction à l'apprentissage profond

2.1.1 Vue d'ensemble

L'apprentissage profond (ou *Deep Learning*) fait partie d'une plus grande famille de méthodes d'apprentissage automatique (*Machine Learning*) basées sur les réseaux neuronaux artificiels (voir Figure 2.1). L'apprentissage peut être supervisé (à l'aide de données annotées), semi-supervisé (nécessitant une quantité moindre de données) ou non supervisé (sans aucune donnée annotée) [17].

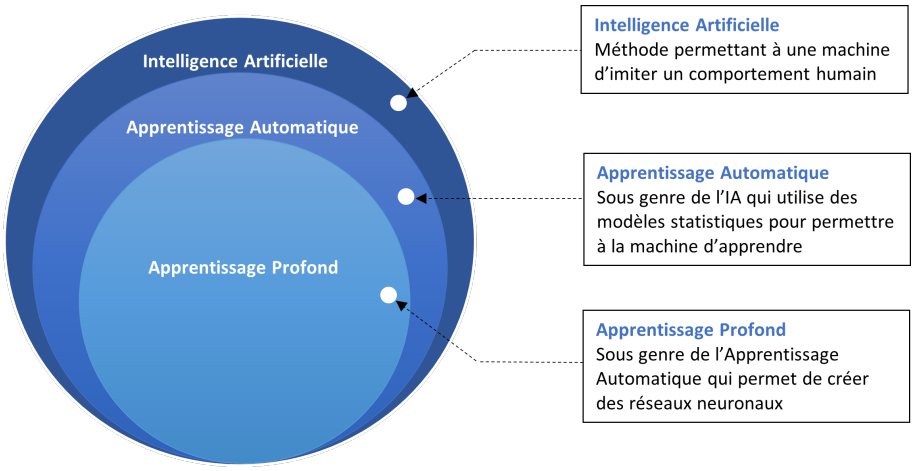

Figure 2.1: Familles de l'Intelligence Artificielle.

Parmi les architectures utilisées dans l'apprentissage profond figurent les réseaux neuronaux profonds, l'apprentissage par renforcement, les réseaux neuronaux récurrents et les réseaux neuronaux convolutifs. Les applications sont très diverses et peuvent inclure la vision par ordinateur, la reconnaissance de l'écriture, la génération de texte, la reconnaissance vocale, le traitement du langage naturel, la traduction automatique, la bioinformatique, la conception de médicaments, l'analyse d'images médicales, la climatologie et l'inspection des matériaux [18, 19, 20, 21].

Le terme "profond" dans "apprentissage profond" fait référence à l'utilisation de plusieurs couches de neurones dans le réseau. Les perceptrons linéaires [22] ne peuvent pas être utilisés dans des cas d'applications complexes comme par exemple pour la classification d'objets dans l'image, tandis qu'un réseau neuronal, ayant des fonctions d'activation non polynomiales et de nombreuses couches cachées, le peut.

Pour les tâches d'apprentissage supervisé, les méthodes d'apprentissage profond éliminent la nécessité pour l'utilisateur de définir les caractéristiques à apprendre en traduisant les données en représentations intermédiaires compactes similaires aux composantes principales (voir Figure 2.2)

Dans l'apprentissage profond, chaque niveau apprend à transformer ses données d'entrées en une

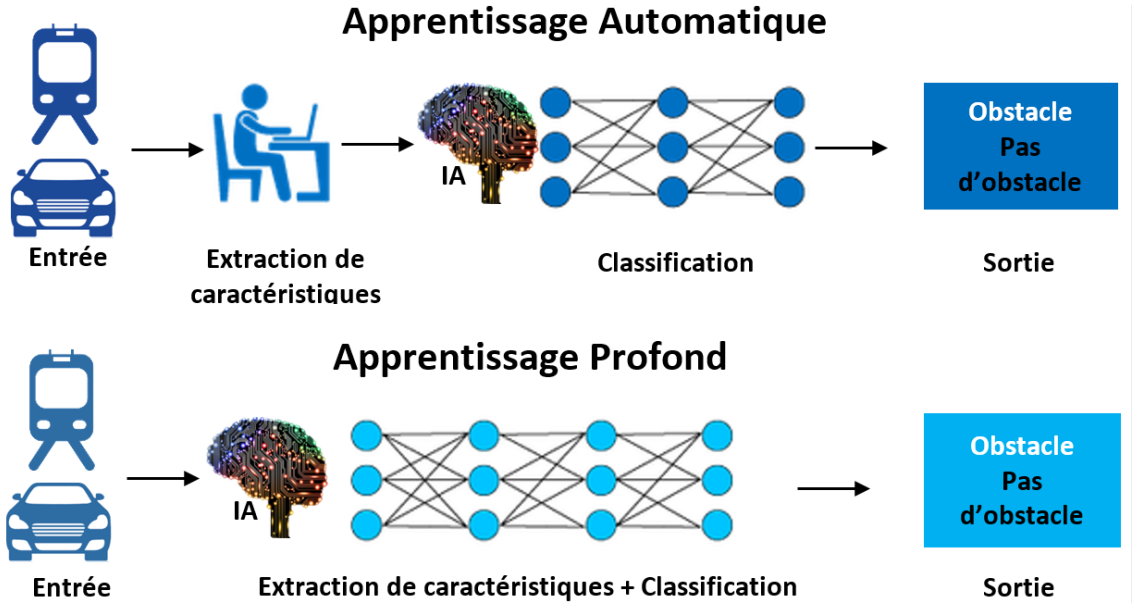

FIGURE 2.2: Exemple pour un cas de classification d'objets. Un algorithme d'apprentissage automatique va nécessiter que l'utilisateur définisse lui même les caractéristiques à apprendre tandis que l'algorithme d'apprentissage profond va pouvoir les déterminer sans aucune intervention de l'utilisateur.

représentation légèrement plus abstraite. Ainsi, si l'on prend par exemple une application de reconnaissance d'images, l'entrée du réseau se présente sous la forme d'une matrice de pixels représentant l'image. Les couches du réseau vont alors abstraire les pixels et coder les bords et les textures des objets présents dans l'image afin que les dernières couches du réseau puissent faire une prédiction du type d'objet présent dans l'image. Il est important de noter qu'un processus d'apprentissage profond peut apprendre, sans intervention humaine, quelles caractéristiques placer de manière optimale dans quelle couche. Cela n'élimine pas la nécessité d'un réglage manuel ; par exemple, en variant le nombre de couches et la taille des couches, on peut obtenir différents degrés d'abstraction [17, 23].

Le neurone est la base des réseaux neuronaux utilisés dans l'apprentissage profond. Leur nom fait référence aux mêmes neurones qui constituent la base des unités de calcul que l'on peut trouver dans le cerveau. En effet, que ce soit pour l'apprentissage profond ou pour les cerveaux organiques, le principe de fonctionnement du neurone reste plus ou moins identique. Le neurone présent dans le cerveau reçoit des signaux d'autres neurones via ses dendrites et renvoie son signal de sortie via son axone. Ce dernier va ensuite être connecté aux neurones suivants via des synapses qui transmettront ensuite le signal à leurs dendrites. Si on formalise mathématiquement cette opération, les signaux d'entrée $\mathbf{x} = [x_0, etc., x_i]^T$ arrivant des axones des neurones précédents vont passer par les synapses du neurone et seront multipliés par des valeurs $\mathbf{w} = [w_0, etc., w_i]^T$ caractérisant la force de ces synapses. Le signal reçu par les dendrites correspond donc à $\mathbf{x}.\mathbf{w}$. La force des synapses est un paramètre qui peut être appris par le neurone et varie donc en fonction de chaque neurone et synapse. Les dentrites conduisent ce signal au centre de la cellule où ils sont additionnés. Si la valeur additionnée des signaux est supérieure à un certain seuil, un pic de signal est envoyé à l'axone. Dans le modèle mathématique ce pic est modélisé grâce à une fonction d'activation f [24]. La Figure 2.3 présente un exemple de neurone organique.

Le neurone utilisé dans l'apprentissage profond est disposé en couches de neurones formant le réseau. Chacun des neurones est connecté avec ceux de la couche précédente. Ainsi pour des valeurs d'entrée $\mathbf{x}$ et $\mathbf{w}$, les poids correspondant à chaque entrée, la sortie d'un neurone peut s'écrire comme dans l'Equation

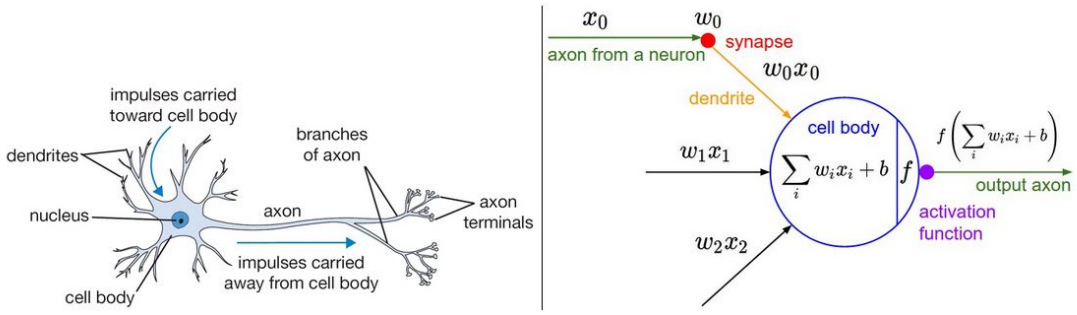

FIGURE 2.3: Exemple d'un neurone utilisé dans l'apprentissage profond [24].

(2.1):

$$s = f\left(\sum_i^N w_i x_i + b\right) \tag{2.1}$$

N représente le nombre de neurones des couches précédentes connectés au neurone courant, b le biais du neurone et f la fonction d'activation. La fonction d'activation permet d'introduire une non-linéarité à une opération (introduire des propriétés non-linéaires à un réseau CNN est justifié par le fait que la plupart des données des environnements routiers et ferroviaires ne sont pas linéaires, or il serait commode qu'un réseau CNN puisse apprendre et prédire des relations non-linéaires), par exemple la fonction la plus répandue est la fonction RELU (*REctified Linear Unit*) introduite dans les travaux de [25]. Un exemple de réseau neuronal est présenté dans la Figure 2.4.

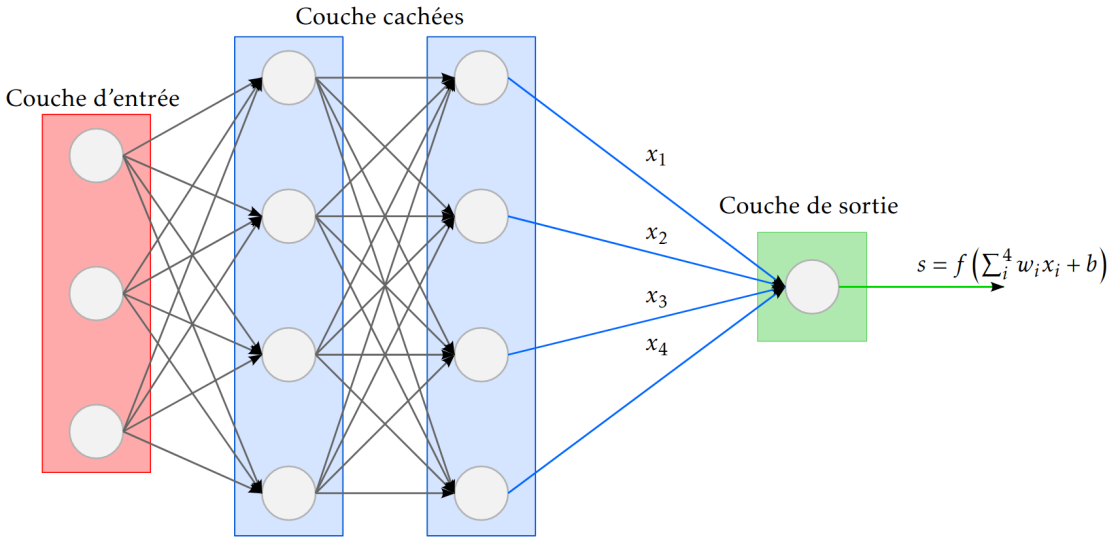

FIGURE 2.4: Exemple d'un réseau de neurones profond.

L'apprentissage d'un réseau de neurones est divisé en quatre étapes:

La prédiction (*Forward Pass*)

Nous avons déjà vu comment un réseau neuronal fonctionne pour obtenir une prédiction à partir d'un vecteur d'entrée (voir Figure 2.4)

La rétropropagation (*Back Propagation*)

Nous avons vu précédemment qu'un neurone possède deux paramètres "apprenables": ses poids $\mathbf{w}$ et son biais b. Cet apprentissage fonctionne d'abord en déterminant l'influence de ces paramètres pour chaque neurone sur la sortie obtenue à l'extrémité du réseau. C'est ce qu'on appelle la rétropropagation par descente du gradient (*Backpropagation*). Pour donner un exemple, si on prend un réseau de neurones à quatre couches avec f_{ik} le k-ième neurone de la i-ième couche, $\mathbf{x} = [x_1, x_2]$ le vecteur d'entrée et y la sortie du réseau, on part de cette valeur de sortie et on calcule les gradients de chacune des couches afin de déterminer la contribution de chacun des neurones de ce réseau pour obtenir ce même résultat. La Figure 2.5 illustre le principe de fonctionnement de la rétro-propagation.

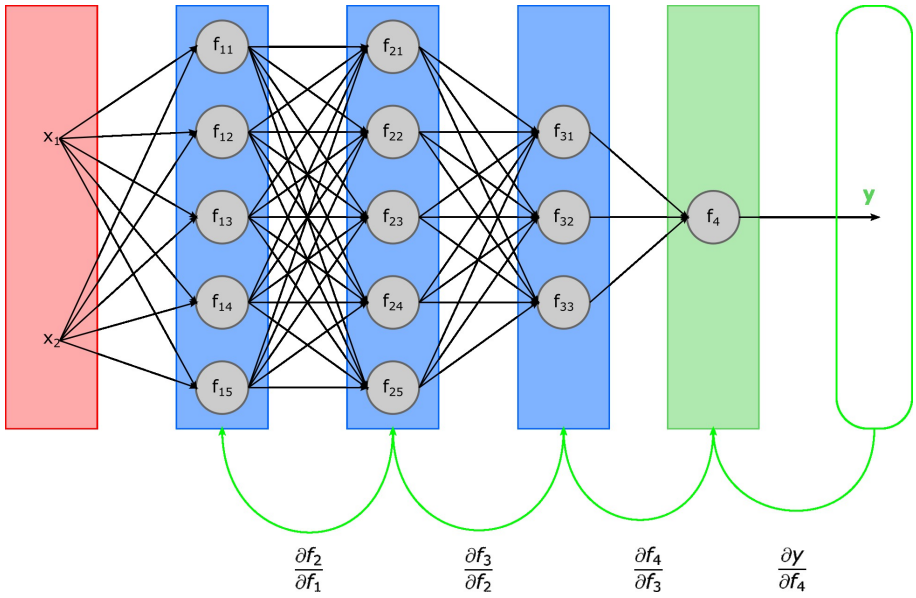

Figure 2.5: Calculs des gradients lors de la rétro-propagation (*Backpropagation*) d'un réseau de neurones.

Calcul de la fonction perte

Afin de déterminer la qualité de notre prédiction, nous devons mesurer ce que l'on appelle la perte. Pour l'entraînement supervisé, nous utilisons la vérité terrain pour la calculer. Il existe de nombreuses fonctions utilisées pour calculer la perte, mais si nous considérons ici que nous sommes dans un problème de régression, avec $\widetilde{y}$ la vérité terrain, la perte peut être calculée à l'aide d'une simple fonction comme l'erreur moyenne absolue (nommée *L1loss*) décrite dans l'équation (2.2):

$$L1loss = \left| y - \widetilde{y} \right| \tag{2.2}$$

Optimisation

Enfin, pour minimiser cette perte, nous allons utiliser un algorithme d'optimisation qui prend en entrée les gradients calculés lors de la rétropropagation afin de trouver les valeurs $\mathbf{w}_{ik}$ et b_{ik} pour chacun des neurones afin d'obtenir une perte minimale. L'algorithme d'optimisation le plus courant pour cette tâche est l'algorithme de descente de gradient. Il permet de faire converger la fonction coût vers un minimum local. La fonction coût étant dépendante de l'ensemble des neurones connectés du réseau, et comme chaque neurone est fonction de poids, de biais et d'entrées, ce sont donc ces paramètres que l'algorithme de descente de gradient ajuste afin de minimiser la fonction coût. L'algorithme met à jour à chaque étape

les paramètres des neurones comme indiqué dans l'équation (2.3) et (2.4):

$$w = w - \alpha \frac{\partial L1loss}{\partial w} \tag{2.3}$$

$$b = b - \alpha \frac{\partial L1loss}{\partial b} \tag{2.4}$$

Où α le taux d'apprentissage (ou *learning rate*). Une fois que cette étape d'optimisation est terminée, nous revenons à l'étape de prédiction et recommençons le processus. L'apprentissage est dit "terminé" lorsque la perte atteint son minimum.

2.1.2 Réseaux de convolutions

Bien que les réseaux de neurones profonds puissent être utilisés dans de nombreux cas, ils présentent l'inconvénient d'avoir un coût de calcul exponentiel par rapport au nombre de neurones mais aussi à la taille de l'entrée. Ainsi, pour une simple matrice unidimensionnelle le temps de calcul reste raisonnable mais lorsqu'une image est utilisée en entrée le coût de calcul devient trop important (une image peut être représentée comme une matrice de dimension (H, W, 3) avec un nombre total d'éléments de $H \times W \times 3$). Mais ce n'est pas seulement une question de coût de calcul mais aussi d'encodage, en effet, les réseaux de neurones profonds classiques auront des difficultés à comprendre les corrélations spatiales que l'on peut trouver dans une image de haute dimension.

C'est pourquoi les réseaux convolutifs profonds ont été introduits. Ces derniers sont par essence similaires aux réseaux neuronaux ordinaires: ils sont également composés de neurones dont les paramètres, tels que les poids et les biais, peuvent être appris. Chacun des neurones effectue un produit scalaire entre ses entrées et ses poids et biais avant d'utiliser une fonction d'activation pour introduire une non-linéarité. Ces réseaux disposent également de fonctions pour calculer la perte. Le processus d'apprentissage est identique.

La différence entre ces deux types de réseaux provient des architectures spéciales pour le traitement des images et le codage de leurs caractéristiques. Cela permet de réduire considérablement le nombre de paramètres mais aussi d'améliorer la compréhension spatiale du réseau. Les caractéristiques codées de l'image lors de son passage dans une couche convolutive sont appelées carte de caractéristiques (voir Figure 2.6).

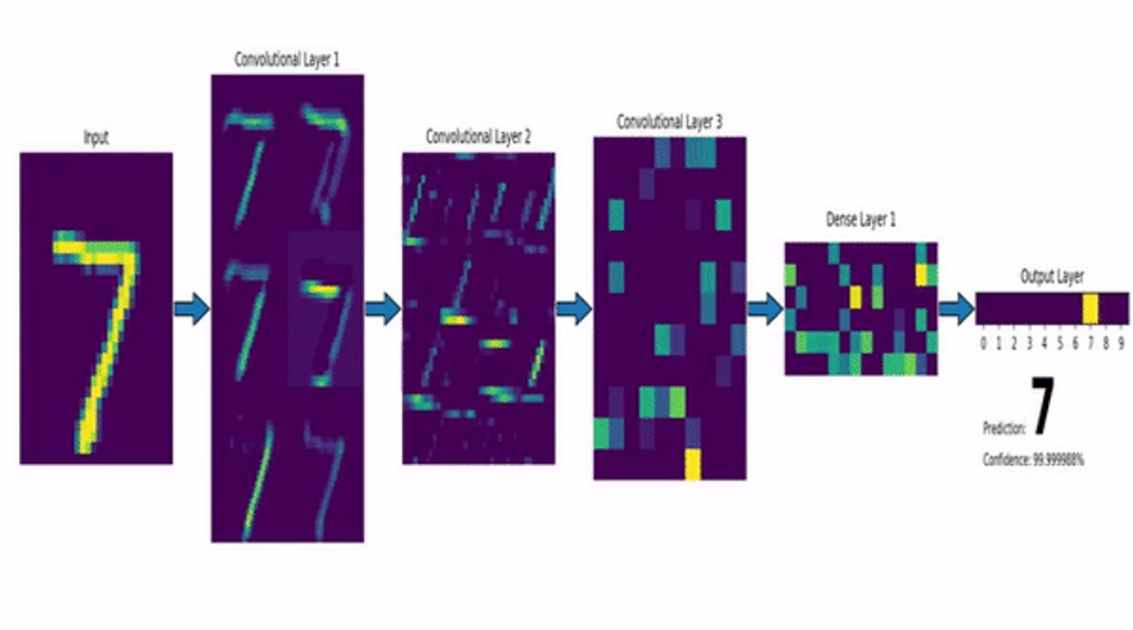

FIGURE 2.6: Exemple des cartes de caractéristiques pouvant être obtenues dans chacune des couche d'un réseau convolutif [26].

Comme nous l'avons dit précédemment, les réseaux neuronaux reçoivent une entrée sous forme de vecteur et à travers une série de couches de neurones, la transforment afin d'obtenir une prédiction. Chaque couche cachée est constituée d'un ensemble de neurones, où chaque neurone est entièrement connecté à tous les neurones de la couche précédente, et où les neurones d'une même couche fonctionnent de manière totalement indépendante et ne partagent aucune connexion. Ce type d'architecture n'est pas adapté aux images complètes en raison du grand nombre de paramètres qu'il impliquerait mais aussi parce que nous n'utiliserions pas les indices des pixels adjacents (car tous les neurones apprennent de manière indépendante).

Les réseaux convolutifs ont des neurones disposés en trois dimensions: largeur, hauteur et profondeur (il s'agit ici du nombre de canaux présents sur l'entrée). Contrairement à l'apprentissage profond "classique", les neurones d'une couche ne sont connectés qu'a un nombre limité de la couche précedent. Un exemple est présenté dans la Figure 2.7.

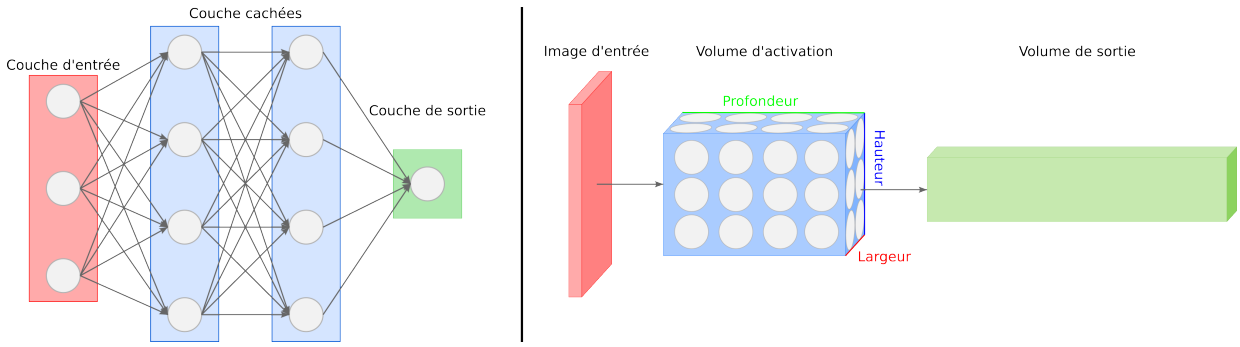

FIGURE 2.7: Comparaison entre une architecture de réseau profond "classique" et un réseau convolutif. Le réseau neuronal profond est représenté à gauche tandis que le réseau convolutif est représenté à droite.

Parmi les couches présentes dans un réseau convolutif, on dénombre les couches convolutives, les couches de *Pooling*, les couches de normalisation et les couches d'activations.

Couche convolutive

La couche convolutive représente la base de chaque réseau convolutif. Les paramètres d'une couche convolutive consistent en un ensemble de filtres "apprenables". Chaque filtre ne prend en charge qu'une petite partie de la hauteur et de la largeur, mais s'étend sur toute la profondeur du volume d'entrée. Par exemple, un filtre de la première couche d'un réseau convolutif pourrait avoir une taille de $5 \times 5 \times 3$ (5 pixels en largeur et en hauteur et 3 en profondeur car nous sommes dans le cas d'une image RVB). Pendant la phase de prédiction (ou *forward pass*) nous faisons glisser (plus précisément, nous convoluons) chaque filtre sur la largeur et la hauteur du volume d'entrée et nous calculons les produits scalaires entre les entrées du filtre et l'entrée à n'importe quelle position. Lorsque nous faisons glisser le filtre sur la largeur et la hauteur du volume d'entrée, nous produisons une carte d'activation bidimensionnelle qui donne les résultats de ce filtre à chaque position spatiale. Un exemple est donné dans la Figure 2.8.

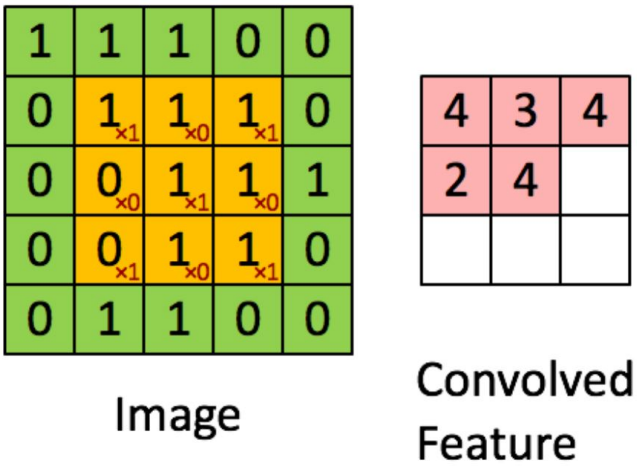

Figure 2.8: Extraction de caractéristiques grâce à un filtre convolutif [27].

Ces filtres permettront au réseau d'apprendre les caractéristiques de l'image telles que les bords et les coins des objets ou les textures qui peuvent être présentes. Les filtres de chacune des couches convolutionnelles produiront chacun une carte d'activation bidimensionnelle indépendante qui peut être empilée pour produire le volume de sortie de la couche en question.

Une couche convolutive est définie par quatre hyperparamètres qui contrôlent la taille du volume de sortie:

(1) Le nombre de filtres qui apprennent chacun quelque chose de différent de l'entrée.

(2) La taille du champ réceptif (*Kernel Size*) est définie par la taille des filtres. Dans les premières couches, les filtres avec de grands champs réceptifs sont préférés afin de réduire le nombre de paramètres et d'augmenter la compréhension spatiale des grandes images, tandis que dans les couches plus profondes, des filtres plus petits sont préférés afin de s'adapter à la réduction progressive, le long du réseau des dimensions des volumes d'entrée.

(3) Le pas (*Stride*) va permettre de définir le pas avec lequel le filtre est déplacé. Par exemple, lorsque cette valeur est fixée à 1, le filtre est déplacé d'un pixel à la fois. Lorsqu'elle est définie sur 3, le filtre saute de 3 pixels à la fois, etc. La définition d'un pas supérieur à 1 conduit logiquement à une réduction des dimensions du volume de sortie par rapport au volume d'entrée.

(4) Enfin, l'espacement des neurones (*Zero Padding*) permet de remplir le volume d'entrée avec des 0

afin d'augmenter ses dimensions. Ceci est utile pour contrôler la taille des volumes de sortie.

Les dimensions du volume de sortie peuvent donc être déterminées en fonction de la taille du volume d'entrée (W), du nombre de filtres (F), du pas utilisé pour les filtres (S) et de l'espacement des neurones (P) par la Equation (2.5):

$$W_{sortie} = \frac{(W - F + 2P)}{S} + 1 \tag{2.5}$$

Par exemple, pour une entrée 7×7 et un filtre 3×3 avec un pas de 1 et sans espacement, on obtient une sortie 5×5. Avec un pas de 2, on obtiendrait une sortie 3×3.

Couche de Pooling

Les couches de pooling sont couramment placées entre des blocs de couches convolutives successives afin de réduire progressivement la taille spatiale des volumes d'activation et ainsi réduire le nombre de paramètres et alléger le réseau. Ces couches, en réduisant la complexité du réseau, permettent également de contrôler le sur-apprentissage. Parmi les différentes couches de pooling, l'une des plus populaires est Max-Pooling. Cette couche applique un petit filtre (2×2 par exemple) avec un pas de 2 sur la hauteur et la largeur du volume d'activation et sous-échantillonne les activations avec les valeurs les plus élevées (par exemple cette couche ne sélectionnera que 25% de ces valeurs) afin de créer un nouveau volume plus petit à partir de ces valeurs.

Pour un volume d'entrée de dimensions $W_e \times H_e \times D_e$, S le pas du filtre de pooling et F la taille du filtre, les dimensions du volume de sortie sont décrites dans les équations ci-dessous:

$$W_s = \frac{W_e - F}{S} + 1 \tag{2.6}$$

$$H_s = \frac{H_e - F}{S} + 1 \tag{2.7}$$

$$D_s = D_e \tag{2.8}$$

La Figure 2.9 illustre une utilisation d'un filtre de Max-Pooling.

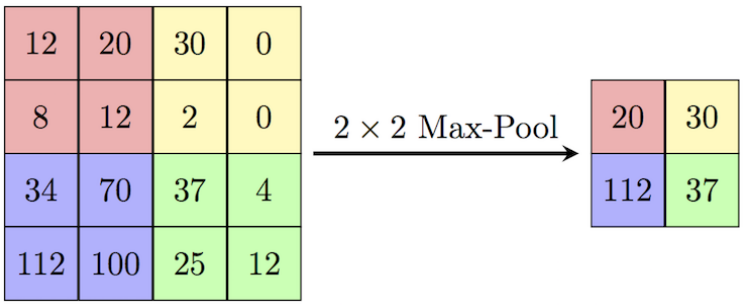

Figure 2.9: Exemple de Max-Pooling [28].

Couche de normalisation

Il s'est avéré que l'entraînement de réseaux convolutifs très profonds pouvait prendre beaucoup de temps et était souvent sujet à des gradients divergents entraînant l'échec de l'entraînement. Un autre problème était que les réseaux particulièrement profonds avaient tendance à provoquer une "disparition

du gradient", c'est-à-dire que le gradient des neurones dans les couches les plus profondes était si faible qu'au fur et à mesure de la formation, il finissait par devenir nul. L'avantage d'utiliser un réseau plus profond pour augmenter le nombre de paramètres "apprenables" afin d'accroître la précision est dans ce cas compromis. Ce problème a été abordé dans [29] qui introduit la notion d'une couche de normalisation entre chacune des couches de convolution (voir Figure 2.10) afin d'éviter le problème d'explosion ou de disparition du gradient, même dans les couches les plus profondes du réseau. La couche BatchNorm introduite dans ce travail est maintenant largement utilisée dans tous les réseaux convolutifs depuis des années. Si nous définissons le volume d'entrée comme ayant des dimensions (N, C, H, W) avec N la taille du *batch* (nombre d'entrées), C la profondeur du volume, H la hauteur et W la largeur du volume. La sortie du réseau est décrite dans l'équation ci-dessous:

$$\mathbf{y} = \frac{\mathbf{x} - E[\mathbf{x}]}{\sqrt{Var[\mathbf{x}] + \epsilon}} \times \gamma + \beta \tag{2.9}$$

La moyenne et l'écart-type de l'entrée sont ici calculés par dimension sur le nombre d'entrées (*batch*) et γ et β sont deux vecteurs de taille C de paramètres "apprenables" par le réseau. ϵ est ici utilisé pour assurer la stabilité numérique de l'opération et est donc défini à une très petite valeur proche de 0.

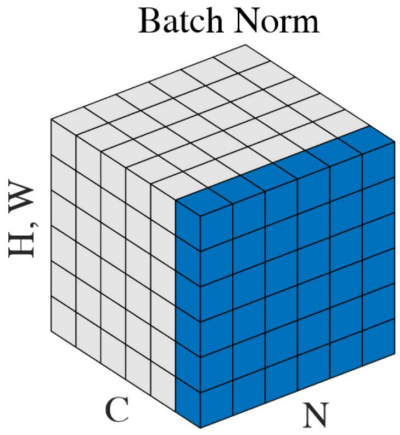

Figure 2.10: Couche de normalisation BatchNorm. H,W,C correspondent à la hauteur,largeur et profondeur respectivement tandis que N correspond au nombre d'échantillons [30].

Couche d'activation

Les réseaux convolutifs sont pour la majorité des cas utilisés pour résoudre un problème non linéaire et difficilement modélisable. Il est donc nécessaire de rendre ce réseau non linéaire en ajoutant des couches d'activation. La couche d'activation va introduire des propriétés non-linéaires à un réseau CNN donné (partant du simple constat que la plupart des données de l'environnement autour du véhicule ne sont pas linéaires). Elle applique une fonction non-linéaire pour chaque élément du volume d'entrée. La fonction d'activation la plus courante est la fonction ReLU [25]. C'est une fonction simple, si nous définissons x comme un élément du volume d'entrée et y comme l'élément correspondant du volume de sortie, la fonction ReLU peut être décrite comme dans l'équation (2.10):

$$y = max(0, x) \tag{2.10}$$

Mais on peut aussi noter d'autres fonctions comme la fonction SiLU (*Sigmoid Linear Unit*) [31] décrite

dans l'équation(2.11):

$$y = x \cdot sigmoid(x) \tag{2.11}$$

Avec $sigmoid$ la fonction sigmoide définie par la Equation (2.12):

$$y = \frac{1}{1 - e^{-x}} \tag{2.12}$$

On peut encore trouver la fonction Mish [32] décrite dans l'équation(2.13):

$$y = x \cdot tanh(softplus(x)) \tag{2.13}$$

où $tanh$ est la tangente hyperbolique et $softplus$ est la fonction softplus [33] décrite dans l'équation (2.14):

$$softplus = ln(1 + e^{x}) \tag{2.14}$$

Toutes ces fonctions d'activation sont illustrées dans la Figure 2.11. Le choix de la fonction d'activation à utiliser dans un réseau se fera principalement après avoir testé chacune d'entre elles en fonction de l'architecture du réseau et du type d'application car il n'existe actuellement, au sein de la communauté scientifique, aucun consensus sur la meilleure fonction.

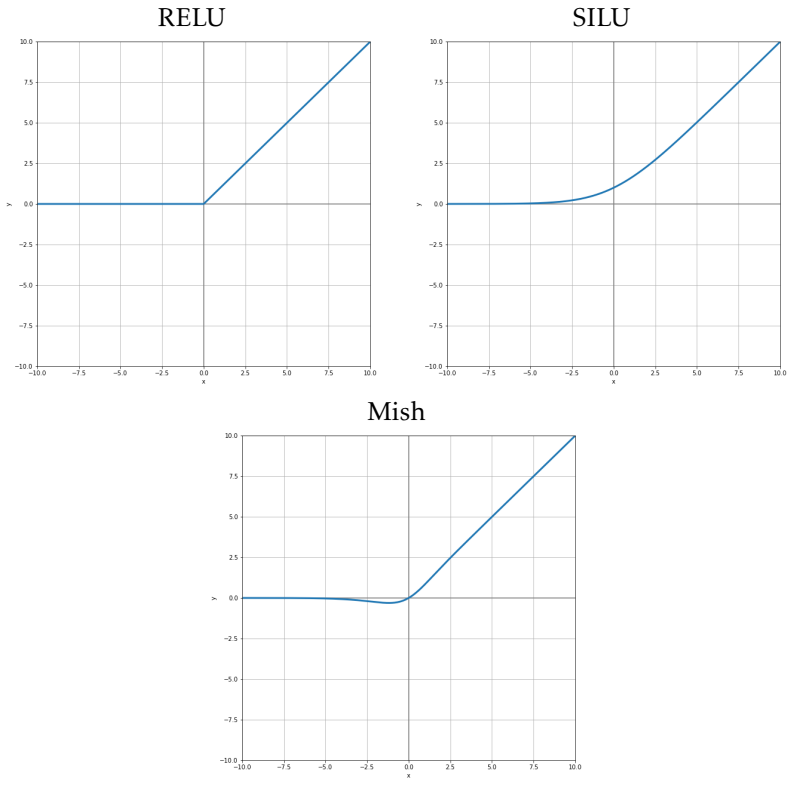

Figure 2.11: Fonction d'activations.

2.1.3 Extraction des caractéristiques de régions de l'image

L'extraction des caractéristiques de régions spécifiques de l'image à partir d'un réseau convolutif a été introduite pour la première fois par le Fast-RCNN [34] pour la détection d'objets dans l'image avec la

fonction RoI Pooling. Le principe de cette technique est d'obtenir uniquement les caractéristiques d'une région de l'image afin de pouvoir les donner en entrée aux branches suivantes du réseau afin de faire une prédiction spécifique sur cette région. La dernière itération de cette méthode est la fonction RoI Align proposée dans Mask-RCNN [35]. Si nous prenons la Figure 2.12 comme exemple, et que nous cherchons à prédire la classe d'un objet particulier dans l'image (ici le chat), nous devons d'abord prédire où se trouvent toutes les régions d'intérêt (RoI) contenant des objets dans l'image. Si plusieurs objets sont présents dans l'image, nous devrons sélectionner uniquement les caractéristiques provenant de l'objet que nous voulons classer. Avec l'information de la région d'intérêt de l'objet, nous avons deux choix pour y parvenir:

(1) Modifier l'image d'entrée pour ne garder que l'objet et l'envoyer au début du réseau pour extraire les caractéristiques.

(2) Utilisez la fonction RoI Align pour extraire les caractéristiques des objets directement de la carte des caractéristiques.

La première solution est loin d'être idéale car elle nécessite le calcul d'une nouvelle carte de caractéristiques pour chaque objet. La solution introduite par Fast-RCNN permet d'obtenir les caractéristiques spécifiques d'un objet dans l'image à moindre coût.

Ainsi, si nous prenons l'exemple de la Figure 2.12, notre image originale est de résolution 512×512 et la carte de caractéristique 16×16. La taille de l'image a donc été réduite d'un facteur de 32. Maintenant, si notre région d'intérêt est de résolution 200×145, sur la carte de caractéristiques, cette région va être de largeur 200/32 = 6.25 et de hauteur 145/32 = 4.53. Cette nouvelle région ne peut donc pas s'aligner avec la grille de la carte de caractéristiques.

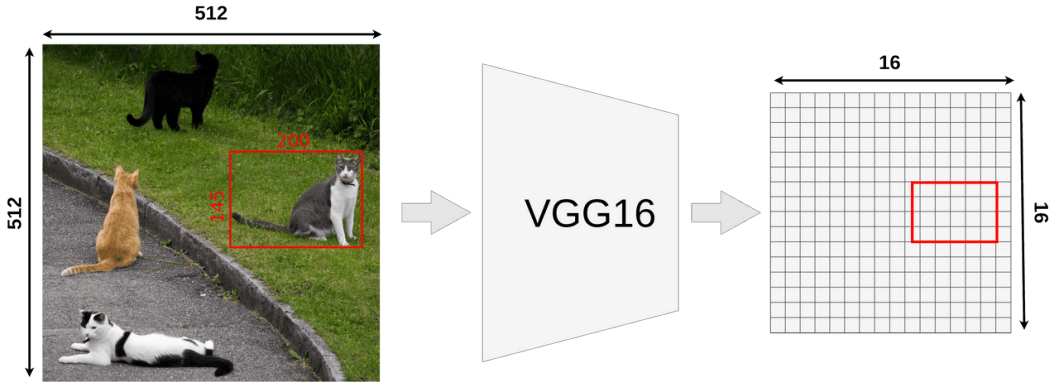

Figure 2.12: Extraction des caractéristiques du chat présent dans l'image. Ici VGG-16 [36] est un réseau convolutif utilisé pour obtenir la carte de caractéristique de l'image [37].

La fonction RoI Align permet d'éviter ce problème de quantification et d'obtenir pour chaque RoI une nouvelle carte de caractéristiques dont la taille est choisie par l'utilisateur. Par exemple, si nous souhaitons obtenir des cartes de caractéristiques de taille 3×3, la fonction RoI Align profitera de l'interpolation bilinéaire. Nous définissons d'abord 4 points d'échantillonnage pour chaque cellule de notre nouvelle carte de caractéristiques (Figure 2.13).

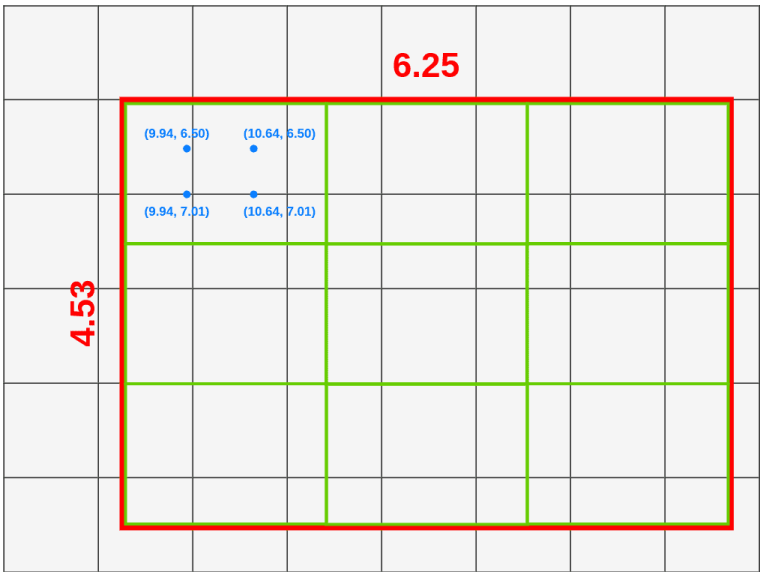

FIGURE 2.13: Distribution des points d'échantillonnages [37].

Si on pose X_{box} et Y_{box} les coordonnées du point en haut à gauche de la cellule, chacun des points d'échantillonage peut être trouvés grâce aux formules (2.15) et (2.16):

$$X_{i,j} = X_{box} + (W/N_{cell}) * i \tag{2.15}$$

$$Y_{i,j} = Y_{box} + (H/N_{cell}) * j \tag{2.16}$$

avec W, H la taille de la cellule et $N_{cell} = 3$ la dimension de la carte de caractéristiques souhaitée.

On peut ensuite appliquer une interpolation bilinéaire afin d'échantillonner pour chacun des points la carte de caractéristiques (Figure 2.14)

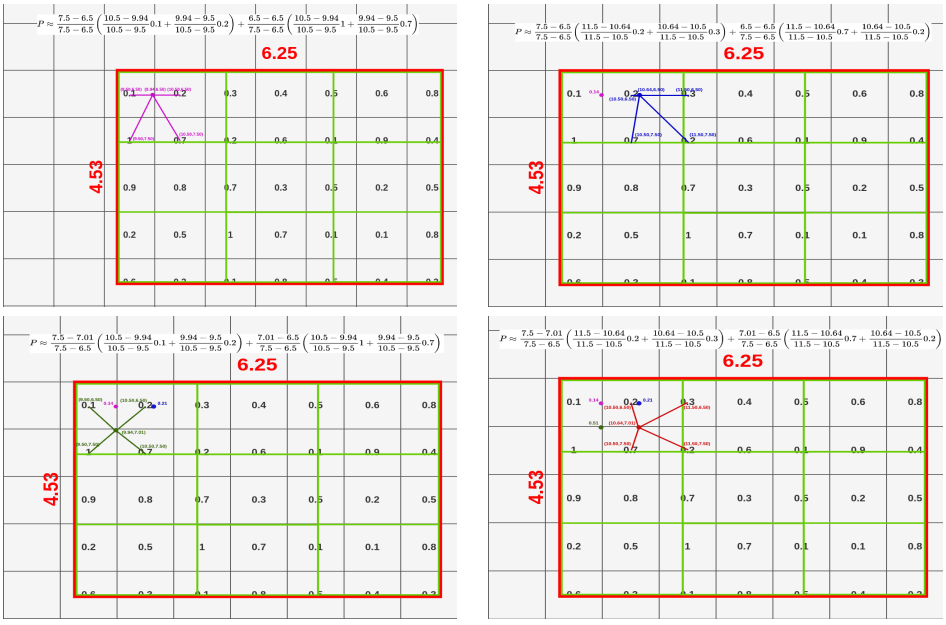

FIGURE 2.14: Interpolation bilinéaire sur chacun des points d'échantillonages de la cellule [37].

Enfin, afin de trouver la valeur de la cellule, on utilise une fonction de Max Pooling (voir Figure 2.15).

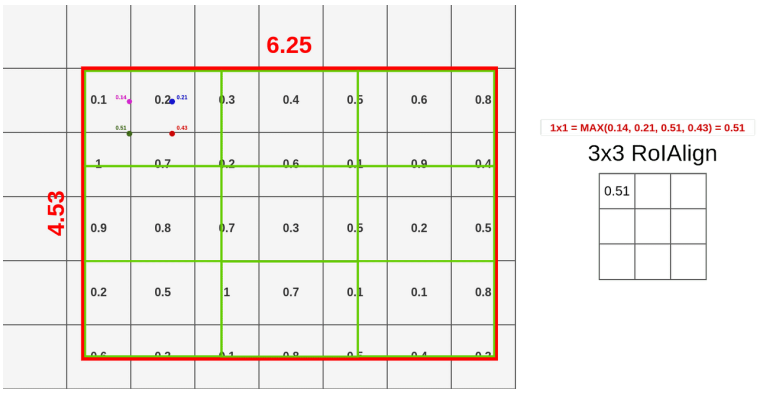

Figure 2.15: Calcul de la valeur de la première cellule [37].

Ce processus est appliqué pour chacune des cellules afin d'obtenir finalement les caractéristiques extraites de la région avec les dimensions choisies. Cette nouvelle carte de caractéristiques peut alors être donnée en entrée à une branche de notre réseau afin d'extraire une prédiction sur la classe de l'objet (ici un chat).

2.2 Matériel et jeux de données

Avant d'exposer de nos contributions et les résultats obtenus dans cette thèse, il est important de présenter dans un premier temps le matériel que nous avons utilisé, les bases de données que nous utiliserons pour tester et évaluer les différentes approches et systèmes d'acquisition pour les tests en situation réelle.

2.2.1 Matériels informatique

L'apprentissage profond nécessite des ressources informatiques importantes et du temps pour fonctionner. Afin de fonctionner avec un temps de calcul raisonnable, il est nécessaire de disposer d'une carte graphique (GPU) avec suffisamment de mémoire vidéo (VRAM) pour supporter les lourdes charges imposées par les modèles d'apprentissage profond. Lors de l'entraînement de ces modèles, les demandes de VRAM sont décuplées, d'où la nécessité de disposer de serveurs de calcul pour obtenir un entraînement optimal.

Pour notre formation, nous nous sommes donc tournés vers le supercalculateur MYRIA [1]. (voir Figure 2.16) du Centre Régional Informatique et des Applications Numériques de Normandie (CRIANN). Celui-ci dispose de 66 nœuds de calcul bi-processeurs Broadwell (28 cœurs à 2,4 GHz, 128 Go de RAM DDR4), dont 20 nœuds dédiés aux calculs de Deep Learning, chacun étant équipé de 4 GPU Nvidia Kepler K80 (12 Go de VRAM par GPU) ou 2 GPU Nvidia Kepler P100 (12 Go de VRAM par GPU) ou 4 GPU Nvidia V100 (32 Go de VRAM par GPU). Chaque utilisateur dispose de 50 Go d'espace de stockage. Cette puissance de calcul nous permettra d'entraîner des modèles d'apprentissage profond sans être limité par la taille des modèles sur de grandes bases de données et ce, relativement rapidement (48 heures).

1. urlhttps://www.criann.fr/renouvellement-2016/

Figure 2.16: Supercalculateur Myria du CRIANN.

2.2.2 Jeux de données

La deuxième contrainte la plus importante dans le domaine de l'apprentissage profond est l'acquisition de bases de données à grande échelle nécessaires pour l'entraînement et le test des performances des algorithmes proposés par rapport à l'état de l'art. En raison de l'importance de la conduite autonome, il existe de nombreux jeux de données dédiés au domaine routier, tels que KITTI [38] qui fournit des images provenant d'une caméra stéréoscopique, la profondeur de la scène mesurée par un Lidar Velodyne ainsi que des annotations de véhicules et de piétons pour la détection d'objets. De nombreux autres ensembles de données similaires sont également disponibles, comme NuScenes [39] qui dispose de 6 caméras, d'un Lidar et d'une plus grande quantité de données qui peuvent être utilisées comme vérité terrain pour des tâches telles que la détection d'objets en 2D et 3D, pour l'estimation de la profondeur, le suivi en 2D et 3D, etc.etc.. On peut également trouver des bases de données telles que CityScapes [40], Pascal VOC [41], MS-COCO [42], ImageNet [43]. OpenImages [44].

Pour évaluer et entraîner nos approches, nous utiliserons principalement KITTI et Nuscenes.

2.2.3 Système d'acquisition

Pour les tests en conditions réelles, nous utilisons des caméras Intel® Realsense D435. Ces capteurs ont la capacité de fournir ce qu'on appelle une carte de profondeur fournissant des informations 3D liées à la distance des surfaces des objets filmés depuis un point de vue spécifique. Ces caméras ont été utilisées dans [45] pour faire du SLAM (Simultaneous Localization and Mapping) visuel. Ce capteur visuel a la capacité de fournir des résultats fiables lorsqu'il est utilisé à l'intérieur, mais n'a pas été testé dans des conditions extérieures.

2.2.4 Librairies d'apprentissage profond

Nous utilisons plusieurs librairies disponibles dans le domaine de l'apprentissage profond. Parmi les bibliothèques les plus importantes, nous pouvons mentionner TensorFlow qui est la bibliothèque open-source la plus célèbre. Elle a été développée en Python et C++ par Google. De même, Keras est une

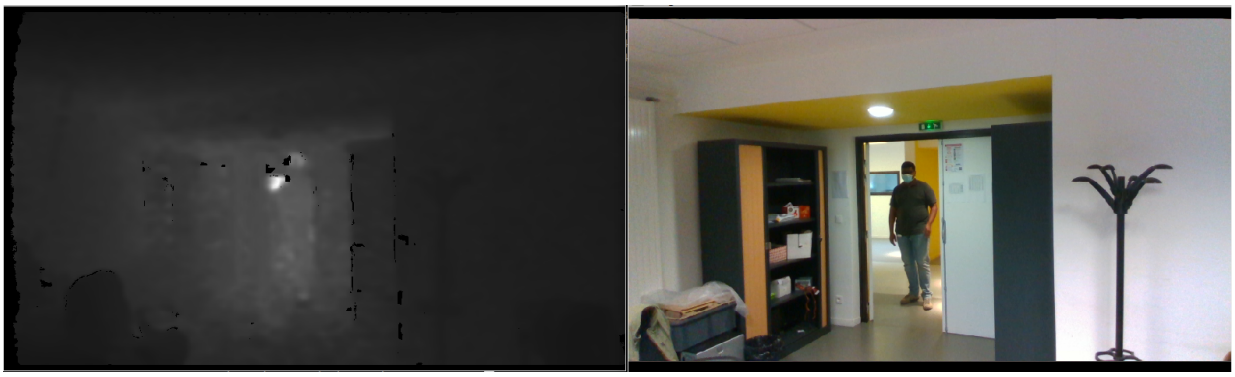

Figure 2.17: Carte de profondeur fournie par la caméra Intel® Realsense D435.

bibliothèque open-source d'apprentissage profond disponible également en Python. Cette dernière offre la possibilité de créer des réseaux neuronaux de haut niveau. Elle est l'une des plus faciles à comprendre et à utiliser, et elle permet de créer rapidement des prototypes de réseaux. Cependant, elle est moins efficace que d'autres bibliothèques récentes, par exemple Pytorch qui est une bibliothèque open-source de haut niveau qui a été développée pour s'intégrer de manière transparente avec d'autres modules Python comme Numpy, SciPy et Cython. Cette meilleure intégration rend cette bibliothèque facile à utiliser tout en offrant de très bonnes performances.

Chapitre 3
Détection, Localisation et Suivi d'Objets

3.1 Introduction

Dans ce chapitre, nous allons nous pencher sur notre première approche pour la détection d'objets et la prévention de collision par vision. Afin d'atteindre cet objectif, nous avons identifié trois tâches à accomplir:

(1) Détecter les objets et les localiser dans l'image
- Trouver la position des objets dans l'image
- Estimer la classe de l'objet (voiture, piéton, etc.)

(2) Localiser les objets dans l'espace en 3D
- Position de l'objet dans un repère (XYZ) en mètres vis-à-vis du véhicule

(3) Prédire le comportement des objets
- Suivre l'objet afin de prédire sa position dans les images suivantes
- Alerter si l'objet se dirige sur la voie du véhicule afin de prévenir une collision

Le fait d'utiliser uniquement une caméra comme capteur réduit considérablement la complexité de la solution, et facilitera son application en conditions réelles. Cette solution doit pouvoir être utilisée sur tout type de caméra. Pour ce faire, nous nous concentrerons sur les approches basées sur l'apprentissage profond. Le principal problème que nous devrons résoudre concerne à la fois la précision d'une telle méthode, notamment pour la partie localisation, qui sans capteurs de profondeur dédiés (RADAR, LiDAR) est difficile, mais aussi pour répondre aux besoins liés au temps réel, indissociable d'une utilisation en conditions réelles.

Il est crucial de répondre à ces questions car la fiabilité de la prédiction du comportement des objets sera directement impactée par la fiabilité de la détection et de la localisation. Une absence de détection d'un piéton, ou une mauvaise localisation de celui-ci, ou encore un délai trop long entre les prédictions pourrait conduire soit à de fausses alarmes, soit à un accident aux conséquences graves.

Heureusement, ces questions sont directement liées au développement du véhicule autonome et de nombreux travaux, tant de la part de pairs que d'entreprises, ont déjà été réalisés pour y répondre. Nous disposons donc d'une abondance de bases de données et d'approches pour la voiture. Cependant, comme nous le verrons dans la section suivante sur l'état de l'art, il n'existait à l'époque aucune solution répondant directement à nos attentes qui se concentrait soit sur la détection d'objets, soit sur l'estimation de la distance, soit sur le suivi des objets.

Cette première solution que nous avons créée va tirer profit de ces algorithmes et les combiner afin de répondre à chacun des objectifs fixés pour arriver à un algorithme complet de suivi d'objets en 3D. Le but d'une telle démarche est de créer une première solution qui servira de base de comparaison pour nos futures méthodes. Mais surtout, elle nous permettra d'identifier les limites et les défis que nous devrons surmonter afin de mieux cibler les contributions possibles.

Nous commencerons donc par identifier les meilleures approches de l'état de l'art, en prenant soin de trouver le meilleur compromis entre temps de calcul et précision. Pour la détection d'objets, nous étudierons les nombreuses approches d'apprentissage profond déjà proposées. Pour la localisation d'objets, nous étudierons les approches d'estimation de la profondeur de chaque pixel de l'image. En combinant les résultats de la détection d'objets et de l'estimation de la profondeur, nous serons en mesure de

déterminer la position des objets dans l'espace 3D. Enfin, nous analyserons les solutions de suivi et de prédiction des trajectoires pour nous permettre de concevoir notre propre algorithme de suivi d'objets en 3D détaillé dans la Figure 3.1.

Afin de faire notre choix de méthode, nous effectuerons une évaluation de chaque méthode en utilisant à la fois les métriques utilisées dans l'état de l'art. Après avoir sélectionné les méthodes les plus adaptées, nous nous concentrerons sur la conception d'un algorithme de suivi d'objets 3D afin de prédire leur trajectoire et ainsi pouvoir alerter en cas de danger de collision.

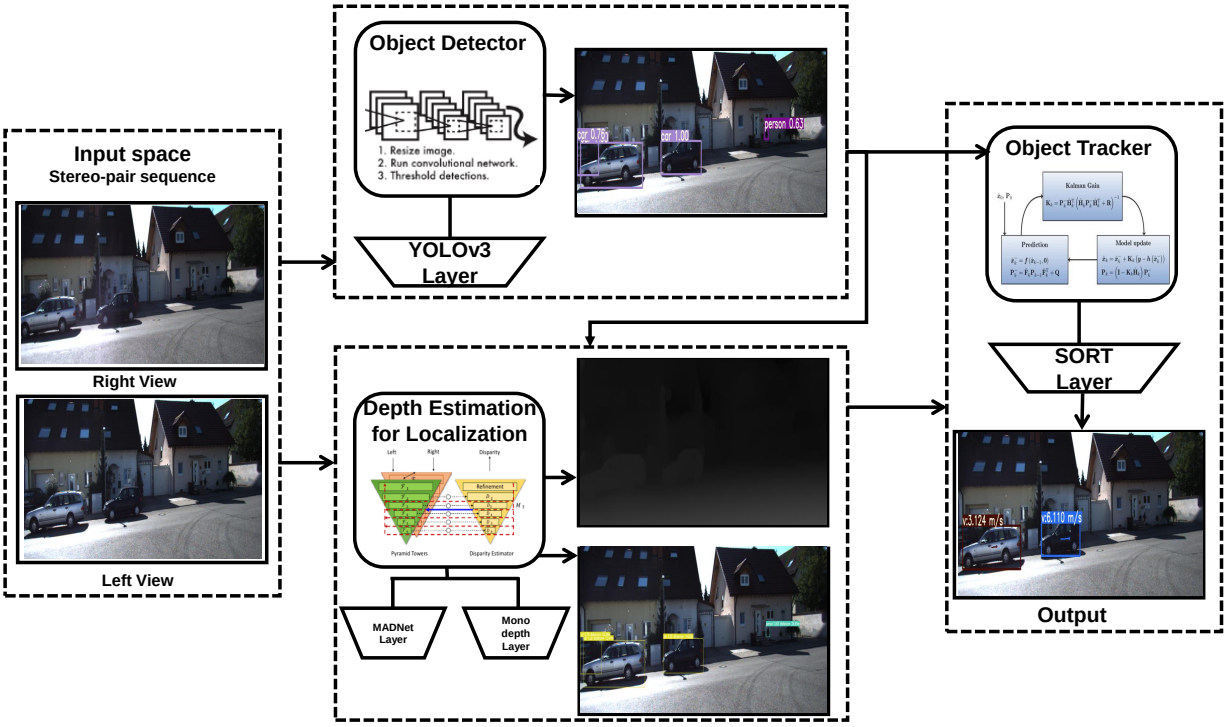

FIGURE 3.1: Vue d'ensemble du système proposé, composé de trois éléments principaux: détection, estimation de la distance pour la localisation et suivi d'objets.

3.2 Etat de l'art

Afin de concevoir cette première solution de détection et de suivi d'objets en temps réel, il est nécessaire de réaliser un état de l'art afin d'identifier les solutions potentielles et leurs limites. Nous nous concentrons ici sur la détection d'objets, l'estimation de la profondeur et le suivi d'objets par apprentissage profond.

3.2.1 Détection d'objets

La détection d'objets est un problème clé de la vision par ordinateur. On peut décomposer cette tâche en deux sous-tâches: la prédiction de la région de l'image où se trouve un objet par régression et l'identification de la classe de l'objet (voiture, personne, camion, etc.) par classification.

Ces dernières années, plusieurs méthodes basées sur les réseaux de neurones convolutifs (CNN) ont été proposées pour répondre à ces attentes.

Les méthodes basées sur les CNN peuvent être divisées en deux catégories principales:

— Les méthodes à une étape, qui fournissent une estimation de la position de l'image et une prédiction de la classe de l'objet en une seule étape.

— Les méthodes à deux étapes, qui détectent d'abord les régions de l'image où un objet pourrait être présent, puis utilisent un classificateur sur ces régions pour extraire une prédiction de la classe de l'objet.

Méthodes à étage unique

Parmi les méthodes de détection d'objets dans une image en une seule étape, on trouve la méthode SSD (Single-Shot-Detector) [46]. Cette méthode introduit la notion de " boîtes par défaut " qui servent de références pour la prédiction de la position de l'objet dans l'image, ce qui facilite la prédiction puisque le réseau doit simplement estimer la différence entre la " boîte par défaut " et la boîte englobante au lieu d'estimer directement les coordonnées de la boîte englobante de l'objet. SSD utilise 4 "boîtes par défaut" avec différentes dimensions et différents ratios sur chacune des boîtes de la carte des caractéristiques (8x8 et 4x4) provenant de différentes couches du CNN. Chacune de ces "boîtes par défaut" est présentée sous la forme d'un vecteur intégrant les déviations sur la position et la dimension de la boîte mais aussi la probabilité qu'un objet soit situé dans la boîte et les probabilités liées à la classe d'objets. Ce vecteur est détaillé dans l'équation (3.1):

$$b = \begin{bmatrix} cx & cy & w & h & c_1 & etc. & c_N \end{bmatrix} \tag{3.1}$$

Avec b le vecteur représentant la "boîte par défaut", cx et cy la distance entre le centre de la boîte englobante prédite et le centre de la boîte par défaut et $c_1 etc. c_N$ le score de classe de l'objet avec N le nombre de classes. L'architecture du SSD est uniquement composée de couches convolutionnelles et ne comporte pas de couches entièrement connectées comme la plupart des CNN pour la détection d'objets. Ces couches sont généralement plus coûteuses en temps de calcul et leur suppression réduit donc le temps de calcul de SSD. Parmi les méthodes de détection en une étape, on trouve également YOLO [47]. Cette méthode utilise le même principe que la SSD pour effectuer la détection. YOLO utilise des "boîtes ancrées" similaires aux "boîtes par défaut" de la SSD. Le vecteur définissant ces "boîtes ancrées" a un score pour définir la probabilité qu'un objet se trouve à l'intérieur de la boîte. Cette première version de YOLO utilise une architecture qui mélange des couches convolutionnelles et des couches entièrement connectées. Les versions ultérieures de YOLO (YOLO 9000 [48] , YOLOv3 [14]) apportent des améliorations significatives à l'architecture du réseau neuronal en remplaçant les couches entièrement connectées pour transformer l'architecture en un réseau composé uniquement de couches convolutionnelles. Ces nouvelles versions améliorent également les fonctions de perte utilisées et l'augmentation des données pendant la formation. Grâce à ces améliorations, la précision de YOLOv3 surpasse celle des SSD tout en ayant un temps de calcul inférieur. Un exemple d'architecture typique de réseau à un étage pour la détection d'objets est illustrée dans la Figure 3.2.

Méthodes à deux étages

Les méthodes à deux étages, comme leur nom l'indique, comportent deux étapes. La première étape du réseau est utilisée pour obtenir une première prédiction de la position et de la taille des boîtes englobantes des objets de l'image. Les caractéristiques de cette région d'intérêt (RoI) sont extraites (à l'aide de RoI Pooling ou RoI Align) pour créer une nouvelle carte de caractéristiques qui servira d'entrée à la deuxième étape du réseau. Cette deuxième étape permettra au réseau d'affiner la prédiction de la boîte englobante de l'objet et de prédire la classe de l'objet.

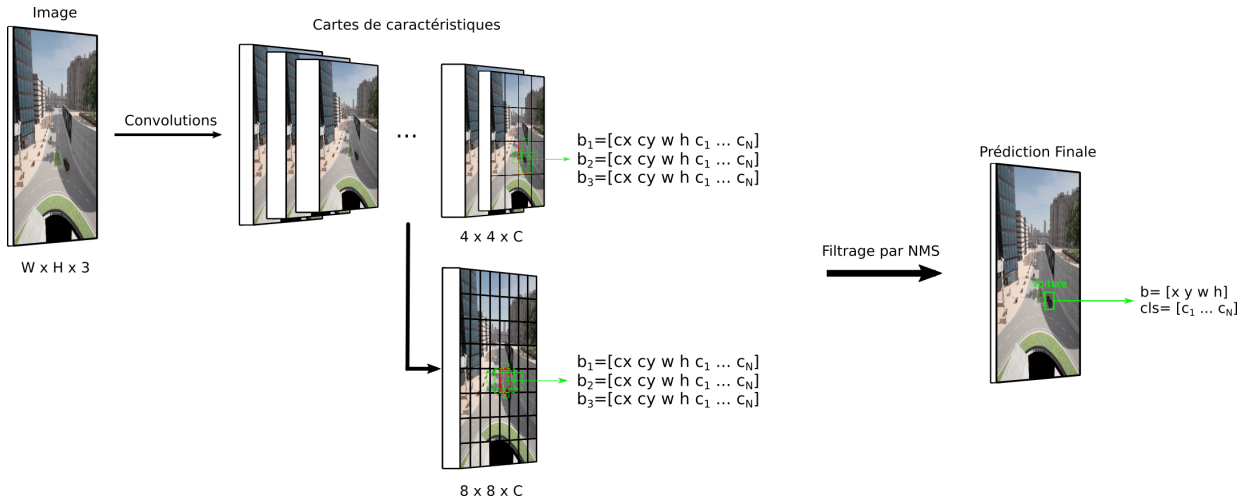

FIGURE 3.2: Architecture typique d'un CNN à un seul étage pour la détection d'objets. Une image de dimensions ($W \times H \times 3$) est utilisée comme entrée du réseau. Des couches convolutionnelles successives transforment cette image en cartes de caractéristiques de différentes dimensions. Les cartes de caractéristiques de différentes dimensions (ici 4×4 et $8 \times 8 \times C$) sont utilisées pour obtenir les "boîtes de défaut" (ou "boîtes ancrées"). Les prédictions des "boîtes par défaut" pour chacune des cartes de caractéristiques sont combinées et filtrées à l'aide d'un filtre de non-maximum-supression (NMS) pour obtenir la prédiction finale.

Ces méthodes offrent une grande précision grâce à leur architecture en deux étapes, mais au prix d'un temps de calcul élevé. Un exemple d'architecture à deux étages est illustré à la Figure 3.3.

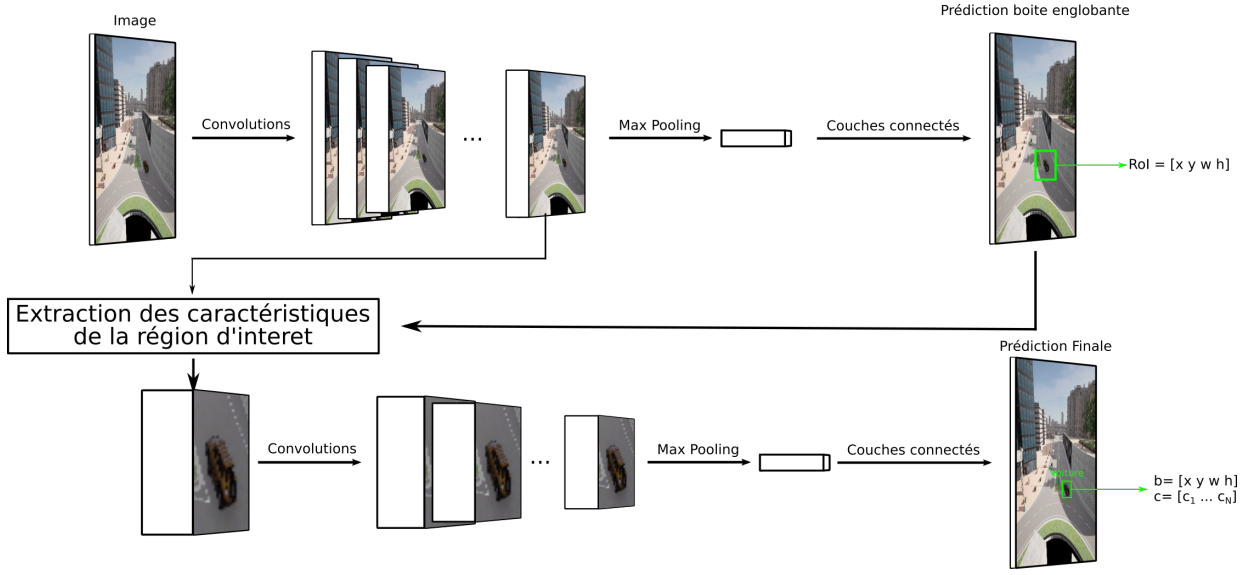

FIGURE 3.3: Architecture typique d'un CNN à deux étages pour la détection d'objets.

Dans la catégorie à deux étages, le RCNN (Region-proposal CNN) [49] présente des résultats exceptionnels sur de nombreux benchmarks ainsi que ses versions améliorées [34, 50]. Les résultats sont excellents en termes de précision, notamment pour la localisation d'objets dans l'image.

3.2.2 Estimation de la distance

Pour estimer la distance entre le véhicule et les objets de la scène, de nombreux capteurs sont disponibles, par exemple les capteurs laser-ultrasons [5] ou les capteurs à temps de vol [6]. Ces types de capteurs sont largement utilisés dans ce contexte, mais l'utilisation de plusieurs types de capteurs rend

le système plus complexe et plus coûteux, d'où l'intérêt de se concentrer sur les méthodes d'obtention de la distance à partir d'images. Afin de répondre à cette problématique, nous allons nous intéresser aux méthodes d'estimation de la profondeur à partir d'images. Ces CNNs peuvent, à partir d'une seule image (unique ou stéréoscopique), obtenir la profondeur de chaque pixel, c'est-à-dire la position du pixel projeté sur l'axe Z du référentiel 3D de la caméra.

FIGURE 3.4: Images de profondeur du jeu de données KITTI [38]. Les images RVB se trouvent à gauche et les images de profondeur à droite.

Ces algorithmes peuvent renvoyer soit des cartes de disparité, soit des cartes de profondeur (voir la Figure 3.4). Une carte de disparité peut être transformée en une carte de profondeur en utilisant l'Equation (3.2):

$$D_{i,j} = \frac{f \times b}{d_{i,j}} \tag{3.2}$$

Avec i, j les coordonnées du point de la carte, D la profondeur en mètres, f la distance focale de notre caméra en mètres, b la distance entre les deux caméras (réelles ou virtuelles) en mètres (*Baseline*) et d la disparité.

Il est important de noter que même les méthodes qui ne travaillent que sur des images uniques peuvent retourner une carte de disparité, auquel cas b est une valeur fixée dans l'algorithme. Nous pouvons donc toujours utiliser la relation (3.2) en utilisant cette valeur.

Pour apprendre à prédire les cartes de profondeur (ou cartes de disparité), ces CNN ont besoin d'une vérité de base pour régresser ces cartes. Cela signifie que pour chaque pixel de l'image, il faut avoir une vérité de base de la profondeur (ou de la disparité), ce qui rend l'acquisition de ces bases de données complexe. Toutes ces bases de données utilisent un capteur LiDAR couplé à une caméra pour extraire ces informations. Pour toutes ces raisons, la disponibilité de ces bases de données est limitée, ce qui constitue un frein à l'entraînement de ce type d'approche.

La grande majorité de ces CNNs sont basés sur une architecture encodeur-décodeur inspirée de U-net [51] qui a introduit la notion de " sablier inversé ". Un schéma explicatif de cette architecture spécifique se trouve dans la Figure 3.5.

Les méthodes que nous avons étudiées se divisent en deux catégories: celles qui ne nécessitent que des images provenant d'une caméra monoculaire (c'est-à-dire tout type de caméra) et celles qui nécessitent

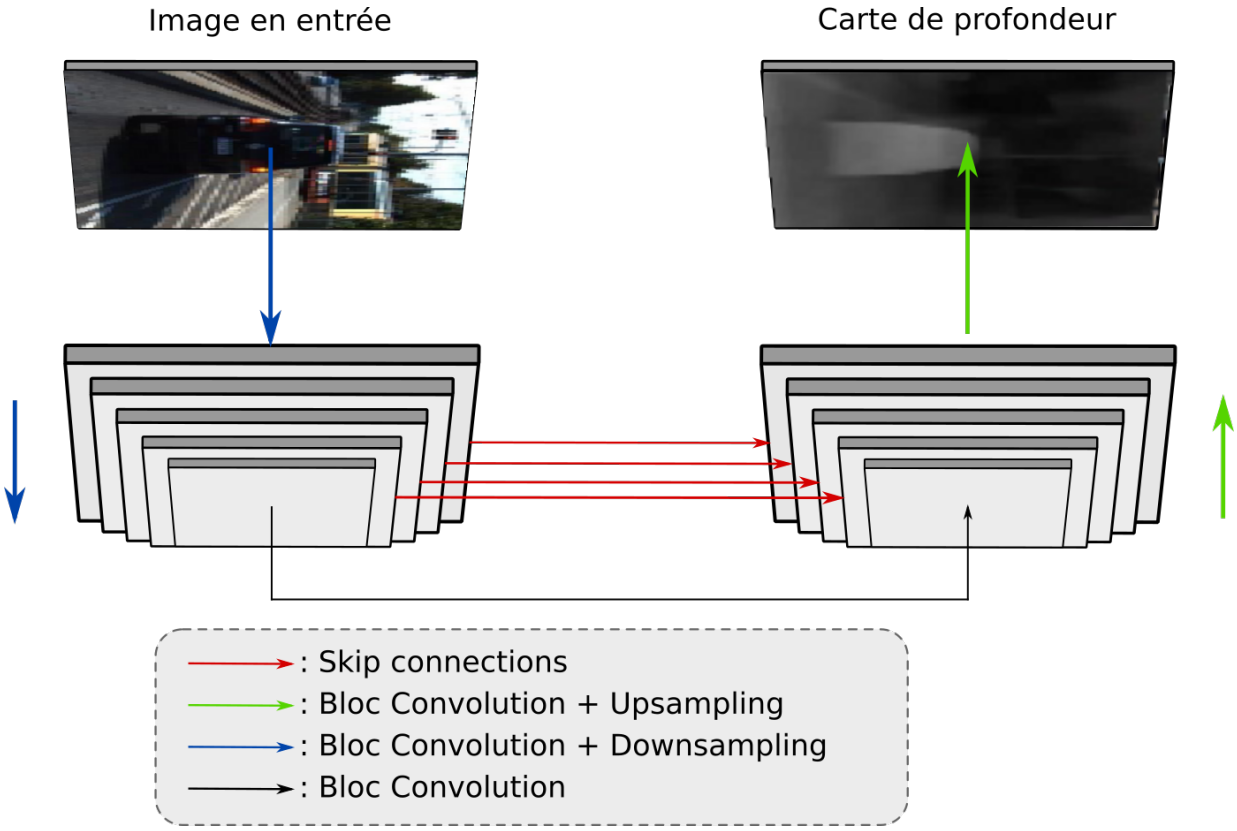

FIGURE 3.5: Architecture en "sablier inversé" pour l'estimation de profondeur à partir d'images uniques.

une image provenant d'une caméra stéréoscopique.

Méthodes monoculaires

Dans [52], un CNN basé sur une architecture entièrement convolutive pour l'estimation de la profondeur à partir d'images RVB d'une scène donnée est présenté. La correspondance équivoque entre les images monoculaires et les cartes de profondeur est modélisée par apprentissage résiduel. L'optimisation du modèle est réalisée par RHL (*Reverse Huber Loss*). L'approche fonctionne en temps réel sur des images et des vidéos. Dans SfmLearner [53], Zhou *et al.* présentent un cadre d'apprentissage non supervisé dédié à la fois à la tâche d'estimation de profondeur monoculaire CNN à vue unique et à l'estimation de l'ego-motion de la caméra en utilisant des séquences vidéo non structurées. Les réseaux de profondeur monoculaire et de pose multi-vues sont utilisés dans ce cadre. Basée sur Dispnet, cette méthode exploite une architecture codeur-décodeur avec des connexions à saut et des prédictions latérales multi-échelles. Le modèle est validé sur le jeu de données Cityscape [54]. Dans Monodepth2 [55], les auteurs proposent un CNN entraîné pour l'estimation de la profondeur d'une seule image sans supervision de la vérité du sol. L'entraînement monoculaire non supervisé de bout en bout est effectué en utilisant une perte d'entraînement qui renforce la cohérence de la profondeur de gauche à droite. L'architecture CNN est également inspirée de DispNet. Les détails de plus haute résolution sont récupérés en utilisant des connexions de saut entre les blocs d'activation de l'encodeur. Deux cartes de disparité sont prédites: de gauche à droite et de droite à gauche. Le modèle est validé sur le jeu de données KITTI [38]. MonoResMatch [56] est une autre méthode pour prédire la profondeur à partir d'une seule image d'entrée. Pour ce faire, cette approche propose de simuler une caméra stéréoscopique en synthétisant les caractéristiques d'un point de vue différent, aligné horizontalement avec l'image d'entrée. Une correspondance entre ces deux points de vue peut alors être effectuée pour extraire une carte de disparité à partir de laquelle la

carte de profondeur peut être déduite.

Méthodes stéréoscopiques

Dans un premier temps, nous avons choisi, pour l'état de l'art des méthodes d'estimation de la profondeur à partir d'images stéréoscopiques, d'étudier à la fois les méthodes basées sur l'apprentissage profond qui sont prometteuses, mais aussi les méthodes dites " classiques " qui ne l'utilisent pas. Cette étude nous permettra de mettre en évidence les avantages de ces nouvelles méthodes d'apprentissage profond par rapport aux anciennes et quels sont les compromis à faire.

En ce qui concerne les méthodes "classiques", nous utiliserons les algorithmes implémentés dans la bibliothèque de traitement d'images OpenCV [57]. La Figure 3.6 présente une manière intuitive de comprendre leur fonctionnement.

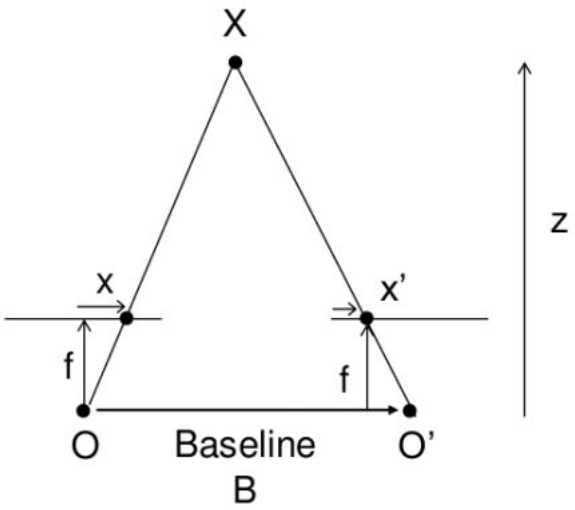

Figure 3.6: Estimation de profondeur en utilisant une caméra stéréoscopique [58].

Ici X est le point 3D dont on cherche la profondeur Z, x et x' sont la distance entre les points du plan image correspondant au point de la scène 3D et leur centre caméra (la différence $x - x'$ est ce que nous avons appelé la disparité) Où B est la distance entre deux caméras (que nous connaissons) et f est la distance focale de la caméra (que nous connaissons également). Nous pouvons utiliser la formule 3.2 définie précédemment pour déduire la profondeur de tous les pixels d'une image. La tâche effectuée par l'approche consiste donc à calculer la correspondance stéréo à l'aide de l'algorithme de correspondance par blocs pour trouver respectivement x et x'. Cette approche fonctionne bien lorsque l'environnement est suffisamment texturé pour que la correspondance fonctionne, mais elle rencontre des difficultés importantes lorsque ce n'est pas le cas, ce qui entraîne l'apparition de "blobs" dans la carte de profondeur. Pour pallier partiellement ce problème, il est possible d'utiliser un filtre WLS (*weighted least squares*) qui permet d'affiner les résultats en cas de demi-occlusion (lorsque seulement 1 caméra peut voir le point X) ou si la zone est uniforme.

Parmi les approches basées sur l'apprentissage profond, nous avons identifié MADNet [15] qui est une méthode permettant de faire de l'entraînement et de l'inférence en même temps afin de s'adapter à l'environnement. Ce type d'approche avait déjà été proposé avec DispNetC [59], mais l'apport principal de MADNet est son mode MAD qui permet de n'entraîner qu'une fraction du réseau de convolutif pendant l'adaptation afin de rendre cette opération possible en temps réel.

Cette adaptation permet à ces méthodes de compenser la perte de précision lorsque l'environnement diffère trop de la base de données utilisée lors de l'entraînement. Ceci est particulièrement intéressant

pour notre application car notre objectif est d'avoir une estimation de la profondeur fiable quel que soit l'environnement (route ou rail) et les conditions météorologiques (pluie, brouillard, etc.)

3.2.3 Suivi des objets

Ces dernières années, le MOT (Multi-Object Tracking) basé sur l'apprentissage profond a atteint des performances de pointe en termes de qualité de suivi [60]. Par exemple, un détecteur d'objets tel que Faster-RCNN [50] associé à un filtre de Kalman linéaire permet un bon compromis entre temps de traitement et qualité de suivi, comme le montre SORT (Simple Online and Realtime Tracking) [61].

Afin d'évaluer les performances du suivi, nous devons définir quelles sont les erreurs possibles. Un premier type d'erreur est un échec, c'est-à-dire un objet qui existe dans une séquence d'images mais qui n'est pas détecté dans une ou plusieurs images. Un deuxième type d'erreur est un faux positif, où un objet détecté est associé par le tracker à un objet et à une trajectoire de vérité terrain, mais ne correspond pas à un objet existant. Un troisième type d'erreur est la non-concordance, lorsque les objets détectés correspondent à des objets existants mais ne sont pas associés par le tracker à des objets corrects. La somme de ces trois types d'erreur, calculée sur le nombre total d'objets présents dans toutes les images, définit la précision du suivi multi-objets (MOTA) [62], qui est un critère couramment utilisé pour le suivi.

3.3 Evaluation des algorithmes

3.3.1 Détection d'objets

Métriques d'évaluations

Afin d'évaluer les différentes approches de détection d'objets, nous utiliserons comme critères la précision moyenne et le temps de calcul. La précision moyenne (mAP), est un critère qui quantifie la qualité de la détection (proportion de détection correcte) en fonction du Rappel (proportion d'objets détectés). Plus précisément, nous définissons la précision et le rappel par les équations (3.3) et (3.4):

$$Precision = \frac{VP}{VP + FP} \tag{3.3}$$

$$Rappel = \frac{VP}{VP + FN} \tag{3.4}$$

Avec VP le nombre de Vrai Positif, FN celui de Faux Négatif et FP celui de Faux positif.

Un objet est considéré comme bien détecté si l'intersection sur l'union (ou IOU) entre la boîte détectée et la boîte de vérité terrain est supérieure à un seuil choisi (voir l'équation (3.5)).

$$Seuil\ IOU < \frac{Surface\ de\ recouvrement}{Surface\ d'union} \tag{3.5}$$

Pour calculer le mAP nous devons tout d'abord définir 11 valeurs de Rappel equidistantes, $Rappel_i = \begin{bmatrix} 0.0 & 0.1 & etc. & 0.9 & 1.0 \end{bmatrix}$. Le mAP se calcule ensuite avec l'équation (3.6):

$$mAP = \frac{1}{11} \sum_{Rappel_i} Precision(Rappel_i) \tag{3.6}$$

La précision au Rappel i est considérée comme étant la précision maximale mesurée à un Rappel

dépassant Rappel i.

Évaluation

Afin de comparer les différentes méthodes, nous avons comparé leurs performances en termes de mAP sur le jeu de données d'images COCO et mesuré leurs temps de calcul sur notre PC de test équipé d'une NVIDIA GTX 1050. Les résultats quantitatifs de cette évaluation se trouvent dans le tableau 3.1. Une comparaison visuelle des performances entre YOLOv3 et SSD est présentée dans la Figure 3.7.

Figure 3.7: Comparaison des performances des détecteurs Yolov3 (**gauche**) et SSD (**droite**).

Approche	mAP	Temps de calculs (ms)
SSD321	45.4	61
SSD513	50.4	125
R-FCN	51.9	85
FPN FRCN	**59.1**	172
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Tableau 3.1: Évaluation des performances des différentes méthodes de détection d'objets de l'état de l'art sur la base de données d'image COCO.

Choix de la méthode

Les résultats présentés dans le tableau 3.1 nous permettent de sélectionner une méthode optimale pour notre cas d'utilisation. Nous pouvons voir que la méthode la plus rapide est Yolov3 et que la plus lente est une méthode à deux étapes: FPN FRCNN. La deuxième observation que nous pouvons faire est que la méthode la plus précise est, comme prévu, la méthode à deux étapes FPN FRCNN. Cependant, nous pouvons voir que YOLOv3 a une précision très proche de celle-ci (-1,2 mAP) et que Yolov3 surpasse de loin SSD. Nous avons donc choisi d'utiliser YOLOv3 pour la partie détection d'objets de notre algorithme.

Un avantage supplémentaire de Yolov3 est qu'il est basé sur la bibliothèque Pytorch, qui est plus facile à apprendre et offre des performances similaires ou même meilleures que d'autres bibliothèques. Cela facilitera grandement les modifications que nous effectuerons pour concevoir notre méthode.

3.3.2 Estimation de la profondeur

Métriques d'évaluations

Afin d'évaluer les méthodes les plus prometteuses, nous avons utilisé le temps de calcul, les résultats qualitatifs et l'erreur quadratique moyenne (RMSE) présentée dans l'équation (3.7):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}\sum_{j}(g_{i,j} - p_{i,j})^2} \tag{3.7}$$

Avec i, j les coordonnées sur l'image, $g_{i,j}$ la vérité de base, $p_{i,j}$ la prédiction de profondeur et N le nombre total de points.

Cependant, l'évaluation des algorithmes est problématique car chaque méthode n'est pas toujours du même type (carte de disparité ou carte de profondeur) ou à la même échelle. Les paramètres de la caméra utilisés lors de l'entraînement, tels que la distance focale ou la distance entre les caméras stéréo, varient également d'une méthode à l'autre. Nous ne pouvons donc pas utiliser la formule (3.2) pour transformer les cartes de disparité en cartes de profondeur à l'échelle.

Pour résoudre ce problème, nous avons testé chaque méthode sur une séquence de l'ensemble de données KITTI, car elle possède une vérité de base de la distance à partir des données LiDAR. Nous pouvons ainsi transformer les cartes de disparité en une carte de profondeur en utilisant l'échelle médiane entre l'inverse de la disparité prédite et la vérité terrain. De la même manière, nous pouvons mettre à l'échelle les cartes de profondeur de certaines méthodes en utilisant l'échelle médiane entre la carte de profondeur et la vérité terrain. L'opération de transformation d'une carte de disparité en une carte de profondeur non échelonnée est détaillée dans l'équation (3.8) et l'opération d'échelonnement d'une carte de profondeur est donnée dans l'équation (3.9):

$$\widetilde{d_{i,j}} = \frac{1}{disp_{i,j}} \tag{3.8}$$

$$d_{i,j} = \widetilde{d_{i,j}} \times \frac{med(d_{gt})}{med(\widetilde{d})} \tag{3.9}$$

avec i, j les coordonnées du point sur la carte, $disp$ la disparité, $\widetilde{d}$ la profondeur non mise à l'échelle, et d la profondeur mise à l'échelle.

La précision de la méthode est ensuite estimée en calculant la différence quadratique moyenne entre la distance prédite par l'algorithme et la vérité terrain. Un exemple d'estimation de la distance des objets est présenté dans la Figure 3.10.

Évaluation

Lors de l'évaluation, nous avons choisi de séparer les méthodes évaluées en fonction du type d'images d'entrée requises (monoculaire ou stéréoscopique) afin de pouvoir identifier la méthode la plus adaptée en fonction du type de caméra choisi.

Approches monoculaires. Parmi les méthodes testées dédiées aux caméras monoculaires, on retrouve des approches telles que SfmLearner [53], MonoResMatch [56], Monodepth [63] et Monodepth2 [55]. A l'exception de [53] qui est entraîné sur des séquences d'images monoculaires, tous ces algorithmes nécessitent une paire d'images provenant d'une caméra stéréo calibrée pour effectuer l'entraînement. Cependant, les inférences de toutes ces méthodes sont réalisées sur des images provenant d'une caméra monoculaire, ce qui constitue un avantage significatif car cela réduit le coût de l'équipement. Des exemples

de cartes de disparité retournées sont présentés dans la Figure 2. Les résultats de notre évaluation sont présentés dans le Tableau 3.2.

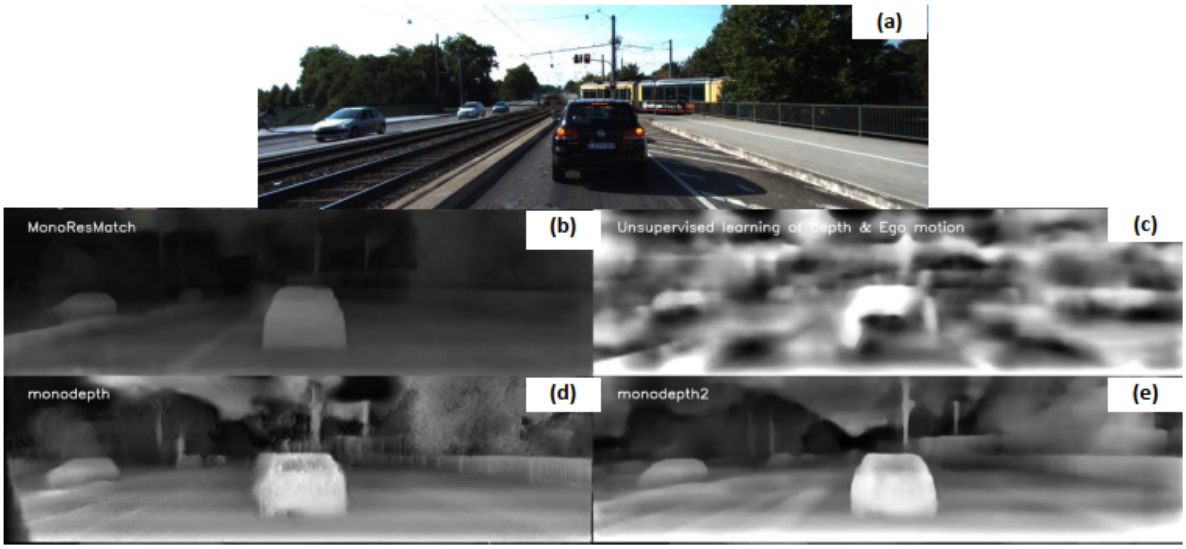

FIGURE 3.8: Résultats des cartes de disparité obtenues par des approches monoculaires: (**a**) image originale (**b**) MonoResMatch ; (**c**) SfmLearner ; (**d**) Monodepth et (**e**) Monodepth2.

Approches	RMSE	Temps de calcul (ms)
sfmLearner	16.530	50
Monodepth	6.225	200
MonoResMatch	5.831	1000
Monodepth2	5.709	50

Tableau 3.2: Résultats expérimentaux sur le dataset routier KITTI. Les cartes de profondeurs provenant d'un capteur est utilisé comme vérité de terrain.

D'après les résultats du tableau 3.2, Monodepth2 est la plus précise et la plus rapide des méthodes évaluées. C'est donc la méthode la plus adaptée à nos besoins si nous utilisons une caméra.

Approches stéréoscopiques. L'estimation de la distance par caméra est souvent associée à l'utilisation de caméras stéréoscopiques calibrées. Ces méthodes classiques sont basées sur l'association des pixels entre les deux caméras pour obtenir une carte de disparité. Cette dernière est utilisée pour produire la carte de profondeur. Cependant, l'association ne fonctionne que si la scène filmée est suffisamment texturée, c'est-à-dire qu'elle possède suffisamment de caractéristiques telles que des coins, des bords, etc. Ceci peut être corrigé par des méthodes telles que l'utilisation d'une "carte de texture". Cela peut être corrigé par des méthodes de post-traitement telles que le filtre WLS [64], mais la qualité des résultats est très variable. C'est pourquoi les algorithmes de Deep Learning ont un réel intérêt à obtenir la carte de disparité. Nous testons ici MadNet [15], il a la particularité d'être adaptatif, c'est-à-dire qu'il peut effectuer l'entraînement en même temps que l'inférence et ainsi pallier un défaut majeur des algorithmes d'estimation de distance de Deep Learning: des performances dégradées sur des environnements inconnus. Nous avons le choix entre 3 modes pendant l'inférence: Le mode None pour ne pas faire l'adaptation, le mode MAD pour réentraîner sur une partie seulement du réseau ou le mode FULL pour réentraîner tout le réseau. Le mode MAD est un bon compromis entre le temps de calcul et la performance de l'estimation de la distance. La Figure 3.9 montre des exemples de cartes de disparité obtenues par ces méthodes.

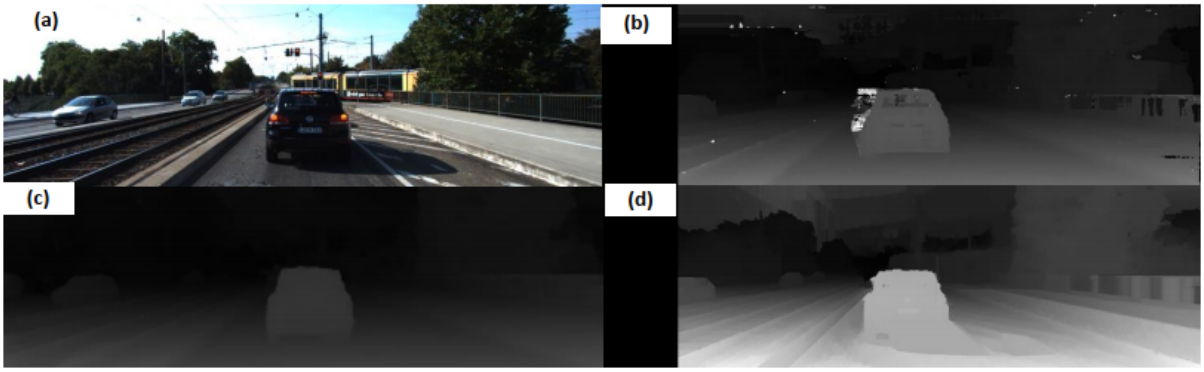

FIGURE 3.9: Résultats des cartes de disparité obtenues par des approches stéréoscopiques: (**a**) image originale ; (**b**) approche stéréo de base ; (**c**) MADNet et (**d**) filtre stéréo WLS.

Approach	RMSE	FPS
Stereo-baseline	9.002	15
Stereo-WLS Filter	8.690	7
MADNet	4.648	7

Tableau 3.3: Résultats expérimentaux des approches stéréoscopiques comparés à notre proposition basée sur MADNet.

Grâce aux données du Tableau 3.3 et de la Figure 3.9 nous pouvons identifier MADNet comme la méthode la plus adaptée et ce malgré un temps de calcul important. Ce temps de calcul peut être expliqué par le fait que MADNet est une méthode adaptative, ce qui signifie qu'elle réintègre une partie de son réseau pendant l'inférence afin de s'adapter à l'environnement pendant l'inférence. Cette évaluation nous permet également de mettre en évidence les performances supérieures des méthodes basées sur l'apprentissage profond par rapport aux approches " classiques " pour l'estimation de la profondeur à partir d'images stéréoscopiques.

Choix de la méthode

Grâce à notre évaluation, nous pouvons déterminer la meilleure méthode pour chacune des catégories: Monodepth2 (monoculaire) et MADNet (stéréoscopique). Les avantages de chaque méthode peuvent être résumés comme suit:

(1) Monodepth2

- Rapide (50 ms/image)
- Capable de fonctionner sur n'importe quel appareil photo

(2) MADNet

- Plus précis (-1,2 RMSE par rapport à Monodepth2)
- Capable d'être adapté

Nous avons choisi d'utiliser une caméra stéréoscopique avec MADNet car l'adaptation nous permet de compenser les problèmes liés au changement de domaine. Celle-ci permet de régler le problème de perte en précision lorsqu'une méthode est utilisée sur des images différentes de celles de la base de données utilisée pour l'entraînement et le test. Comme notre objectif est d'utiliser notre approche dans des conditions réelles, ce problème doit être abordé et l'adaptation offerte par MADNet semble être une solution prometteuse.

3.4 Localisation des objets

En utilisant les informations fournies par la détection des objets (boîtes de délimitation des objets) et l'estimation de la profondeur (carte de profondeur de la scène), il est maintenant possible de localiser chaque objet sur un plan 3D par rapport au système de coordonnées de la caméra. Notre problème se résume donc à projeter les coordonnées des pixels d'un objet dans le système de coordonnées de la caméra.

Avec (o_x, o_y) les coordonnées du centre optique de la caméra, (s_x, s_y) la taille d'un pixel et f la distance focale de la caméra. Les coordonnées Z peuvent être déterminées en utilisant les informations de la carte de profondeur avec le théorème de Pythagore 3D.

On se retrouve donc avec un système de 3 équations à 3 inconnues, qui permet de trouver les coordonnées d'un objet dans la caméra. Afin d'effectuer la localisation, nous combinons la détection d'objet et l'estimation de la profondeur dans une seule image pour vérifier si l'application en temps réel est réalisable. Le programme effectue la détection d'objet et l'estimation de la distance en parallèle, ce qui réduit le temps de calcul. Les résultats sont présentés dans la Figure 3.10.

Figure 3.10: Résultats de la détection et de la localisation d'objets sur un échantillon du jeu de données KITTI.

3.5 Suivi d'objets

3.5.1 Suivi en 2D

Plutôt que d'utiliser une approche classique de suivi visuel (Meanshift, Camshift, etc.), nous avons plutôt choisi de tirer profit des informations que nous avions déjà acquises par la détection et la localisation d'objets. Nous nous sommes donc basés sur le Simple Online Real-Time Tracking (SORT) [61] qui correspond parfaitement à nos besoins. Cet algorithme offre de bonnes performances, un temps de calcul extrêmement faible mais aussi un concept simple à modifier pour s'adapter à nos besoins. Cet algorithme utilise des boîtes englobantes fournies par un algorithme de détection pour initialiser les cibles qui sont ensuite suivies à l'aide d'un filtre de Kalman [16].

Le filtre de Kalman possède un vecteur d'état X et un vecteur de mesure z, définis par:

$$X = (x \;\; y \;\; r \;\; a \;\; \dot{x} \;\; \dot{y} \;\; \dot{a})^t, \;\; z = (x \;\; y \;\; r \;\; a)^t \tag{3.10}$$

où (x, y) sont les coordonnées de la boîte englobante, r son rapport largeur/hauteur et a son aire. Les quantités marquées d'un point sont les dérivées par rapport au temps. Le modèle de prédiction est basé

sur une vitesse constante.

L'association entre les cibles et les détections se fait en calculant le IOU (intersection over union) entre une détection et la position prédite par le filtre de Kalman. Si la détection obtient un IOU supérieur à un seuil prédéfini et a le IOU le plus élevé parmi les autres détections, alors elle est associée à la cible et le vecteur d'état du filtre de Kalman correspondant à la cible est mis à jour. La performance de cet algorithme dépend directement de la qualité de la détection de l'objet et le temps de calcul est inférieur à 1ms/frame.

3.5.2 Suivi en 3D

Compte tenu de notre objectif de suivre des objets en 3D, de nombreuses modifications ont été apportées à SORT. Ainsi, le Kalman a été remplacé par un filtre de Kalman étendu. Les principaux paramètres se trouvent ci-dessous:

$$X_s = (X \;\; Y \;\; Z \; a \;\; r \;\; \dot{X} \;\; \dot{Y} \;\; \dot{Z} \;\; \dot{a})^t, \;\; z = (x \;\; y \;\; d \;\; r \;\; a)^t \tag{3.11}$$

avec (X, Y, Z) correspondant aux coordonnées 3D et d est la profondeur estimée par la détection. Afin d'effectuer des prédictions, nous utilisons le modèle à vitesse constante. Le vecteur de mesure z , composé des coordonnées mesurées avec la détection d'objet et l'estimation de la distance, nous permet de mettre à jour le vecteur d'état du filtre. Nous avons également testé le modèle à accélération constante, mais les prédictions ont tendance à diverger, rendant le suivi impossible. Le modèle d'accélération constante pourrait donner de meilleurs résultats que le modèle de vitesse constante si les objets suivis changent brusquement à intervalles réguliers, ce qui n'est pas le cas ici car le filtre de Kalman étendu a un taux d'échantillonnage suffisant (ici il est égal à la fréquence d'images).

3.5.3 Ajustement des paramètres du filtre de Kalman étendu

La matrice de bruit de contrôle ajuste le comportement du filtre. Ainsi, le niveau de confiance entre les mesures et le modèle de prédiction peut être ajusté. Afin de filtrer une grande quantité de bruit de mesure, il est préférable de réduire la confiance des mesures, mais le filtre aura plus de difficultés lorsque la trajectoire change. Il s'agit donc de trouver un équilibre entre le filtrage du bruit et le contrôle des mesures. Nous avons utilisé la méthode des essais et erreurs pour trouver ces paramètres.

3.5.4 Résultats de l'algorithme de suivi d'objets

Nous avons testé l'algorithme de suivi dans plusieurs environnements. Dans un premier temps, les tests ont été effectués en intérieur, l'avantage étant qu'il est plus facile de déterminer si les valeurs prédites divergent ou non. Nous avons utilisé la caméra Intel RealSense comme caméra stéréoscopique dans les tests présentés sur la Figure 3.11.

FIGURE 3.11: Résultat de l'approche de suivi sur une scène d'intérieur. L'image de gauche est acquise à l'instant $t = 1$ et celle de droite à l'instant $t + 3$. A $t = 1$, nous attribuons un ID à chaque objet détecté. Ensuite, à $t + 3$, le tracklet est validé et nous affichons la vitesse estimée.

Nous avons ensuite effectué des tests dans le domaine routier en utilisant le jeu de données KITTI (voir Figure 3.12). Ici, nous pouvons voir que le vélo à droite n'est plus détecté à partir de l'image ($t + 1$). La prédiction du filtre de Kalman est alors utilisée pour estimer la position de cet objet. S'il n'est pas associé à une détection avant 3 images, l'algorithme supprime cette cible. Notre méthode permet également de suivre une cible même si elle n'est pas associée à un blob de détection jusqu'à 3 images avant sa suppression. Nous pouvons voir dans la Figure 3.12 et la Figure 3.13 les résultats quantitatifs et qualitatifs de notre méthode sur deux séquences différentes du jeu de données KITTI.

FIGURE 3.12: Résultats du suivi d'objets dans un environnement routier sur une séquence du jeu de données KITTI.

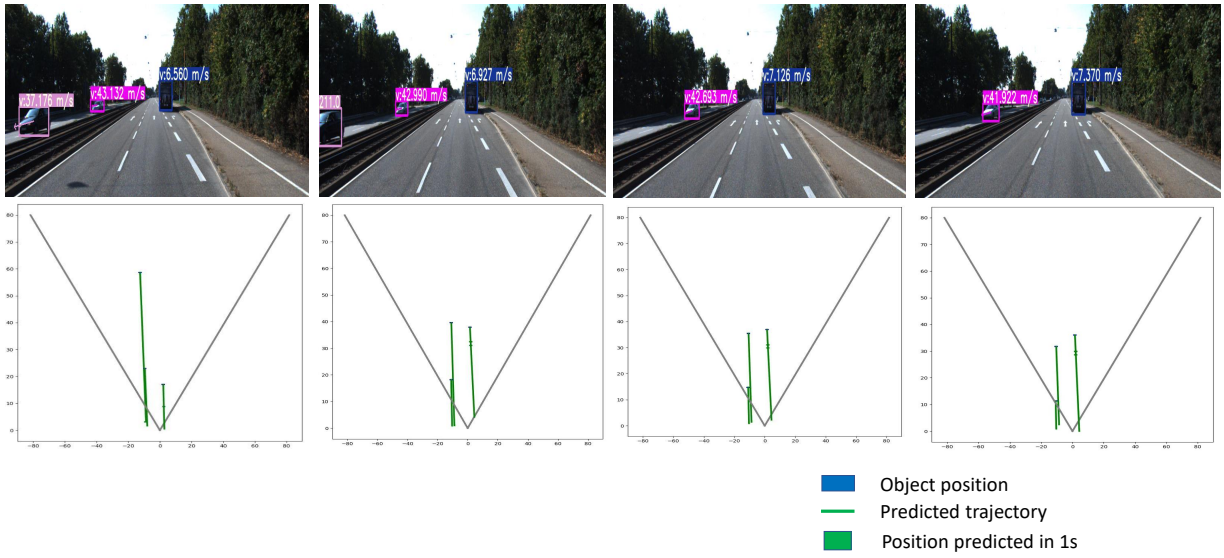

FIGURE 3.13: Résultats du suivi d'objets sur une séquence de la base de données KITTI. En haut, 4 images RVB provenant d'une séquence de route acquise à différents moments avec les boîtes de suivi correspondantes des objets en mouvement et leurs valeurs de vitesse. En bas, les cartes montrées permettent un aperçu simple mais complet des mouvements et des distances des objets suivis.

3.6 Conclusion

Dans ce chapitre, nous avons présenté notre première approche pour la détection, la localisation et la prédiction de trajectoire d'objets afin de prévenir les risques de collision pour les domaines ferroviaire et routier, basée sur Yolov3 pour la détection, MADNet pour l'estimation de la profondeur et enfin un filtre de Kalman pour leurs suivi. Cette première approche nous a permis de confirmer la faisabilité d'une telle solution mais surtout, elle nous a permis d'identifier les principaux défis à surmonter pour concevoir une telle méthode. Nous avons pu constater les limites des approches de l'état de l'art, qui ne permettent pas de répondre directement à notre problématique. Cette expérience nous permettra d'identifier les contributions que nous pouvons apporter dans cette thèse.

En raison de l'absence de bases de données ferroviaires avec suffisamment de vérité terrain pour entraîner et évaluer les différentes approches, nous n'avons pas pu tester notre approche sur le domaine ferroviaire. Notre approche basée sur 3 modules indépendants n'est pas une solution optimale à notre problème, en effet le temps de calcul reste important car nous cumulons le temps de calcul de chaque module. Nous avons également constaté que la précision globale de notre approche dépend principalement de la précision de la détection et de la localisation des objets.

Mais si la détection d'objets est un domaine très développé en apprentissage profond, l'estimation de la profondeur l'est moins car elle est plus complexe.

Nous pouvons donc déduire un plan à partir de ces observations:

(1) Approfondir l'évaluation des méthodes d'estimation de la profondeur

(2) Concevoir notre propre base de données avec des images de l'environnement ferroviaire

(3) Fusionner les différents modules (détection, localisation, suivi d'objet) afin de réduire le nombre de modules indépendants et de diminuer le temps de calcul.

Chapitre 4

Protocole d'Evaluation

4.1 Introduction

Dans ce chapitre nous aborderons les travaux que nous avons menés afin d'approfondir l'étude des méthodes d'estimations de profondeurs, indispensable pour la localisation des objets. La localisation des objets étant une partie essentielle pour l'évitement d'obstacles et celle-ci étant la plus complexe, il est indispensable pour nous d'évaluer rigoureusement ces méthodes. Dans le chapitre précédent, nous avons déjà évalué exhaustivement les méthodes monoculaires pour l'estimation de profondeur. Cependant, nous avions seulement évalué une seule méthode stéréoscopique pour l'estimation de profondeur basée sur l'apprentissage profond.

Une autre lacune que nous pouvons constater est que le protocole d'évaluation, identique à celui de la littérature, que nous avions utilisé n'est pas réellement adapté pour la localisation d'objets. En effet, l'évaluation des cartes de profondeurs fournies par les différentes méthodes sont évaluées sur la totalité de l'image et ne fournissent pas d'information sur la qualité de la prédiction de profondeur des pixels appartenant aux objets dont on s'intéresse (voitures, piétons, etc.). De plus, le protocole d'évaluation actuel ne donne aucune information sur la dégradation de la précision de l'estimation de profondeur à mesure que la distance augmente.

Dans ce chapitre, nous allons donc répondre à ces problématiques en suivant le plan ci-dessous:

(1) Approfondir l'étude des méthodes stéréoscopiques basées sur l'apprentissage profond pour l'estimation de profondeur

(2) Créer un nouveau protocole pour l'évaluation des méthodes d'estimations de profondeur

4.2 Etat de l'art

4.2.1 Méthodes d'estimation de profondeur

Dans cette partie nous allons présenter les nouvelles méthodes pour l'estimation de profondeur que nous avons identifié. Ces méthodes seront ensuite évaluées afin de déterminer la différence de précision entre les méthodes basées sur des images monoculaire et celles stéréoscopiques.

Estimation de profondeur monoculaire

Nous avons choisi, pour cette étude comparative, d'utiliser Monodepth2 qui avait été identifié dans le chapitre précédent comme étant la méthode monoculaire la plus précise. Cependant, nous avons aussi identifié une nouvelle méthode, nommée BTS (Big-To-Small) [65] qui propose d'obtenir la profondeur à pleine résolution en fusionnant les sorties des différentes couches intermédiaires de la partie décodeur du réseau. L'architecture CNN proposée par cette approche permet de contourner le problème causé par le goulot d'étranglement de l'architecture codeur-décodeur pour obtenir une carte de profondeur à pleine résolution. Cette architecture est actuellement parmi les méthodes les plus performantes sur le benchmark d'estimation de la profondeur de la base de données KITTI.

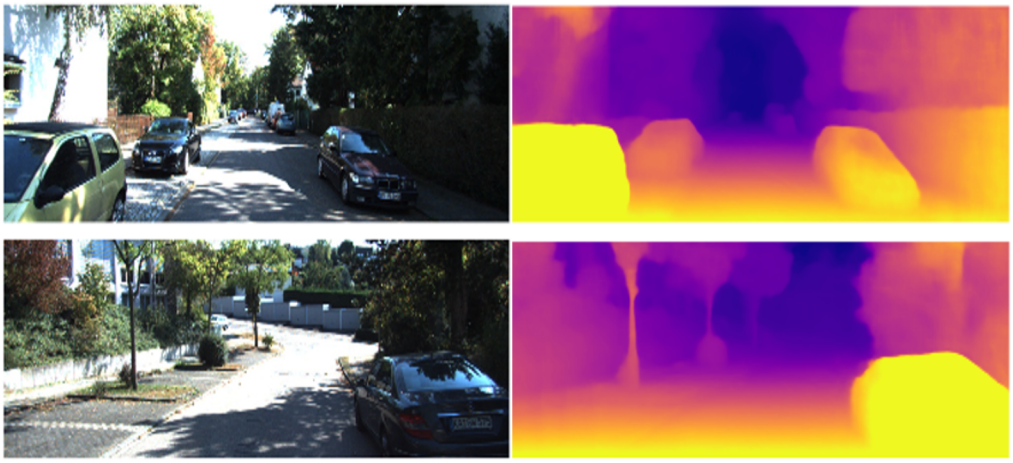

FIGURE 4.1: Images de profondeur provenant du jeu de données KITTI [38]. Les images RVB se trouvent à gauche et les images de profondeur à droite.

Estimation de profondeur stéréoscopique

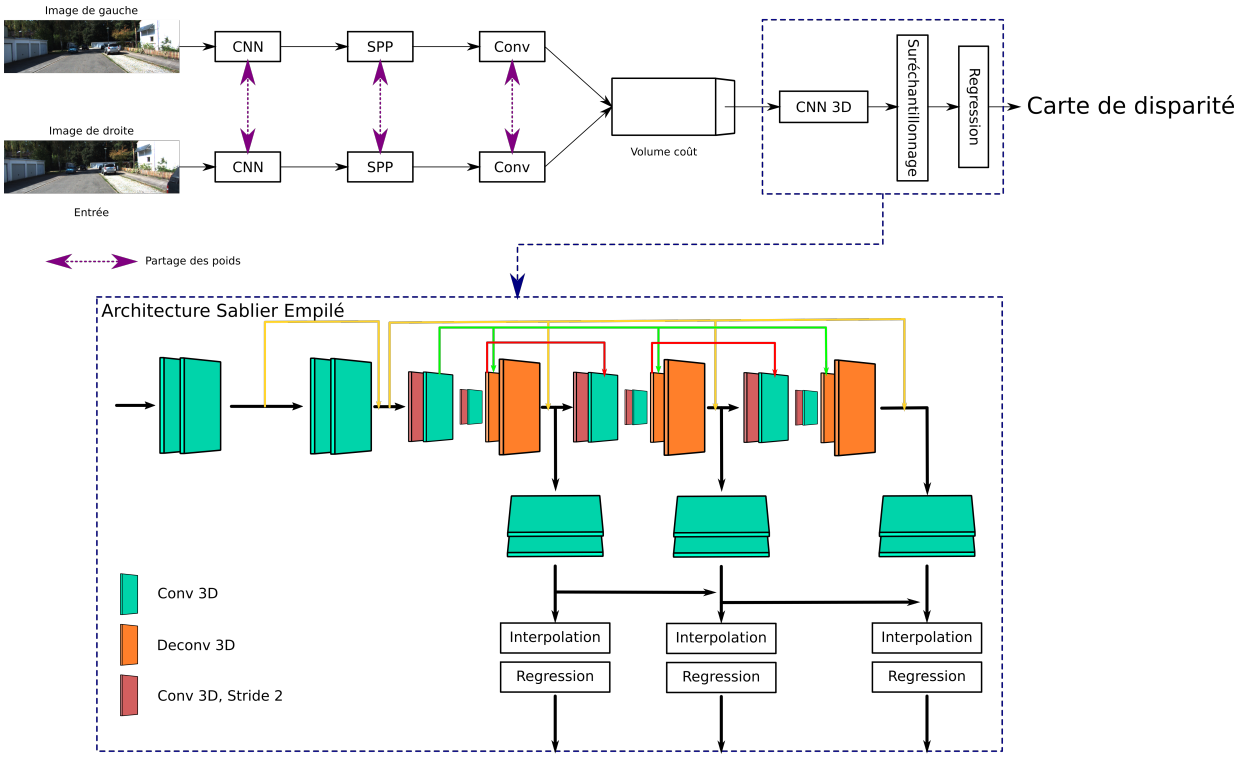

FIGURE 4.2: Architecture du réseau PSMNet.

PSMNet.[66] Des travaux récents ont montré que l'estimation de la profondeur à partir d'un couple d'images stéréo peut être formulée comme une tâche d'apprentissage supervisé à résoudre avec des réseaux de neurones convolutifs. Cependant, les architectures actuelles s'appuient sur des réseaux siamois basés sur des patchs, manquant de moyens pour exploiter l'information contextuelle pour trouver la correspondance dans des régions mal disposées. C'est dans ce contexte qu'a été proposé PSMNet, un réseau de correspondance stéréo pyramidale composé de deux modules principaux: le pooling pyramidal spatial (SPP) et un CNN 3D. L'architecture de PSMNet est détaillé dans la Figure 4.2. L'approche proposée crée ce qui est appelé le volume coût (*cost volume*) en envoyant les images stéréo d'entrée gauche et droite à deux pipelines de partage de poids composés d'un CNN pour le calcul des cartes de caractéristiques,

d'un module SPP pour la récolte des caractéristiques en concaténant les représentations de sous-régions de tailles différentes, et d'une couche de convolution pour la fusion des caractéristiques. Les sorties de ces deux pipelines sont ensuite combinés afin de former le volume de coût. Ce volume coût est ensuite passé par un CNN 3D qui se présente sous la forme d'une architecture en "sablier empilé" composé de trois réseaux principaux, chacun d'eux générant une carte de disparité. Lors de l'entrainement, ces trois cartes de disparités ont leurs propres fonctions perte afin de mieux apprendre les informations contextuelles. Mais lors de l'inférence, seul la dernière carte de disparité est renvoyée comme sortie.

GWCNet.[67] Afin d'améliorer l'approche proposée par PSMNet, une nouvelle méthode d'estimation de profondeur stéréoscopique a été proposée, cette approche incorpore des améliorations pour l'architecture du "sablier empilé" présent dans le CNN 3D ainsi qu'un nouveau type de réseau: le réseau stéréoscopique à corrélation par groupe (Group-Wise Correlation stereo Network ou GWCNet). L'architecture détaillée est présenté dans la Figure 4.3. Cette approche peut-être décomposée en quatre modules: extraction des cartes de caractéristiques des images gauche et droite, construction de volumes de coûts, agrégation 3D et prédiction de la disparité. La contribution principale de cette approche vis-à-vis de PSMNet repose sur l'utilisation de la corrélation par groupe pour construire le volume de coût. Dans PSMNet le volume de coût était créé simplement en concaténant les cartes de caractéristique extraites des images gauches et droites, et les coûts de correspondance pour les caractéristiques concaténées devaient être appris à partir de zéro par le réseau d'agrégation 3D, ce qui entrainait généralement un plus grand nombre de paramètres et un temps de calcul augmenté. En revanche, la corrélation complète offre un moyen efficace de mesurer les similarités entre les caractéristiques via le produit scalaire, mais cela se fait au prix d'une perte importante d'informations car il ne produit qu'une carte de corrélation à un seul canal pour chaque niveau de disparité. La corrélation par groupe a donc été proposée afin de surmonter ces deux inconvénients et fournir de bonnes caractéristiques pour les mesures de similarité.

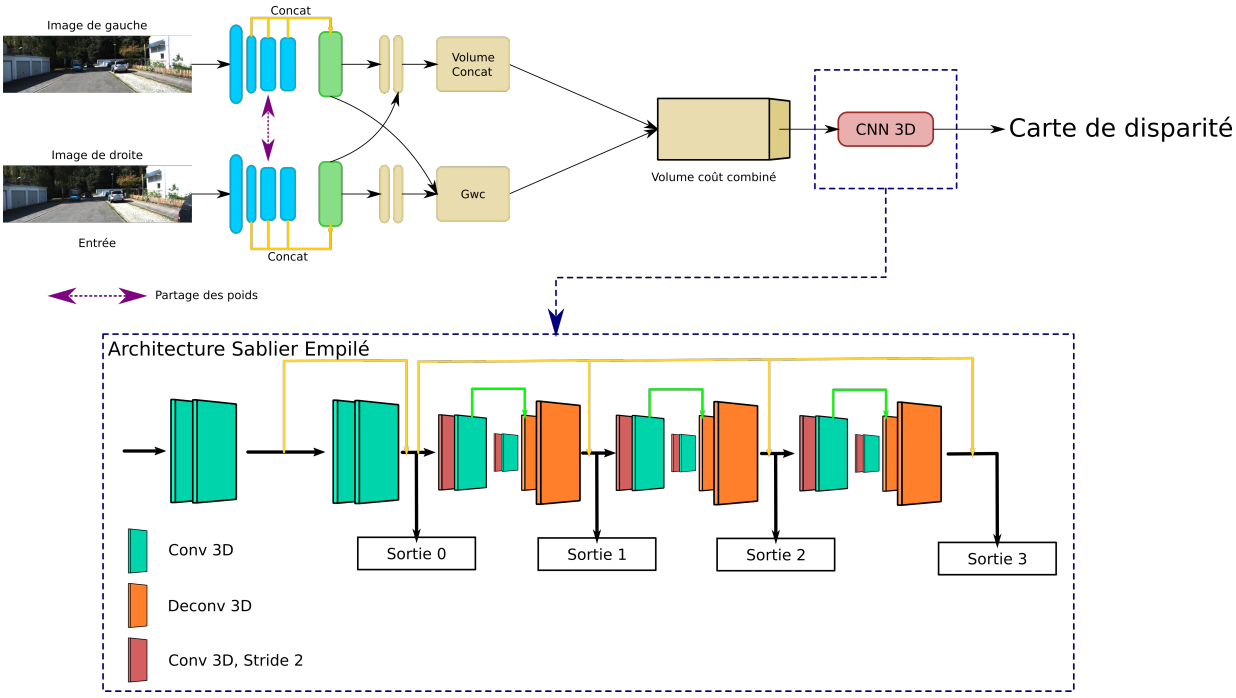

Figure 4.3: Architecture du réseau GWCNet.

4.2.2 Évaluation des méthodes d'estimation de la profondeur

Bien que certains travaux aient été réalisés en termes d'étude comparative des méthodes d'estimation de la profondeur pour une caméra stéréo ou monoculaire, peu de travaux présentent une étude comparative des méthodes monoculaires et stéréo. [68] présente une étude complète de l'estimation de la profondeur basée sur la stéréo ainsi qu'une évaluation approfondie des méthodes. [69] propose un nouvel ensemble de protocoles d'évaluation consacrés à l'estimation de la profondeur sur une seule image afin de mieux évaluer les performances des méthodes proposées. Il fournit principalement une évaluation de plusieurs méthodes monoculaires dans des environnements intérieurs. Le document [70] propose également une comparaison et une évaluation des architectures de codeurs multiples pour l'estimation de la profondeur.

Afin d'évaluer les performances des approches d'estimation de profondeur, différentes erreurs statistiques sont utilisés: L'erreur relative (RelErr), l'erreur relative au carré (SRE), l'erreur quadratique moyenne (RMSE) et l'erreur quadratique moyenne logarithmique (logRMSE). Ces métriques donnent une évaluation globale de la performance d'une méthode sur l'ensemble de l'image testée. Nous présentons expressément chaque métrique dans ce qui suit. Nous notons par p la prédiction de profondeur, gt comme sa vérité de terrain correspondante et N le nombre total de points. Enfin u et v correspondent aux coordonnées en pixels du point dans le repère de l'image.

Erreur relative (RE) est donnée par l'équation (4.1):

$$RE = \frac{1}{N}\sum_{u}\sum_{v}\frac{\left|gt_{u,v} - p_{u,v}\right|}{gt_{u,v}} \tag{4.1}$$

Erreur relative au carré (SRE) est donnée par l'équation (4.2):

$$SRE = \frac{1}{N}\sum_{u}\sum_{v}\frac{\left|gt_{u,v} - p_{u,v}\right|^2}{gt_{u,v}} \tag{4.2}$$

Erreur quadratique moyenne (RMSE) est donnée par l'équation (4.3):

$$RMSE = \sqrt{\frac{1}{N}\sum_{u}\sum_{v}(gt_{u,v} - p_{u,v})^2} \tag{4.3}$$

Erreur quadratique moyenne logarithmique (logRMSE) est donnée par l'équation (4.4):

$$logRMSE = \sqrt{\frac{1}{N}\sum_{u}\sum_{v}(log(gt_{u,v}) - log(p_{u,v}))^2} \tag{4.4}$$

Pourcentage de pixels non conformes: Le pourcentage de pixels non conformes (appelé BMP pour *Bad Matching Pixels*) est donné par l'équation (4.5), où C est un seuil utilisé pour définir une tolérance d'erreur:

$$[a]_{k=[1..3]} = \frac{1}{N}\sum_{u}\sum_{v} max(\frac{g_{u,v}}{p_{u,v}}, \frac{p_{u,v}}{g_{u,v}}) < C^k \tag{4.5}$$

4.3 Étude comparative des approches d'estimation de la profondeur

4.3.1 Résultats expérimentaux

Dans cette section sont exposés les résultats menés lors de notre évaluation en utilisant les métriques présenté dans la section précédente. Nous avons choisi d'explorer à la fois des méthodes stéréoscopiques (GWCNet, PSMNet) mais aussi monoculaires (BTS, Monodepth2) afin de comparer l'écart de précisions entre ces deux types de méthodes. Nous avons fait le choix de ces méthodes car elles représentent les méthodes de l'état de l'art les plus prometteuse en terme de précision dans chacune de leurs catégories (stéréoscopiques / monoculaires et supervisée / non supervisée).

Évaluation des méthodes basées sur une seule image

Nous avons utilisé les modèles pré-entraînés publiés par les auteurs des réseaux BTS et Monodepth2 pour l'évaluation sur le jeu de données KITTI. BTS a été entraîné avec une résolution de 704×352 sur la partie d'entraînement de la base de donnée définie par Eigen [71], cet échantillon de la base de donnée possède une vérité terrain dense ce qui permet d'améliorer les performances des méthodes lors de l'entraînement. Monodepth2 a été entrainé en utilisant le mode monoculaire non supervisé sur la partie d'entraînement de KITTI définie dans les travaux de Zhou [53] avec une résolution d'image de 1024×320. Les deux méthodes ont été évaluées sur la partie de test définie par Eigen. Les résultats sont présentés dans le tableau 4.1.

	RE	SRE	RMSE	logRMSE
Monodepth2	0.115	0.882	4.701	0.190
BTS	**0.060**	**0.249**	**2.798**	**0.096**

Tableau 4.1: Évaluation de la profondeur monoculaire sur KITTI. Les méthodes évaluées comprennent Monodepth2 (MD2) et BTS, deux méthodes d'estimation de la profondeur monoculaire à la pointe de la technologie. L'erreur relative au carré (SRE) et l'erreur quadratique moyenne (RMSE) sont exprimées en mètres.

Les résultats montrent que BTS est globalement plus performant que Monodepth2 pour toutes les mesures d'erreur. L'une des raisons pour lesquelles BTS est plus performant que Monodepth2 est probablement qu'il a été entraîné de façon supervisé avec des informations de vérité de terrain pour la profondeur alors que Monodepth2 a été entrainé de manière non supervisée. Monodepth2 a utilisé des séquences d'images pour l'entraînement non supervisé et a utilisé un modèle pour apprendre l'ego-motion de la caméra pour la supervision. Il en résulte un grand nombre d'approximations pendant l'apprentissage. De plus, le modèle BTS est beaucoup plus profond et plus lourd en termes de calculs que Monodepth2.

Évaluation des méthodes basées sur des images stéréoscopiques

Les résultats des méthodes stéréoscopiques sur le jeu de données KITTI montrent que GWcNet est significativement meilleur que PSMNet. Les résultats qualitatifs sont présentés dans la Figure 4.4 aux cotés de la vérité de terrain provenant du LiDAR, mais il est difficile de constater une différence en terme de précision entre les deux méthodes à l'œil nu. L'évaluation quantitative présenté dans le Tableau 4.2 quand à elle permet est un meilleur outil de comparaison afin de déterminer la méthode la plus précise.

Grâce aux résultats expérimentaux des deux approches stéréo comparées, nous pouvons expliquer la différence de performance entre le GwcNet et le PSMNet par le fait que les fonctionnalités d'agrégation des caractéristiques rendent le réseau plus robuste face aux scènes routières difficiles (comme les objets et

	RE	SRE	RMSE	logRMSE
GWCNET	**0.018**	**0.048**	**0.981**	**0.042**
PSMNET	0.032	0.061	1.139	0.056

Tableau 4.2: Évaluation de la profondeur stéréoscopique sur KITTI. Les méthodes évaluées comprennent GWCNet et PSMNet, deux méthodes d'estimation de la profondeur stéréoscopique à la pointe de la technologie. L'erreur relative au carré (SRE) et l'erreur quadratique moyenne (RMSE) sont exprimées en mètres.

les régions sans texture, les changements soudains d'éclairage, etc.) qui sont plus présents dans le GcwNet en raison de la corrélation de groupe fortement recommandée dans les réseaux stéréo.

Grâce aux résultats expérimentaux des deux approches stéréo comparées, nous pouvons expliquer la différence de performance entre le GwcNet et le PSMNet par le fait que la corrélation de groupe proposé par GWCNet permet de rendre le réseau plus robuste face aux scènes routières difficiles pour l'estimation de profondeur (par exemple la présence d'objets sans textures, changement soudains d'éclairages etc.)

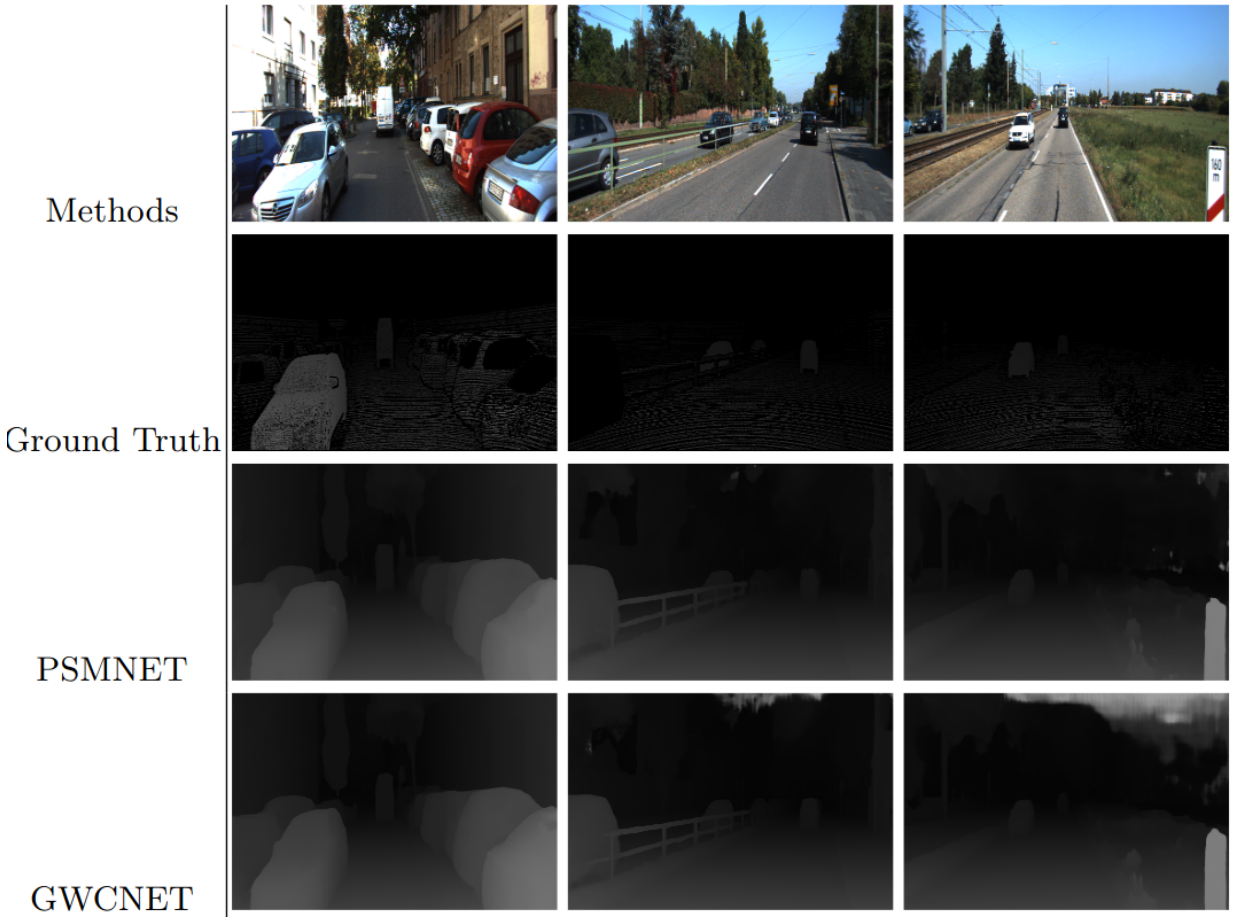

Figure 4.4: Résultats qualitatifs des méthodes d'estimation de la profondeur basées sur la stéréo (GWCNet et PSMNet) sur des échantillons du jeu de donnée KITTI.

4.3.2 Bilan

Nous avons donc présenté une étude comparative complète des méthodes d'estimation de la profondeur basées sur l'apprentissage profond pour les caméras monoculaires et stéréoscopiques. Cette étude comparative a été menée sur la base de donnée routière de KITTI car les conditions de trafic réelles que présente ce jeu de données permettent d'obtenir une bonne appréciation des performances d'une

méthode dans des conditions réalistes. Notre évaluation expérimentale a montré que BTS et GWCNet offrent les meilleures performances pour les caméras monoculaires et stéréoscopiques respectivement. L'analyse menée montre également que les méthodes stéréoscopiques ont une plus grande précision que les méthodes basées sur la vision monoculaire. Cette étude plus approfondie des méthodes d'estimation de profondeur nous permettent de donner une estimation des méthodes les plus intéressantes pour notre application, cependant, cette évaluation ne permet pas de déterminer quantitativement la performance de ces approches pour l'estimation de profondeur des objets spécifiquement. Il est aussi impossible, via cette évaluation, de mesurer la dégradation de la précision en fonction de la distance.

4.4 Nouveau protocole d'évaluation pour l'estimation de profondeur

Afin de pallier les problèmes soulevés dans la section précédente, nous avons conçu un nouveau protocole d'évaluation de l'estimation de profondeur avec pour but d'obtenir une évaluation cohérente avec notre tâche de localisation d'objets. Ce nouveau protocole présente deux nouveaux protocoles pour l'évaluation. Le premier permet d'évaluer la qualité de la prédiction de la profondeur en fonction de la distance afin de quantifier la dégradation de la précision lorsque la distance augmente. Le deuxième protocole vise à quantifier la précision de l'estimation de profondeur des points appartenant à un objet, cela permettra une évaluation de la précision des approches pour l'estimation de la distance des objets spécifiquement.

Chacune des méthodes qui seront évaluées renvoient soit des disparités de distance soit la profondeur elle-même. Bien qu'il soit possible de trouver la profondeur à partir de la disparité en utilisant la formule (3.8), chacune des méthodes a été entraîné à une résolution d'image particulière, ce qui par extension change aussi la focale utilisé dans la formule. Nous avons donc opté pour utiliser une mise à l'échelle en utilisant les information de la vérité de terrain. Cela nous permet d'obtenir les profondeurs prédites pour chacune des méthodes sans passer par l'équation (3.8).

4.4.1 Évaluation de la profondeur selon l'objet

Alors que l'évaluation actuelle des estimations de la profondeur donne une évaluation complète de la performance globale d'une méthode donnée, elle est faite sur l'image globale et n'évalue pas la prédiction de la distance des objets. Or la distance des objets est un aspect fondamental pour les applications de conduite autonome et de perception des scènes. C'est pourquoi nous avons conçu un nouveau protocole d'évaluation de la profondeur qui nous permet de calculer l'erreur de prédiction de la profondeur pour les objets pertinents que l'on rencontre régulièrement dans les environnements routiers (personne, voiture, camion, etc.).

Notre protocole d'évaluation est composé de 4 étapes:

(1) La carte de profondeur prédite est mise à l'échelle en utilisant une mise à l'échelle médiane. Soit p la carte de profondeur prédite et g la carte de profondeur de la vérité de terrain, la mise à l'échelle médiane est décrite comme suit: $p = p * \frac{med(gt)}{med(p)}$.

(2) Les masques d'objets sont générés en utilisant Mask-RCNN [35] (voir la Figure 4.5 pour un exemple de sortie du réseau)

(3) Les masques des objets générés sont ensuite utilisés pour segmenter les cartes de profondeur et les erreurs de profondeur sont calculées pour chaque masque dans l'image

(4) Enfin, la moyenne des erreurs est calculée pour chaque classe. Ce nouveau protocole d'évaluation

permettra de mieux comprendre la façon dont une méthode donnée estime la distance des objets présents dans l'environnement routier. Il est particulièrement utile pour les applications de conduite autonome. Les différentes étapes sont illustrées dans la Figure 4.6.

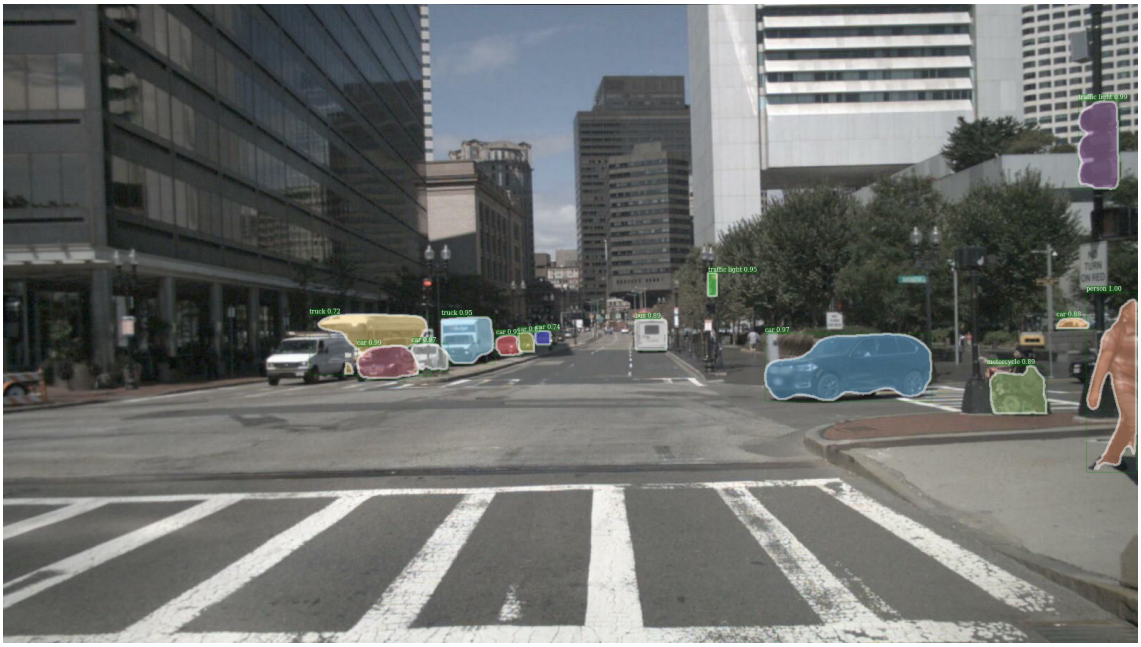

FIGURE 4.5: Masques des objets obtenus avec Mask-RCNN sur une image du jeu de données NuScenes.

4.4.2 Évaluation de la profondeur sur des plages de distance

Un autre inconvénient du protocole classique d'évaluation de la profondeur est qu'il ne permet pas d'évaluer les performances d'une méthode selon la distance. La performance d'une méthode sur des distances plus longues est un paramètre important qui doit être pris en compte pour la compréhension de la scène. Nous proposons ici de suivre le travail de [72] qui a décrit un protocole d'évaluation sur des plages de distance, mais alors qu'ils ont utilisé ce protocole pour des scènes intérieures, nous l'avons utilisé dans un environnement routier où les plages de distance sont plus importantes. Le protocole que nous utilisons est donc composé des étapes suivantes:

(1) La carte de profondeur prédite est mise à l'échelle en utilisant une échelle médiane

(2) Des plages de distance de 10m à 80m ([0-10m], [10-20m], etc., [70-80m]) sont créées

(3) Chaque pixel est affecté à une plage de distance en fonction de sa valeur dans la vérité terrain de la profondeur

(4) Pour chaque plage de distance, les erreurs de profondeur sont calculées.

Ce nouveau protocole permet d'évaluer la dégradation de l'estimation de la profondeur en fonction des distances.

4.5 Analyse des résultats

Dans cette section, nous présentons les résultats de notre évaluation des méthodes monoculaires d'estimation de la profondeur supervisées et auto-supervisées BTS et Monodepth2 (respectivement), sur les jeux de données KITTI et NuScenes.

	RE		SRE		RMSE		logRMSE		a_1		a_2		a_3	
Plage de distance	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS
$0-80m$	0.115	**0.060**	0.882	**0.249**	4.701	**2.798**	0.190	**0.096**	0.879	**0.955**	0.961	**0.993**	0.982	**0.998**
$0-10m$	0.102	**0.071**	0.503	**0.188**	1.489	**0.991**	0.141	**0.106**	0.929	**0.959**	0.979	**0.988**	0.990	**0.994**
$10-20m$	0.116	**0.088**	0.845	**0.395**	3.035	**2.198**	0.180	**0.149**	0.891	**0.924**	0.960	**0.971**	0.979	**0.985**
$20-30m$	0.168	**0.130**	1.866	**1.055**	6.208	**4.745**	0.261	**0.229**	0.773	**0.836**	0.916	**0.934**	0.957	**0.964**
$30-40m$	0.196	**0.160**	2.788	**1.945**	9.110	**7.476**	0.307	**0.279**	0.694	**0.764**	0.886	**0.906**	0.942	**0.947**
$40-50m$	0.209	**0.174**	3.504	**2.640**	11.682	**10.008**	0.318	**0.298**	0.641	**0.725**	0.865	**0.889**	**0.943**	0.941
$50-60m$	0.221	**0.190**	4.394	**3.739**	14.252	**12.852**	0.332	**0.326**	0.583	**0.675**	0.857	**0.868**	**0.927**	0.922
$60-70m$	0.212	**0.201**	4.657	**4.584**	15.855	**15.585**	**0.325**	0.334	0.609	**0.619**	0.854	**0.856**	**0.930**	0.923
$70-80m$	**0.181**	0.214	**4.340**	5.454	**15.800**	18.219	**0.284**	0.333	**0.652**	0.548	**0.873**	0.843	**0.945**	0.925

Tableau 4.3: Évaluation de la profondeur sur des plages de distance pour KITTI. Les algorithmes évalués sont des méthodes monoculaires d'estimation de la profondeur reconnues: Monodepth2 (MD2) et BTS. Les erreurs de profondeur globales sont indiquées ainsi que les erreurs de profondeur pour des distances comprises entre 10m et 80m. La RMSE est exprimée en mètres.

	RE		SRE		RMSE		logRMSE		a_1		a_2		a_3	
Classe de l'objet	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS
Personne	0.314	**0.166**	5.721	**1.786**	8.430	**5.892**	0.326	**0.253**	0.601	**0.772**	0.829	**0.894**	0.920	**0.947**
Deux roues	0.131	**0.116**	0.517	**0.467**	2.810	**2.669**	0.172	**0.163**	0.829	**0.839**	**0.964**	0.962	0.993	**0.994**
Voiture	0.206	**0.137**	3.132	**1.491**	7.924	**6.052**	0.271	**0.223**	0.773	**0.838**	0.883	**0.922**	0.938	**0.955**
Camion	0.215	**0.122**	2.769	**0.826**	6.978	**4.523**	0.259	**0.177**	0.694	**0.854**	0.903	**0.969**	0.964	**0.985**

Tableau 4.4: Évaluation de la distance selon l'objet sur KITTI. Les algorithmes évalués sont des méthodes monoculaires d'estimation de la profondeur reconnues: Monodepth2 (MD2) et BTS. Les erreurs de profondeur ont été calculées pour les classes d'objets ayant suffisamment d'instances dans l'ensemble de test. La RMSE est exprimée en mètres.

4.5.1 Base de données KITTI

Pour l'évaluation sur la base KITTI, nous avons utilisé les modèles pré-entraînés fournis par les auteurs des modèles BTS et Monodepth2. Le modèle BTS a été entraîné avec des images provenant de l'entraînement d'Eigen [71] à une résolution de 704×352 et avec une vérité terrain dense. Les poids de Monodepth2 ont été entraînés en utilisant la vérité terrain monoculaire sur l'ensemble de Zhou[53], à une résolution de 1024×320. L'évaluation a été réalisée sur l'ensemble de test d'Eigen. Les résultats de notre évaluation de BTS et Monodepth2 sont présentés dans les tableaux 4.3 et 4.4. La valeur de C pour l'erreur de seuil définie dans l'équation (4.5) a été fixée à 1,25.

4.5.2 Base de données NuScenes

Pour l'évaluation sur NuScenes, nous avons dû entraîner les deux méthodes sur l'ensemble d'entraînement de ce jeu de données. Pour BTS, nous avons réalisé un entraînement de 50 epochs avec une taille de batch de 20 et une résolution de 192×192, avec les données éparses de la supervision LiDAR. Étant donné que Monodepth2 est une méthode non supervisée, elle repose sur la fonction de perte de reprojection monoculaire. Si les images ont une mauvaise visibilité (due aux conditions météorologiques, de l'obscurité, etc.), l'entraînement peut ne pas converger. C'est pourquoi nous avons sélectionné les scènes de l'entraînement où la visibilité est suffisamment bonne pour que l'entraînement converge. Nous avons également utilisé toutes les images de chaque scène et pas seulement celles qui étaient synchronisées avec le LiDAR afin d'obtenir une fréquence d'images suffisamment élevée pour que l'entraînement monoculaire fonctionne. Nous avons entraîné Monodepth2 pendant 20 epochs avec une taille de batch de 12 et une résolution de 446×224. Les résultats de notre évaluation de BTS et de Monodepth2 sont présentés dans les tableaux 4.5 et 4.6. La valeur de C pour l'erreur de seuil définie dans l'équation (4.5) a été fixée à 1,25.

	RE		SRE		RMSE		logRMSE		a_1		a_2		a_3	
Plage de distance	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS
$0-80m$	0.176	**0.147**	2.521	**1.184**	7.746	**5.849**	0.271	**0.214**	0.787	**0.817**	0.911	**0.94**	0.955	**0.977**
$0-10m$	**0.115**	0.116	0.982	**0.430**	1.561	**1.347**	**0.139**	0.151	**0.919**	0.915	0.972	**0.974**	0.986	**0.987**
$10-20m$	0.187	**0.153**	2.690	**0.966**	4.854	**3.452**	0.243	**0.197**	0.794	**0.810**	0.916	**0.945**	0.958	**0.984**
$20-30m$	0.242	**0.162**	3.488	**1.386**	8.316	**5.427**	0.320	**0.217**	0.643	**0.768**	0.859	**0.933**	0.932	**0.978**
$30-40m$	0.256	**0.183**	4.296	**2.010**	11.327	**7.922**	0.368	**0.255**	0.569	**0.677**	0.807	**0.909**	0.904	**0.969**
$40-50m$	0.264	**0.206**	5.140	**3.047**	14.167	**10.952**	0.404	**0.300**	0.518	**0.598**	0.760	**0.845**	0.888	**0.946**
$50-60m$	0.270	**0.225**	6.298	**4.441**	17.267	**14.398**	0.440	**0.346**	0.483	**0.556**	0.746	**0.789**	0.854	**0.904**
$60-70m$	0.280	**0.248**	8.024	**6.275**	20.725	**18.243**	0.475	**0.385**	0.480	**0.495**	0.692	**0.736**	0.817	**0.868**
$70-80m$	0.289	**0.284**	10.299	**8.802**	24.621	**23.160**	0.505	**0.434**	**0.463**	0.394	0.645	**0.677**	0.776	**0.834**

Tableau 4.5: Évaluation de la profondeur sur des plages de distance pour NuScenes. Les algorithmes évalués sont des méthodes monoculaires d'estimation de la profondeur reconnues: Monodepth2 (MD2) et BTS. Les erreurs de profondeur globales sont indiquées ainsi que les erreurs de profondeur pour des distances de 10m et jusqu'à 80m. La RMSE est exprimée en mètres.

	RE		SRE		RMSE		logRMSE		a_1		a_2		a_3	
Classe de l'objet	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS
Voiture	0.346	**0.218**	6.853	**2.144**	10.420	**6.862**	0.448	**0.278**	0.546	**0.708**	0.736	**0.880**	0.920	**0.949**
Personne	0.501	**0.384**	8.312	**3.910**	9.291	**7.858**	0.531	**0.449**	0.438	**0.492**	0.679	**0.717**	0.803	**0.839**
Bus	0.448	**0.228**	11.837	**2.226**	13.929	**7.811**	0.448	**0.274**	0.465	**0.644**	0.729	**0.891**	0.848	**0.958**
Camion	0.324	**0.218**	6.803	**2.091**	11.425	**7.263**	0.378	**0.260**	0.574	**0.674**	0.793	**0.902**	0.887	**0.964**
Moto	0.284	**0.245**	1.671	**1.430**	4.509	**3.917**	0.320	**0.288**	0.512	**0.658**	0.868	**0.869**	0.935	**0.946**

Tableau 4.6: Évaluation de la distance selon la classe des objets sur NuScenes. Les algorithmes évalués sont des méthodes monoculaires d'estimation de la profondeur reconnues: Monodepth2 (MD2) et BTS. Les erreurs de profondeur ont été calculées pour les classes d'objets ayant suffisamment d'instances dans l'ensemble de test. La RMSE est exprimée en mètres.

4.5.3 Analyse des résultats expérimentaux

Nos résultats sur les deux jeux de données montrent que, globalement, BTS donne de meilleurs résultats que Monodepth2. Notre évaluation sur les plages de distance montre également que les deux méthodes, comme prévu, ont tendance à avoir une précision plus faible lorsque la distance augmente. Notre évaluation de la profondeur des objets montre également que les erreurs de prédiction de la profondeur sont significativement plus élevées que les erreurs sur l'image globale (voir la Figure 4.8). Cela peut s'expliquer par la grande variété de chaque classe d'objets qui rend l'apprentissage de la profondeur plus difficile pour les CNN, alors que l'environnement environnant est moins variable, ce qui facilite l'apprentissage de la profondeur par ces méthodes. Nous pouvons voir que les erreurs peuvent être 2 fois plus élevées que l'erreur globale pour les personnes et les voitures, et cela doit être pris en compte si ces méthodes sont utilisées dans le cadre du véhicule autonome. Enfin, nous pouvons voir que nos résultats sur la base de données NuScenes ont des erreurs plus élevées que sur KITTI, ce qui peut s'expliquer par les différences dans l'entraînement entre les deux jeux de données, notamment que NuScenes possède des scènes plus difficiles pour l'estimation de la profondeur. Par exemple, certaines scènes ont été acquises par temps de pluie et présentent des réflexions dues à la route mouillée. Des scènes ont également été capturées de nuit avec une mauvaise visibilité. Ces scènes ont une erreur beaucoup plus élevée que celles ayant une bonne visibilité et cela contribue à augmenter les erreurs moyennes utilisées pour calculer l'erreur globale. En combinant nos deux protocoles d'évaluation, nous avons également calculé l'évolution de l'erreur pour des objets tels que les voitures sur des plages de distance (voir Figure 4.7). Ces résultats comparatifs peuvent être utilisés pour évaluer l'adéquation d'une méthode d'estimation de la profondeur à un scénario particulier dans les environnements routiers. Par exemple, pour la conduite sur une route dans des conditions idéales, où nous supposons que le véhicule roule à 90 km/h, la méthode doit être précise jusqu'à 60 m (distance de sécurité entre deux véhicules à cette vitesse).

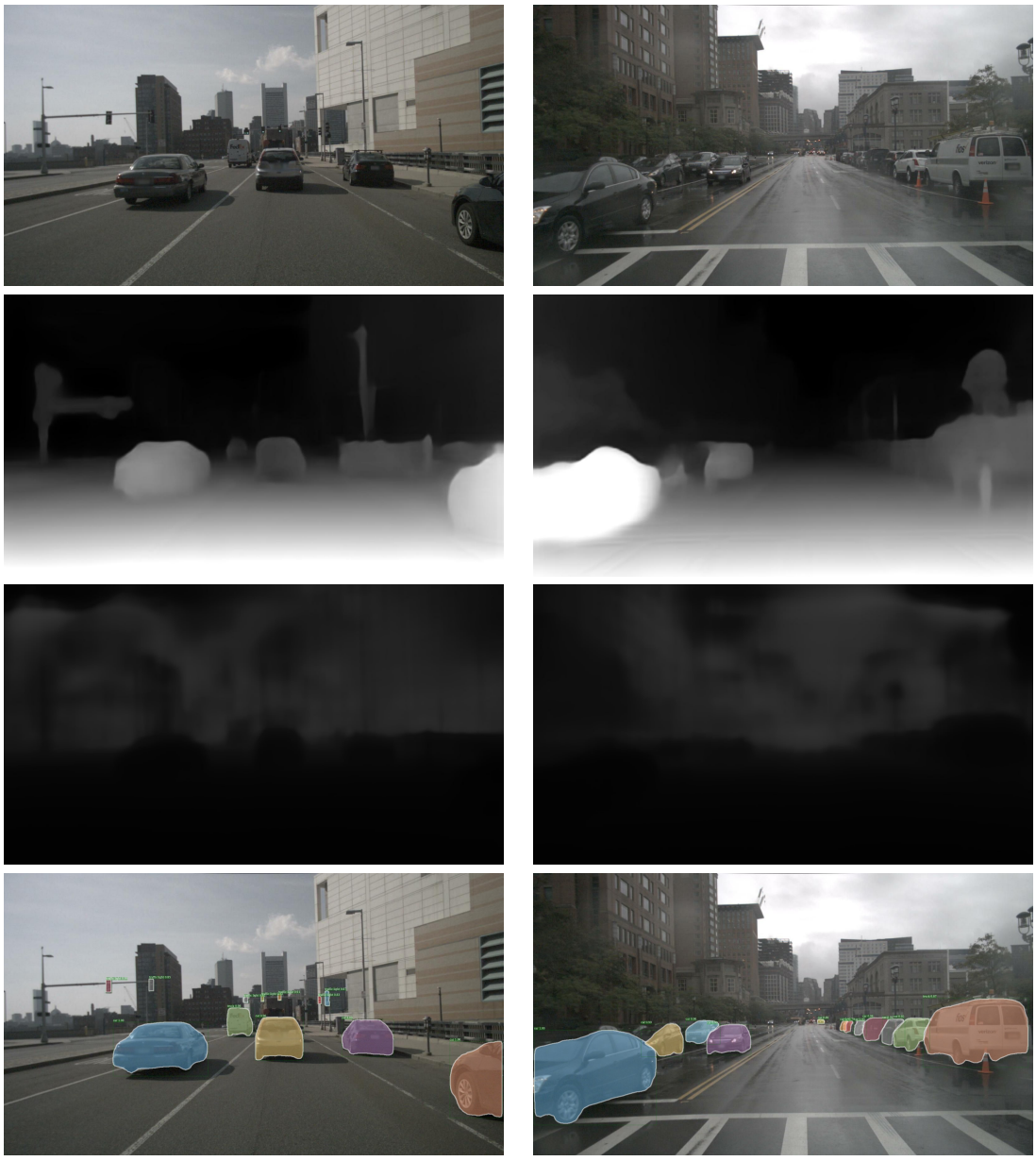

FIGURE 4.6: Données d'entrée pour notre protocole d'évaluation de la distance selon la classe des objets. (1) l'image d'entrée alimentant l'algorithme de prédiction de la profondeur, (2) la carte de disparité, (3) la carte de profondeur normalisée après mise à l'échelle médiane et (4) les masques des objets de Mask-RCNN.

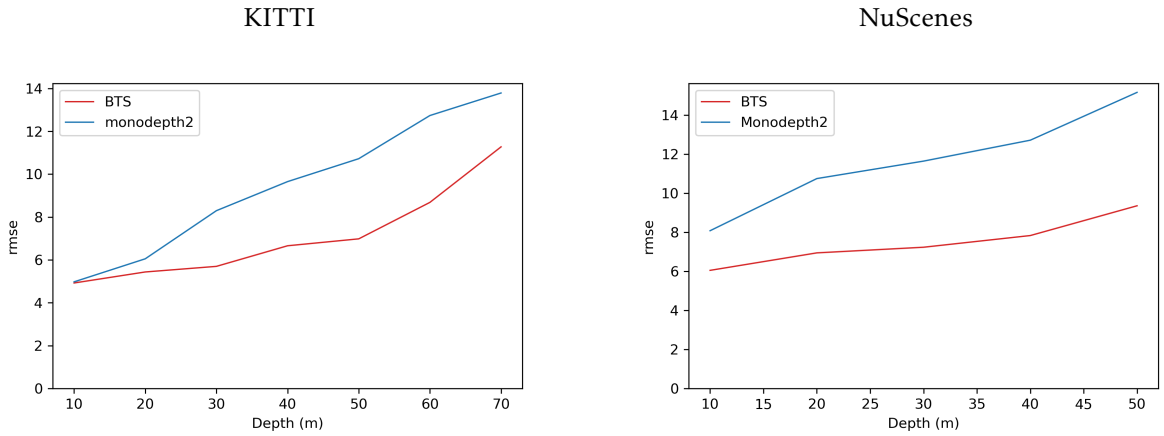

FIGURE 4.7: RMSE pour la classe d'objets voiture selon la distance pour les bases de données KITTI et NuScenes. La RMSE est exprimée en mètres.

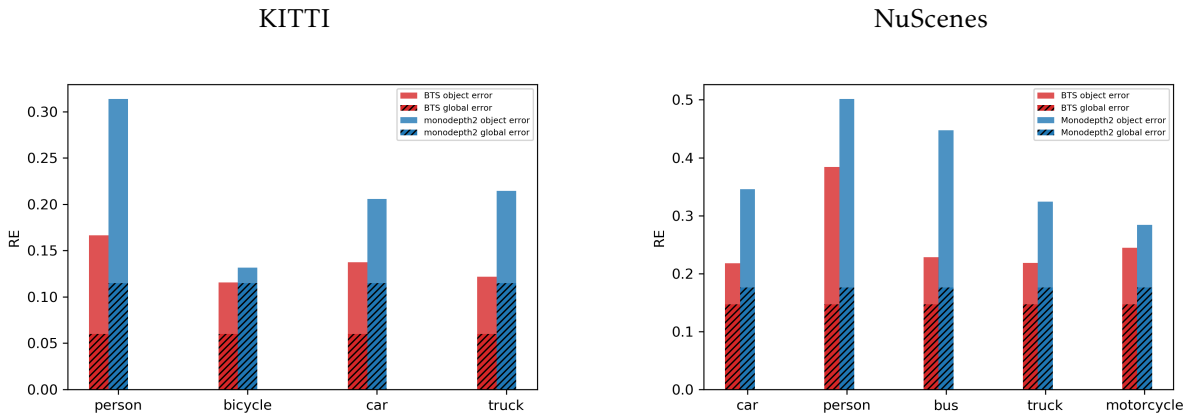

FIGURE 4.8: Nos résultats d'erreur relative (RE) de BTS et Monodepth2 pour différentes classes d'objets comparés à l'erreur relative (RE) globale (hachuré) sur KITTI et Nuscenes.

4.6 Conclusion

Dans ce chapitre, nous avons présenté nos travaux visant à approfondir l'évaluation des méthodes d'apprentissage profond pour l'estimation de profondeur. Dans un premier temps, nous avons enrichi notre étude comparative en évaluant de nouvelles méthodes stéréoscopiques et monoculaires pour l'estimation de profondeur. Nous avons aussi conçu notre propre protocole d'évaluation pour l'estimation de profondeur. Ce nouveau protocole a permis d'évaluer la performance des approches spécifiquement pour l'estimation de la profondeur des objets ainsi que de mesurer les performances de celles-ci en fonction de la distance.

Grâce à ces travaux, nous avons fait les observations suivantes:

(1) Les méthodes d'estimation de profondeur basées sur des images stéréoscopiques sont nettement plus précise que celles basées sur des images uniques

(2) La précision de l'estimation de la profondeur se dégrade nettement lorsque la distance augmente

(3) L'estimation de la profondeur des points appartenant à un objet (donc l'estimation de la distance d'un objet) présente une détérioration notable de la précision vis-à-vis de l'estimation de profondeur sur l'image globale.

Ces observations ont permis de mettre en avant la nécessité d'utiliser une méthode dédiée mêlant détection et localisation d'objets. Notre protocole d'évaluation a mis en avant les problèmes de précision des méthodes d'estimation de profondeur pour localiser les objets dans l'espace.

Chapitre 5

ase de Données Multimodales Routière/Ferroviaire

5.1 Introduction

Afin de combler le manque de bases de données ferroviaires avec une vérité de terrain permettant la détection et la localisation des objets, nous nous sommes intéressés à la conception de notre propre base de données. En prenant en compte à la fois les similarités entre le domaine routier et celui ferroviaire ainsi que la complexité inhérente de l'acquisition de données dans le domaine ferroviaire, nous avons fait le choix de nous orienter sur une base de données multimodale routière et ferroviaire. Cette nouvelle base de données nous permettra à la fois d'entraîner, mais aussi de valider nos approches dans un environnement ferroviaire. Pour ce faire, nous comptons fournir une vérité de terrain riche pour la détection et la localisation des objets (classe, boite englobante 2D, position dans l'espace 3D, dimensions en mètres, orientation de l'objet, etc.).

Dans ce chapitre, nous présenterons notre première base de données virtuelle conçue grâce à un jeu vidéo et notre seconde base, réelle, acquise dans les villes de Rouen et Le Havre.

5.2 Base de données virtuelle crée à partir d'un jeu vidéo

Pour compenser le manque de données réelles dans l'environnement ferroviaire, nous nous sommes tournés vers un jeu de données virtuel. Le principal avantage d'un jeu de données virtuel est la simplification de son acquisition et de son annotation. Cependant, cela se fait au détriment de la fidélité par rapport à la réalité. En effet, la plupart de ces jeux de données provenant de simulateurs (tels que CARLA [73] ou SYNTHIA [74]) n'ont pas une fidélité graphique suffisante pour permettre aux méthodes entraînées sur ceux-ci d'offrir de bonnes performances une fois appliquées dans des conditions réelles. Pour surmonter ce problème, nous nous tournons vers les jeux vidéo afin d'acquérir une base de données ferroviaire pour la détection d'objets 3D. En effet, là où les simulateurs manquent de fidélité graphique, les jeux vidéo tentent d'offrir la meilleure fidélité graphique pour offrir aux utilisateurs un sentiment d'authenticité. C'est pourquoi plusieurs travaux ont déjà été réalisés dans ce domaine pour extraire des bases de données. L'un des jeux les plus intéressants est le jeu vidéo Grand Theft Auto V, qui permet à l'utilisateur de conduire des véhicules routiers et présente des scènes routières réalistes avec, par exemple, des passages à niveau, des passages piétons, un trafic intense, etc. De nombreux travaux ont été réalisés dans ce sens pour construire des bases de données pour l'environnement routier [75, 76]. Notre base de données ferroviaire a été acquise avec un pipeline d'acquisition de données basé sur le travail de [77] et de [78]. Pour ce faire, nous avons utilisé une version modifiée du code du jeu [79] pour permettre à l'utilisateur de conduire un métro ou un train, ce qui nous permet de construire une base de données hybride avec des scènes routières et ferroviaires qui peuvent ensuite être utilisées pour l'entraînement et l'évaluation de notre approche de détection 3D.

5.2.1 Méthode d'acquisition

Afin d'acquérir cette nouvelle base de donnée, nous avons utilisé une version du jeu Grand Theft Auto V modifiée, [77, 78]. Cette version nous permet d'extraire directement les données de l'API du jeu. Les données extraites incluent les données des entités présentes dans l'environnement proche du joueur

comme la classe de l'entité (personne, type de véhicule, etc.), leurs positions dans le repère global du jeu, les matrices de projections, les conditions météorologiques, la position de la caméra dans le repère global, etc.

Via le code du jeu modifié, nous pouvons aisément modifier le jeu afin d'extraire des données adaptées à nos besoins, nous avons ainsi ajouter une seconde caméra pour imiter l'acquisition d'une caméra stéréoscopique. Pour acquérir cette base automatiquement, nous avons aussi utilisé une fonction de pilote automatique présent dans le jeu pour contrôler notre véhicule et nous avons fait usage d'une autre modification du jeu [79] dans le but de pouvoir contrôler les métros et les trains présents dans le jeu.

Toutes les données sont ensuite stockées dans une base de données SQL et nécessitent un post-traitement dans le but de transformer ces données pour être utilisables lors de l'apprentissage de nos méthodes de détection 3D. Les données que nous avons sélectionnées pour l'apprentissage sont détaillées ci-dessous:

(1) Classes des objets

(2) Position et dimensions des boites englobantes 2D des objets présents sur l'image en pixels

(3) Orientation des objets

(4) Position des objets relativement à la caméra en mètres

(5) Dimensions des objets en mètres

5.2.2 Préparation de la vérité de terrain de la base de donnée

Après l'acquisition de la base de données, il est nécessaire d'effectuer un traitement supplémentaire afin de créer la vérité de terrain utilisable pour les méthodes d'apprentissage profond. En effet, dans GTA V toutes les coordonnées des objets sont en repère monde en une unité que nous pouvons associer au mètre. Du fait que nous utilisons un jeu vidéo, la reprojection des points dans le repère monde sur le repère de l'image en pixels est différent. En effet, le jeu ne permet pas d'obtenir une matrice intrinsèque et une matrice extrinsèque afin de faire la reprojection, mais à la place nous avons la matrice de vue (similaire à la matrice extrinsèque) et la matrice de projection. Dans le but de projeter des coordonnées dans le repère monde sur le repère image en pixels, nous devons donc suivre le procédé ci-dessous:

Coordonnées monde – > (matrice de vue) – > Coordonnée caméra – > (Matrice de projection) – > NDC (*Normalized device coordinate*) – > (redimmensionnement) – > pixels dans l'image

Donc en utilisant la relation ci-dessous:

$$\begin{pmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \\ 1 \end{pmatrix} = \begin{pmatrix} matrice\ de\ vue \\ (4\times 4) \end{pmatrix} \begin{pmatrix} X_{monde} \\ Y_{monde} \\ Z_{monde} \\ 1 \end{pmatrix} \tag{5.1}$$

Nous obtenons les coordonnées dans le repère caméra à partir des coordonnées en repère monde. Ici la matrice de vue correspond à une matrice extrinsèque entre le repère monde et le repère de la caméra. Les coordonnées en pixels peuvent ensuite être calculées via l'équation ci-dessous:

$$\begin{pmatrix} x_{pixels}\cdot k \\ y_{pixels}\cdot k \\ k \end{pmatrix} = \begin{pmatrix} w & 0 & 0 & w \\ 0 & -h & 0 & h \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} matrice\ de\ projection \\ (4\times 4) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \\ 1 \end{pmatrix} \tag{5.2}$$

Avec (w,h) la largeur et la hauteur de l'image en pixels. Ici il est nécessaire de normaliser le vecteur des coordonnés pixels afin que la dernière coordonnée soit 1.

Afin de trouver l'orientation des objets selon l'axe Y (azimut), il est nécessaire de constituer la matrice extrinsèque entre l'objet et la caméra. À partir de cette matrice, nous pouvons extraire la matrice de rotation et donc par extension l'azimut. Pour restituer cette matrice, nous utilisons la relation ci-dessous:

$$\begin{pmatrix} \mathbf{T}_{obj->cam} \\ (4\times4) \end{pmatrix} = \begin{pmatrix} matrice\ de\ vue \\ (4\times4) \end{pmatrix} \begin{pmatrix} \mathbf{T}_{monde->obj} \\ (4\times4) \end{pmatrix}^{-1} \tag{5.3}$$

$$\begin{pmatrix} \mathbf{T}_{obj->cam} \\ (4\times4) \end{pmatrix} = \begin{pmatrix} & \mathbf{R} & & \mathbf{t} \\ & (3\times3) & & (3\times1) \\ & & & \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.4}$$

L'API du jeu nous fournit directement le vecteur de rotation et de translation des objets vis-à-vis du repère monde, ce qui nous permet de créer la matrice de transformation $\mathbf{T}_{monde->obj}$ entre le repère monde et le repère de l'objet. **R** représente la matrice de rotation qui va nous permettre de retrouver l'azimut de l'objet vis-à-vis du repère caméra et **t** est la position de l'objet vis-à-vis du centre de ce repère.

5.2.3 Architecture de la base de données multimodale virtuelle GTAV

Notre base de donnée comporte 220 000 images stéréoscopiques divisées en 72 séquences. Parmi ces séquences, 13 sont des séquences prises du point de vue d'un métro et 59 sont prises du point de vue d'une voiture. Les environnements dans lesquels ont été menés les acquisitions varient entre environnement urbain, campagne et autoroutier. Dans la Figure 5.1 sont détaillées différentes statistiques sur la composition de notre base de donnée finale.

Lors de l'acquisition du jeu de données, plusieurs classes d'objets ont été annotées. Cependant, pour combiner notre jeu de données virtuel avec le jeu de données KITTI de détection 3D, nous avons sélectionné uniquement les classes qui sont également présentes sur le jeu de données KITTI (**Voiture, Personne, Cycliste**). Notre jeu de données offre diverses situations et conditions environnementales, ce qui en fait un jeu de données pertinent pour les tâches de vision par ordinateur. Le jeu de données comprend des scènes de nuit, de jour, de pluie et de temps clair, offrant une grande variété de conditions de visibilité. Des exemples provenant de notre base de données peuvent être trouvés dans la Figure 5.2.

5.3 Base de données réelle multimodale routière et ferroviaire

Afin de combler le manque de base de données d'images réelles pour le domaine ferroviaire, nous avons entrepris l'acquisition et l'annotation de notre propre base de données réelle. Cette nouvelle base, nommée ESRORAD (*Esigelec engineering high school and Segula technologies ROad and RAilway Dataset*), a pour but de tester et valider nos approches spécifiquement pour le domaine routier et le domaine ferroviaire. Cette nouvelle base de données sera la première incorporant des acquisitions provenant d'environnements ferroviaires dédiée à la détection d'objets en 3D. Cela est particulièrement important car, malgré les nombreuses similarités entre le domaine ferroviaire et le domaine routier (même type d'objets à détecter, environnement similaire, etc.), il nous est impossible actuellement de confirmer la validité d'une méthode de détection 3D sur le domaine ferroviaire, et ce, à cause du manque de bases de données dédiées. Avec cette nouvelle base, nous pourrons donc enfin valider (ou infirmer) la fiabilité de différentes approches pour le ferroviaire.

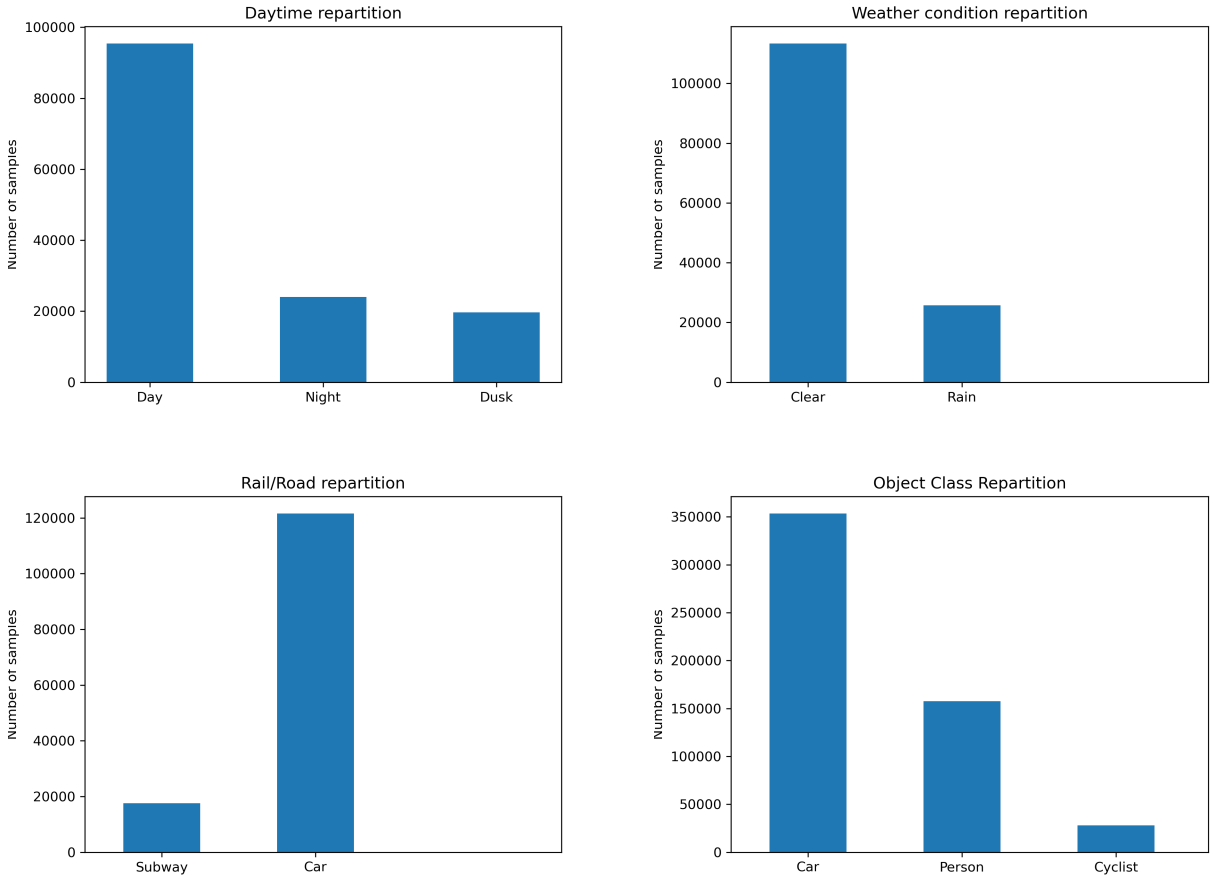

FIGURE 5.1: Détails statistiques pour notre nouvelle base de données: En haut à gauche, la distribution des images en fonction du jour, en haut à droite, la distribution des conditions météorologiques, en bas à gauche, la distribution des routes/chemins de fer, en bas à droite, la distribution des classes.

Nous visons à créer une base de donnée hybride incorporant à la fois des images routières et ferroviaires afin d'obtenir une base de données plus complète. Nous avons entrepris l'acquisition de 80 000 échantillons pour cette base de donnée avec une création de vérité de terrain progressive jusqu'à obtenir une base avec quelques milliers d'échantillons annotés pour nous permettre d'entreprendre une validation pertinente de nos algorithmes. Ces échantillons seront constitués d'acquisitions provenant d'une caméra stéréoscopique, pour les images RGB, et de balayages provenant d'un LiDAR pour l'acquisition d'une carte de points 3D. Ces données nous permettrons ensuite de créer une vérité de terrain pour la détection d'objets en 3D.

5.3.1 Architecture du système d'acquisition

Les principales difficultés liées à la création de bases de données d'images sont dues aux techniques requises telles que la mise en œuvre du système d'acquisition, composé d'une caméra et d'un LiDAR, qui doit être calibré et synchronisé. Pour réaliser ces acquisitions dans les conditions de trafic les plus réalistes possibles, nous avons opté pour l'utilisation de notre véhicule de test IRSEEM (voir Figure 5.3) comprenant les dispositifs suivants:

(1) Caméras stéréoscopiques Intel RealSense (L515).

(2) GPS AsteRx (septentrio).

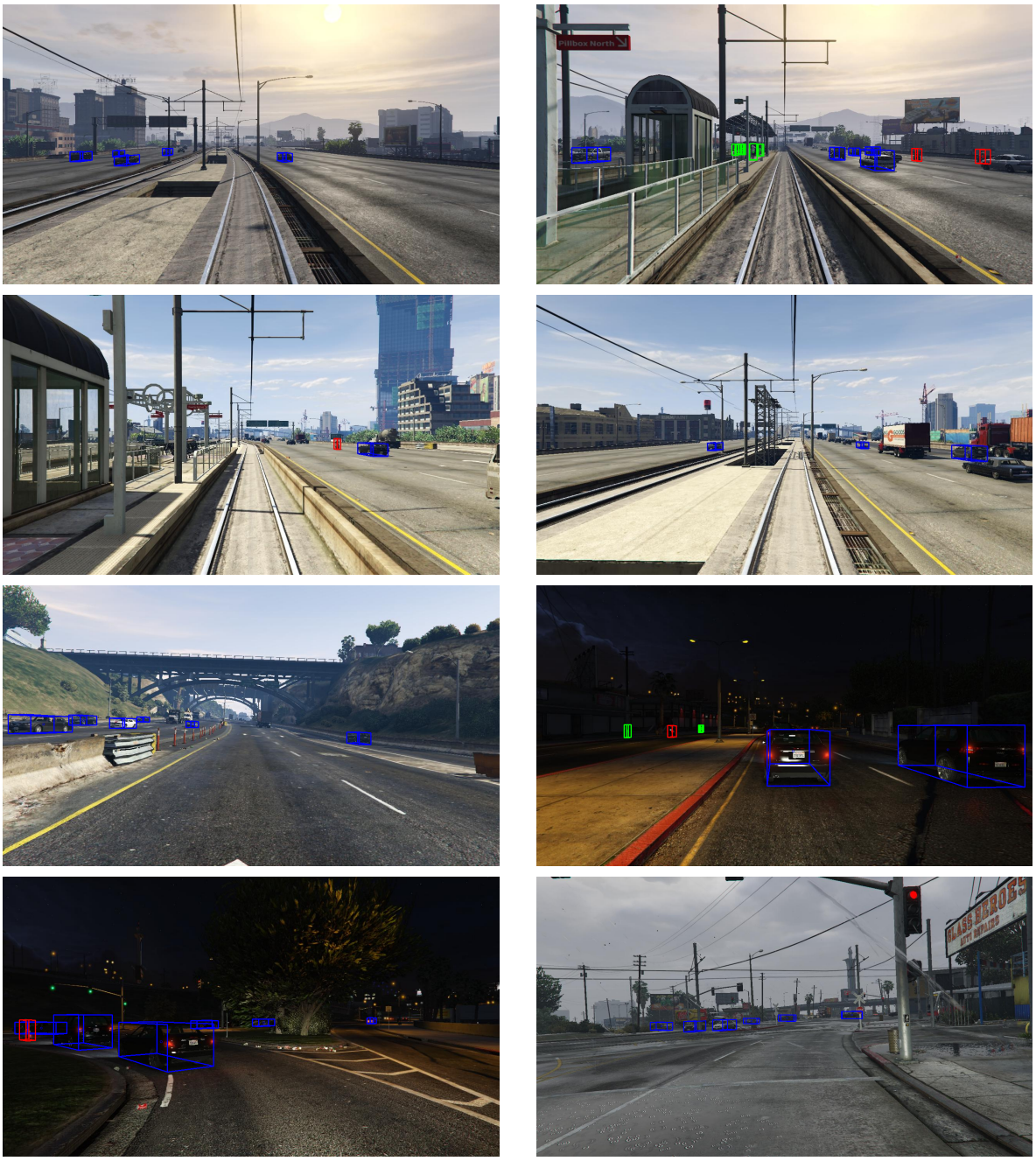

Figure 5.2: Images avec vérités de terrain en 3D. Notre jeu de données présente une variété d'environnements et de conditions. Les vérités de terrain présentées correspondent aux mêmes classes que KITTI: voiture, personne et vélo.

(3) Unité de mesure inertielle (IMU) LANDYN (IXblue) composée d'un accéléromètre, d'un magnétomètre et d'un gyroscope. Le système dispose également d'un logiciel de post-traitement APPS.

(4) Odomètre monté sur la roue arrière droite

(5) VLP16 LiDAR type Velodyne synchronisé avec le GPS

(6) Un système d'acquisition de données en temps réel RTMAPS (Intempora).

On trouve tout d'abord un GPS qui va permettre une géolocalisation du véhicule en temps réel. L'utilisation du GPS seul a néanmoins quelques limites, surtout lorsqu'on souhaite une précision centimétrique. En effet, on peut se trouver en présence d'erreurs de positionnement (appelées sauts GPS) à proximité

FIGURE 5.3: Véhicule IRSEEM instrumenté pour la collecte de données dans la mobilité intelligente. Le coffre de la voiture sur le toit du véhicule comprenant l'ensemble des capteurs (Caméras, GPS/IMU, LiDAR, RADAR, Odomètre). Le capteur de l'odomètre instrumenté sur la roue arrière droite.

de bâtiments ou de structures imposantes, le signal GPS pouvant être, dans ce cas, perturbé lors de la transmission par satellite. Pour améliorer la précision de la géolocalisation, nous avons opté pour une localisation DGPS (Differential GPS): des balises géoréférencées vont renvoyer l'erreur de positionnement au véhicule et permettre une correction en temps réel via, dans notre cas, le réseau "Teria".

Cependant, une position précise n'est pas suffisante pour la navigation, des informations sur l'orientation sont également nécessaires, d'où l'intérêt de la fusion de données. Pour cela, le véhicule est équipé d'une IMU combinant gyroscopes, accéléromètres et magnétomètres. Le gyroscope permet de mesurer l'orientation et la vitesse angulaire. Dans notre cas, il s'agit d'un gyroscope à fibre optique 3 axes (FOG), une technologie qui assure un très faible décalage et bruit entre deux positions GPS. Un logiciel de post-traitement est nécessaire pour améliorer la précision en filtrant (filtre de Kalman avant/arrière) [16, 80] les sauts et anomalies GPS en termes de localisation en temps réel. La fusion de l'IMU et du DGPS permet de disposer de toutes les données de position et de rotation. Le DGPS empêche la dérive de l'IMU, tandis que la centrale inertielle empêche les sauts du GPS.

Le capteur multi-nappes Velodyne LiDAR VLP16 permet d'obtenir une carte de distance sur 16 couches simultanément. Chaque mesure de distance est obtenue à une distance angulaire fixe par rapport à la mesure précédente. Il est donc possible de convertir la carte de distance en un ensemble de points 3D, appelé

nuage de points (par exemple, Figure 5.4).

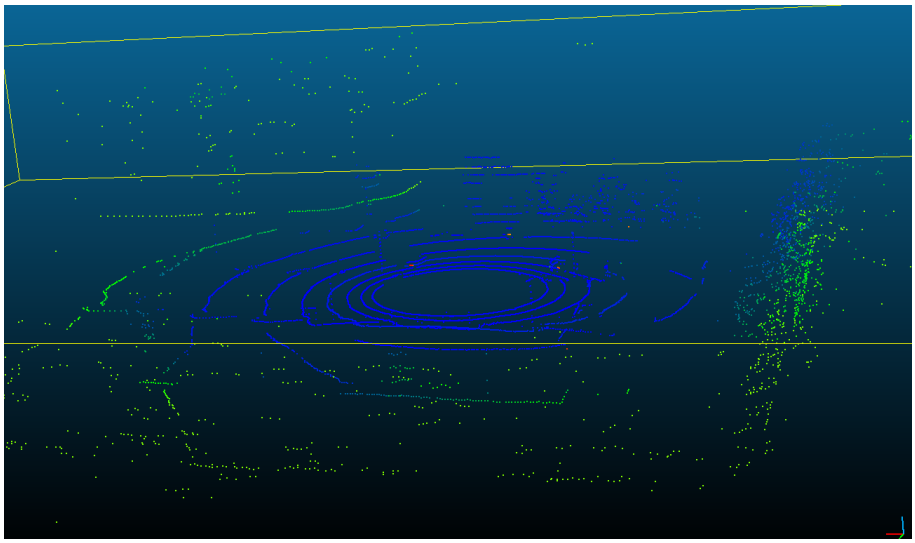

FIGURE 5.4: Un exemple de points de nuage 3D acquis par le Velodyne VLP16 LiDAR.

En plus des données LiDAR, le système d'acquisition décrit ci-dessus comprend également une caméra stéréoscopique permettant de collecter des images RVB de la scène routière et ferroviaire. L'objectif sera alors de synchroniser chaque scan LIDAR avec la capture d'image de la caméra correspondant au même moment d'acquisition, puis de projeter le nuage de points LiDAR enregistré sur l'image. Les caméras RealSense L515 ont une bonne dynamique d'image RVB en extérieur et nous permettrons d'obtenir des images de bonnes qualité dans ces conditions.

5.3.2 Calibrage et synchronisation du système d'acquisition

Calibrage des capteurs

L'opération de calibrage de la caméra correspond à la détermination de la relation entre les coordonnées spatiales d'un point dans l'espace et le point associé dans l'image capturé par la caméra. Nous utilisons le modèle du sténopé qui modélise une caméra par une projection perspective en transformant un point de l'espace en un point de l'image. Cette transformation peut être décomposée en trois transformations élémentaires successives (par exemple, Équations (5.5) et Équation (5.6)).

La première étape de la calibrage vise à déterminer la matrice de transformation entre le référentiel du LiDAR et le référentiel de la caméra (dont l'origine est située au centre optique de la caméra). Cette transformation peut être décomposée en une rotation $\mathbf{R}$ et une translation $\mathbf{t}$. Tous ces paramètres sont nécessaires pour projeter les points LiDAR sur l'image enregistrée par la caméra. Il est alors nécessaire de réunir les paramètres intrinsèques et extrinsèques représentant le passage du repère de la caméra au LiDAR. Les paramètres intrinsèques sont déterminés à l'aide d'une mire d'étalonnage constituée de damiers noirs et blancs.

Les paramètres extrinsèques sont déterminés en utilisant la fonction LiDAR de la caméra. L'objectif étant de superposer les scans LiDAR du véhicule avec les scans LiDAR de la caméra L515 pour obtenir la matrice de transformation entre les deux repères. Une cible d'étalonnage en forme de cube est placée devant la caméra et les capteurs LiDAR. Elle est constituée de trois plans orthogonaux avec des motifs connus. Une fois les motifs détectés, la matrice de transformation extrinsèque est calculée de la caméra au LiDAR en alignant les plans de la cible de calibrage. Ce procédé est illustré dans la Figure 5.5.

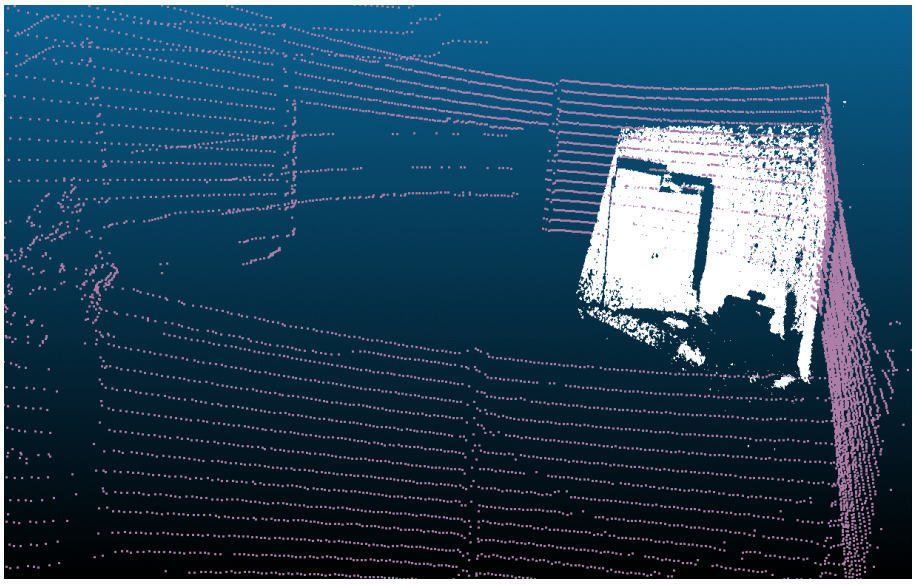

FIGURE 5.5: Alignement des points du nuage LiDAR de la caméra RealSense *L*515 (blanc) sur le point du nuage du véhicule LiDAR (violet).

Projection des points LiDAR sur l'image 2D

Afin de vérifier que les matrices de passage résultantes sont correctes, la matrice de transformation est appliquée aux images et aux nuages de points. Ceci permet d'afficher chaque point LiDAR sur l'image. Tout d'abord, nous récupérons le fichier des points LiDAR dont les coordonnées sont exprimées dans le repère de référence du LiDAR. Ces points sont ensuite projetés dans le repère de la caméra grâce à la matrice extrinsèque entre la caméra et le LiDAR comme détaillé dans l'Equation (5.5). Soit $(X_{LiDAR},Y_{LiDAR},Z_{LiDAR})$ et $\mathbf{T}_{LiDAR->cam}$ les coordonnées d'un point dans le repère du LiDAR et la matrice extrinsèque entre le LiDAR et la caméra. Les coordonnées de ce même point dans le repère de la caméra $(X_{cam},Y_{cam},Z_{cam})$ sont obtenues à la fin de l'opération.

$$\begin{pmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{T}_{LiDAR->cam} \\ (4\times 4) \end{pmatrix} \begin{pmatrix} X_{LiDAR} \\ Y_{LiDAR} \\ Z_{LiDAR} \\ 1 \end{pmatrix} \tag{5.5}$$

Ce point en coordonnée de la caméra doit ensuite être projeté grâce à la matrice intrinsèque de la caméra afin d'obtenir ses coordonnées en pixels. Avec (u,v) les coordonnées en pixels du point et f_x, f_y la focale horizontale et verticale de la caméra et enfin (c_x,c_y) le centre optique, la projection est détaillée dans l'Equation (5.6):

$$\begin{pmatrix} u\times Z_{cam} \\ v\times Z_{cam} \\ Z_{cam} \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{pmatrix} \tag{5.6}$$

(u,v) peuvent être retrouvés en normalisant le résultat de cette opération afin que la dernière coordonnée du vecteur soit 1. La Figure 5.6 illustre la cohérence cohérence des paramètres de calibrage obtenus. En effet, on peut remarquer que les points sont bien projetés dans les coins du mur ou de l'armoire.

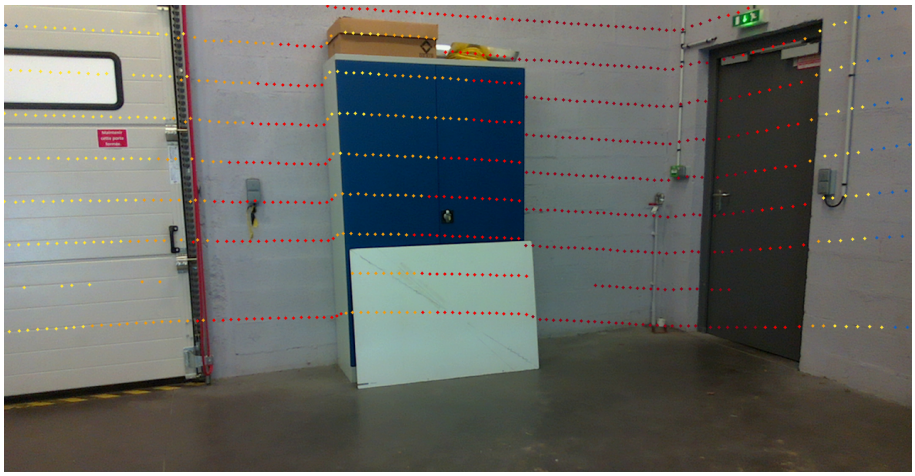

FIGURE 5.6: Résultat de la projection des points LiDAR sur l'image en statique.

Synchronisation caméra-LiDAR

Lors de la partie précédente de calibrage, la projection, en statique, des points LiDAR sur l'image 2D ne posait pas de problème puisque le véhicule était à l'arrêt, ainsi, même si le LiDAR et la caméra fonctionnent à des fréquences de déclenchement différentes, la scène reste la même. En dynamique, ce décalage de déclenchement est plus problématique. En effet, pour réaliser la projection sur des scènes en mouvement, il est nécessaire d'associer chaque scan LiDAR à l'image correspondante qui a été capturée au même moment. Nous avons donc développé un protocole de validation pour vérifier la synchronisation des données de la caméra et du LiDAR comme le montre la Figure 5.7.

Pour obtenir un alignement intermodal correct des données entre le LiDAR et les caméras stéréoscopiques, l'exposition d'une caméra est déclenchée lorsque le LiDAR supérieur balaie le centre du champ de vision de la caméra. L'horodatage de l'image représente l'heure de déclenchement de l'exposition et l'horodatage du balayage LiDAR représente l'heure à laquelle la rotation complète de l'image LiDAR actuelle est atteinte. Comme le temps d'exposition de la caméra est presque instantané, cette méthode donne généralement un bon alignement des données. Les caméras fonctionnent à $30Hz$ alors que le LiDAR fonctionne à $20Hz$. Il faut ensuite faire correspondre un balayage LiDAR à une image en identifiant les temps de capture.

La Figure 5.8 illustre la cohérence de la profondeur matérialisée par l'aspect dégradé des couleurs (bleu pour les objets proches, rouge pour les objets éloignés) ainsi que les contours adoptés par les points LiDAR pour certains objets comme l'arbre ou le véhicule. Il est possible que les réflexions sur la carrosserie créent des points parasites.

5.3.3 Collecte des données

Sélection de l'itinéraire pour la collecte des données

Pour réaliser la collecte de données, nous avons planifié un protocole d'enregistrement prenant en compte deux agglomérations normandes: Rouen et Le Havre. Elles possèdent un important réseau de circulation routière et ferroviaire comprenant non seulement des routes et des tramways, mais aussi des zones sur lesquelles la circulation automobile est autorisée sur la voie ferrée. Cela nous a permis d'acquérir des scènes routières ou ferroviaires avec une vue directe sur les voies. Cela nous a également permis de collecter des données dans des conditions normales, sans interruption du trafic du tramway.

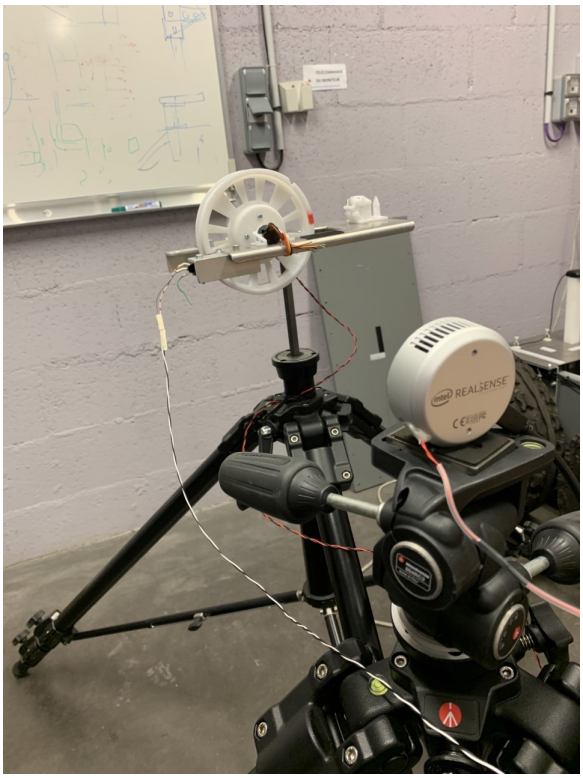

FIGURE 5.7: Protocole de validation de la synchronisation des données de la caméra et du LiDAR.

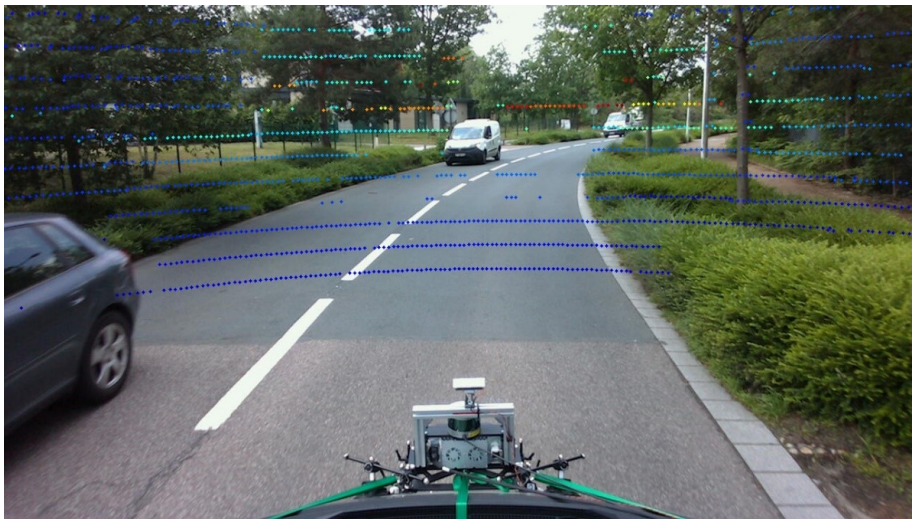

FIGURE 5.8: Résultats de la projection des points LiDAR sur l'image en dynamique.

La Figure 5.9 montre un exemple de scène où le véhicule de test enregistre des données dans un environnement multimodal route/rail.

Notre jeu de données comprend différentes scènes enregistrées: route uniquement, chemin de fer uniquement (tramway ou train) ou hybride route et chemin de fer. Nous avons pu collecter environ $100k$ d'images réparties entre la ville du Havre ($45k$ d'images) et la ville de Rouen ($55k$ d'images).

Protocole d'acquisition des données

Avant toute collecte de données, une initialisation du système d'acquisition est nécessaire et se fait en deux modes: 1. Statique (arrêt du véhicule): l'IMU basée sur le FOG est assez sensible pour mesurer la rotation de la terre et ainsi pouvoir donner un cap en statique. Une période d'attente de 10 minutes

FIGURE 5.9: Collecte de données sur l'environnement ferroviaire dans la ville du Havre (France).

est nécessaire pour initialiser le système. 2. Dynamique (le véhicule en mouvement): l'idée est de se déplacer avec le véhicule pendant quelques minutes pour faire converger le filtre de Kalman étendu (EKF) [80] en réduisant l'erreur de cap. Ensuite, l'enregistrement des données peut commencer et cette étape d'initialisation est également répétée à la fin du processus d'enregistrement.

5.4 Processus d'annotation des données

5.4.1 Logiciels d'annotation de jeux de données

Après plusieurs jours de collecte de données, les données LiDAR sont exportées, puis une reprojection est exécutée sur les différentes acquisitions (comme montré dans la Figure 5.10).

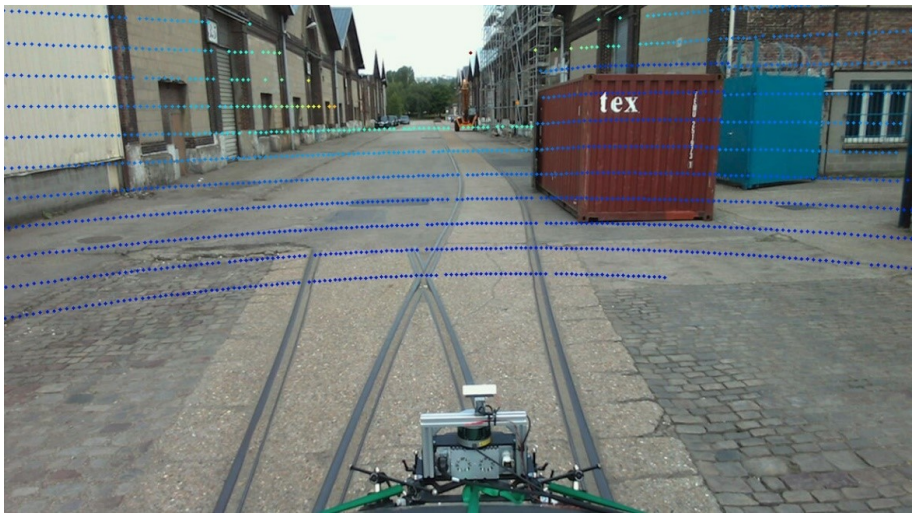

FIGURE 5.10: Aperçu du résultat de la projection des points LiDAR sur une image du jeu de données ferroviaires collecté dans la ville du Havre (France).

Une fois la projection validée, la partie post-traitement commence, c'est-à-dire la documentation des images avec la vérité terrain. L'objectif était pour chaque image, d'identifier dans l'environnement nos trois classes principales (véhicule, piéton, cycliste) et également la distance de chaque objet par rapport au véhicule. Toutes ces informations constitueront la vérité terrain du jeu de données réel qui sera dédié à l'entraînement et au test des algorithmes de détection d'objets 2D et 3D. Nous avons donc réalisé un Benchmark de tous les outils d'annotation existant dans la littérature. L'idée est d'afficher les nuages de points LiDAR et de placer manuellement des boîtes 3D correspondant à la classe de l'objet identifié, puis

d'afficher le résultat sur l'image couleur. Cette étape d'étiquetage sera la dernière étape du développement du jeu de données.

Dans un premier temps, nous nous sommes intéressés aux entreprises offrant soit un service d'annotations soit un logiciel pour créer la vérité de terrain des boites englobantes 3D nécessaire pour la détection d'objets en 3D. Nous avons identifié Supervisely [81] et Scale [82], et bien qu'ils proposent une offre fiable, la solution offerte n'était pas Open-source. Notre objectif étant d'avoir à disposition un outil accessible et modulable afin de s'adapter à nos besoins, nous avons fait le choix de nous orienter vers des outils gratuits et Open-source.

Nous avons finalement utilisé un logiciel d'annotation 3D semi-automatique basé sur JavaScript pour les flux de données multimodaux appelé "3D Bounding Box Annotation Tool" (BAT 3D) [83] qui est un nouveau système d'annotation Open-source consacré à l'annotation d'ensembles de données 2D et 3D. Il est efficace, précis et comprend des outils pour la localisation d'objets en 3D à l'aide de flux de données multimodaux en continu. Il comporte également de nombreuses fonctionnalités que les autres outils d'annotation 3D n'offrent pas, comme par exemple une option d'interpolation afin de faciliter l'annotation des séquences d'images. L'outil se concentre sur la tâche difficile de l'annotation d'images 2D et de nuages de points 3D. L'utilisateur peut annoter des scènes directement en 3D et transférer ces annotations dans le domaine de l'image. Les données laser sont d'abord annotées avec des primitives de délimitation grossières, puis un modèle géométrique est utilisé pour transférer ces étiquettes dans l'espace image. L'interface de cet outil est présenté dans la Figure 5.11.

Figure 5.11: Interface de BAT 3D avec une vue de l'image frontale tirée de notre jeu de données dans la ville du Havre.

5.5 Résultats

Nous avons, à ce jour, annotés 2 500 images avec Bat-3D. Nous avons fait le choix, dans un premier temps, d'annoter que les images provenant du point de vue de droite de la caméra stéréoscopique. En effet, nous concentrer sur l'annotation d'un seul point de vue va nous permettre de valider dans un premier temps nos approches de détection d'objets en 3D sur une partie de cette base de donnée. Le nombre d'images n'est actuellement pas suffisant pour permettre un entraînement, elle pourront néanmoins être

utilisés pour la validation de nos approches. La vérité de terrain obtenue suite à notre processus d'annotation est décrite ci-dessous:

— Classe de l'objet

— Position de l'objet vis-à-vis du LiDAR (XYZ en mètres)

— Dimensions de l'objet en mètres

— Orientation de l'objet vis-à-vis du LiDAR

Ces images comportent à la fois des scènes routières et des scènes ferroviaire. Les annotations fournies par Bat-3D sont relatives au LiDAR, cependant, afin pouvoir être par la suite utilisé par nos approches, nous devons les transformer afin que celles-ci soient dans le repère caméra et le repère image pour permettre un entraînement pour la détection 2D et 3D des objets. Les annotations que nous cherchons sont décrites ci-dessous:

— Classe de l'objet

— Boite englobante 2D de l'objet

— Position de l'objet vis-à-vis de la caméra (XYZ en mètres)

— Dimensions de l'objets en mètres

— Azimuth de l'objet vis-à-vis de la caméra

Nous pouvons obtenir la position de l'objet dans le repère caméra en utilisant la matrice extrinsèque entre le LiDAR et la caméra et la relation (5.5). De la même façon, nous pouvons trouver la Boite englobante 2D de l'objet en re-projetant sa boite 3D dans le repère caméra grâce à la relation (5.6). L'azimut de l'objet relatif au repère caméra peut être calculé à partir de la matrice de transition entre le repère de l'objet et le repère de la caméra. Cette matrice est obtenue par la relation (5.7):

$$\begin{pmatrix} \mathbf{T}_{obj->cam} \\ (4\times4) \end{pmatrix} = \begin{pmatrix} \mathbf{T}_{LiDAR->cam} \\ (4\times4) \end{pmatrix} \begin{pmatrix} \mathbf{T}_{obj->LiDAR} \\ (4\times4) \end{pmatrix} \tag{5.7}$$

$$\begin{pmatrix} \mathbf{T}_{obj->cam} \\ (4\times4) \end{pmatrix} = \begin{pmatrix} & \mathbf{R} & & \mathbf{t} \\ & (3\times3) & & (3\times1) \\ & & & \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.8}$$

Avec **R** la matrice de rotation entre l'objet et la caméra , qui nous permet de calculer l'azimut, et **t** la matrice de translation. $\mathbf{T}_{LiDAR->cam}$ étant la matrice de transition entre le LiDAR et la caméra et $\mathbf{T}_{obj->LiDAR}$ celle entre l'objet et le LiDAR, qui peut être construite directement à partir des annotations provenant de Bat-3D.

Des exemples d'annotation de notre base de donnée sont présentés dans la Figure 5.12

Bien que seulement 2500 images soient actuellement annotées, ce qui ne permet pas en l'état de mener un entraînement efficace, les travaux sur cette base ne sont pas encore finis. En effet, l'annotation va se poursuivre jusqu'à obtenir un nombre suffisant d'images pour pouvoir mener un apprentissage pertinent. Notre base deviendra donc la première base, à notre connaissance, dédiée à l'apprentissage/évaluation de méthodes d'apprentissage profond pour le domaine ferroviaire. Cette base sera aussi, dans un futur proche, renforcée avec de nouvelles acquisitions avec un capteur LiDAR plus précis.

Figure 5.12: Boites englobantes des objets en 3D provenant de notre vérité de terrain.

5.6 Conclusion

Dans ce chapitre, nous avons présenté nos nouvelles bases de données pour la détection et la localisation d'objets en 3D. Ces deux nouvelles bases ont la particularité d'être composées d'images provenant à la fois de scènes routières, mais aussi ferroviaires. À ce jour, ces deux bases sont les seules à fournir des images provenant de scènes ferroviaires spécifiquement pour la détection et la localisation d'obstacles. Ces jeux de données vont nous permettre à la fois d'entraîner et de valider dans des conditions réalistes les différentes approches que nous allons créer.

Notre première base, virtuelle basée sur le jeu GTAV, comporte 220000 images et offre des scènes de circulation réalistes à la fois pour le domaine routier, mais aussi pour le domaine ferroviaire avec des images prises du point de vue d'un métro. Les conditions sont variées (nuit/jour, météo) et les environnements le sont aussi (Urbain, Campagne, Autoroute, Métro, etc.). Pour ces raisons, cette base est idéale pour le pré-entraînement des méthodes d'apprentissage profond. En effet, grâce à l'apprentissage par transfert (*tranfert learning*), nous pouvons utiliser un modèle pré-entraîné sur notre base GTAV et continuer l'entraînement sur une base réelle, qui contient généralement beaucoup moins d'échantillons, afin d'augmenter la précision finale de la méthode vis-à-vis d'un entraînement "classique". Cette base nous permettra aussi d'évaluer les méthodes dans des scénarios spécifiques (test évitement d'obstacle, etc.) là où sur une base réelle de ce genre de scénarios serait logiquement impossible.

Nous avons aussi présenté notre second jeu de données ESRORAD (*Esigelec engineering high school and Segula technologies ROad and RAilway Dataset*) qui est le premier jeu de données réelles multimodales et hybride avec vérité terrain pour la mobilité intelligente sur route et sur rail. Il comprend des données entièrement développées dans des conditions réelles de trafic routier et ferroviaire. ESRORAD a été conçu pour tester différents ADAS dans les véhicules autonomes comme la détection d'objets en 3D, le suivi d'objets, la segmentation sémantique, l'analyse de scènes et la compréhension. ESRORAD comprend 34 vidéos et $100k$ images réelles (dont 2500 déjà annotées) pour des scènes routières et ferroviaires collectées dans deux villes normandes, Rouen et Le Havre. Les images et les vidéos ont été prises du point de vue de la route et de rails de tramway. Les images sont annotées avec des boîtes englobantes 3D montrant au moins trois classes différentes de personnes, de voitures et de bicyclettes. ESRORAD est un jeu de données en libre accès qui permet aux chercheurs et aux entreprises de tester leurs algorithmes pour les environnements multimodaux routiers et ferroviaires (voitures, trains, tramways, etc.).

De nouvelles données quantitatives sont prévues pour l'avenir afin de compléter cette base, notamment avec l'ajout de plus d'images annotées, davantage de données routières et ferroviaires incorporant plus de scènes et de vidéos dans différentes conditions météorologiques (pluie, neige, soleil, nuages, etc.) et différentes conditions d'éclairage (jour, nuit).

Cette base pourra, à terme, servir de référence pour l'évaluation, voire pour l'entraînement, de méthodes d'apprentissage profond diverses (détection, localisation, etc.) pour le domaine ferroviaire.

Chapitre 6

Détection d'Objets en 3D

6.1 Introduction

Dans ce chapitre, nous nous concentrerons sur l'utilisation de techniques basées sur la vision pour la détection d'objets 3D, qui combinent la détection d'objets et la localisation 3D, afin d'éviter de devoir combiner deux approches différentes pour accomplir ces deux tâches et ainsi alléger le réseau. Ces dernières années, des méthodes basées sur les réseaux de neurones convolutifs (CNN) ont été explorées pour détecter et localiser des objets dans l'espace 3D avec seulement des images comme entrées. L'un des principaux avantages de ces méthodes est la réduction du coût des capteurs. La détection et la télémétrie par la lumière (LiDAR) et le RADAR sont remplacés par une caméra standard, qui est facile à intégrer et peu coûteuse. De nombreuses approches ont été proposées dans la littérature. Cependant, nous constatons un déficit dans l'évaluation de ces approches dans des conditions environnementales réalistes, notamment dans des environnements de trafic ferroviaire.

Nous avons décomposé la détection d'objets en 3D en une liste de différentes prédictions détaillé ci-dessous:

(1) Position de l'objets dans l'image en coordonnées pixels

(2) Classe de l'objet

(3) Distance de l'objet par rapport à la caméra en mètres

(4) Centres 3D de l'objet projetés sur le plan de l'image en pixels

(5) Dimensions de l'objet en 3D

(6) L'angle de lacet de l'objet dans le repère caméra

Grâce à ces prédictions, il est possible de dessiner des boîtes englobantes 3D pour chacun des objets sur l'image et de les localiser dans le repère de notre caméra en mètres.

Nos approches sont principalement basées sur des méthodes de détection d'objets éprouvées afin de créer un nouvel algorithme de détection et de suivi d'objets 3D à partir d'images monoculaires qui prend en compte deux critères importants: la précision et le temps réel pour les environnements routiers et ferroviaires. Alors que les autres méthodes de l'état de l'art se concentrent principalement sur la précision de la détection d'objets 3D ; de plus, dans les applications ferroviaires, nous constatons qu'aucun travail n'est publié sur la détection d'objets 3D. Aucun ensemble de données n'est actuellement disponible dans l'état de l'art qui inclut des données de scènes ferroviaires avec la vérité terrain, ce qui rend l'entraînement impossible. Ainsi, nous utiliserons notre nouveau jeu de données virtuel présenté dans le chapitre précédent pour l'entraînement sur des environnements ferroviaires ainsi que des jeux de données routières bien connus comme KITTI [38] ou NuScenes [39]. Nous testerons ensuite nos méthodes, tant quantitativement que qualitativement, sur nos nouveaux jeux de données basés sur des images réelles présentés dans le chapitre précédent.

6.2 Etat de l'art

Ces dernières années, de nombreuses méthodes basées sur l'apprentissage profond ont été proposées pour la détection d'objets 3D à partir d'images monoculaires. La plupart d'entre elles utilisent des images RVB uniques, et quelques-unes utilisent des séquences d'images. Beaucoup de ces méthodes sont des

extensions de détecteurs 2D éprouvés, tirant ainsi parti de leurs très bonnes performances. Le traitement des paramètres 3D est ensuite ajouté aux modèles. D'autres méthodes infèrent directement les paramètres 3D par un apprentissage de bout en bout.

Dans [84], le problème est abordé en deux étapes. D'abord, des boîtes 3D candidates sont générées et notées en exploitant plusieurs caractéristiques, à savoir la sémantique de classe, la sémantique d'instance, le contour, la forme de l'objet, le contexte et la localisation antérieure. Ensuite, les candidats ayant obtenu les meilleurs scores sont soumis à un détecteur d'objets modifié, ici Faster-RCNN [34], pour prédire les classes d'objets, les décalages des boîtes englobantes et l'orientation des objets.

Dans [85], la détection 2D est effectuée par la proposition de région multi-échelle MS-CNN [86], étendue pour régresser l'orientation et les dimensions des boîtes englobantes 3D. L'estimation de l'orientation utilise un principe MultiBin, dans lequel la prédiction de l'angle est transformée en une tâche hybride de classification et de régression. Les angles possibles sont divisés en deux intervalles de valeurs, le réseau prédit alors dans quel intervalle se trouve l'angle à trouver (classification), dans un deuxième temps le réseau prédit la déviation entre l'angle recherché et le centre de l'intervalle choisi (régression). A partir de là, l'angle peut être prédit.

DeepMANTA [87] est un modèle pour la détection, la localisation, la caractérisation de la visibilité et l'estimation des dimensions 3D simultanées des véhicules. Il est basé sur un processus en deux étapes, dont la première produit des boîtes englobantes associées aux informations sur les véhicules, et la seconde utilise ces sorties et un ensemble de données virtuelles de véhicules 3D pour récupérer les orientations et les localisations 3D.

Dans [88], une carte de profondeur obtenue au moyen d'un réseau entièrement connectés (FCN) est fusionnée avec l'image RVB d'entrée avant d'alimenter un réseau de proposition de région (RPN). Elle est également combinée à différents niveaux du flux pour obtenir finalement une estimation des étiquettes de classe, de la position de la boîte en 2D et de l'orientation, des dimensions et de l'emplacement de la boîte en 3D. Pour l'orientation, le même principe MultiBin que dans [86] est utilisé.

Dans [76], un modèle CNN pour la détection et le suivi conjoints d'objets 3D est proposé. Il est basé sur un RCNN plus rapide pour prédire les boîtes englobantes 2D, qui sont ensuite associées à des informations 3D comprenant la position, l'orientation, les dimensions et les centres des boîtes 3D projetées de chaque objet. Le suivi à travers des séquences d'images est obtenu par deux couches LSTM (Long Short-Term Memory), permettant un raffinement supplémentaire des positions des boîtes de délimitation 3D.

Dans SMOKE [89], un CNN est entraîné de bout en bout en une seule étape au moyen d'une fonction de perte unifiée. Les paramètres des boîtes englobantes 3D sont régressés directement sans utiliser la détection 2D, grâce à un réseau composé de deux branches: une branche de classification où un objet est encodé par son centre 3D (projeté sur le plan de l'image), et une branche de régression des paramètres 3D pour construire une boîte englobante 3D autour de chaque centre.

Dans MonoGRNet [90], la détection d'objets 3D est décomposée en quatre sous-tâches: détection d'objets 2D, estimation de la profondeur du centre de l'objet profondeur du centre de l'objet, estimation du centre 3D projeté et régression des coins locaux. Ces quatre éléments d'information sont obtenus en parallèle par un processus à passage unique, puis combinés.

Des méthodes basées sur des réseaux à étages uniques ont également été proposées dans M3D-RPN [91] et GAM3D [92]. M3D-RPN exploite la convolution en fonction de la profondeur pour localiser les caractéristiques spécifiques et améliorer la compréhension de la scène 3D. L'approche consiste en une seule étape en utilisant des boîtes d'ancrage pour prédire les positions des objets en 2D et en 3D. GAM3D utilise également des boîtes d'ancrage 3D/2D pour prédire la boîte de délimitation 3D des objets et introduit un module de convolution sensible à la profondeur pour fournir des indices 3D supplémentaires au réseau,

améliorant ainsi la précision du réseau.

Dans [93], la détection 3D est reformulée comme un problème de détection de points clés. Un réseau pyramidal de caractéristiques de points clés (KFPN) est défini, fournissant le centre et les points de perspective des boîtes de délimitation 3D. Le fait que l'épine dorsale du réseau soit basée sur une architecture légère (comme ResNet-18 [94] ou DLA-34 [95]) et la nature sans ancrage du détecteur, permettent un traitement en temps réel sur GPU 1080Ti.

La plupart de ces modèles sont destinés à améliorer les performances de la détection d'objets 3D en termes de précision. Quelques autres se concentrent davantage sur le temps de traitement, visant le temps réel sur des GPU puissants. Dans notre cas, nous voulons nous concentrer sur des architectures CNN légères permettant d'atteindre le temps réel sur des GPU de faible puissance comme la Nvidia Jetson-TX2. Nous pensons qu'il est important d'aller plus loin sur ce point pour faciliter le développement à grande échelle de solutions d'apprentissage profond pour la sécurité des véhicules autonomes. Un autre aspect important de nos besoins est de pouvoir être performant aussi bien en environnement routier qu'en environnement ferroviaire. L'absence d'un jeu de données ferroviaires existant nous a conduit à développer notre propre jeu de données, constitué d'images virtuelles issues du jeu GTA-V. Nous utiliserons ce jeu de données personnalisé en complément de KITTI [38].

6.3 Détection d'objets en 3D par approche multi-étages

6.3.1 Vue d'ensemble

L'objectif de notre méthode est de récupérer des boîtes englobantes 3D à partir d'images RVB monoculaires tout en maintenant un temps de calcul faible pour être compatible avec les contraintes du temps réel. Comme la plupart des travaux relatifs à la détection d'objets 3D à partir d'une seule image présentés dans la section précédente, nous nous basons sur les RoIs (Régions d'interets) présentes sur l'image fournis par un détecteur d'objets 2D pour estimer les boîtes englobantes 3D orientées des objets. Contrairement aux autres méthodes de détection 3D à partir d'une image monoculaire, qui s'appuient sur un réseau RPN (Region Proposal Network) distinct, tel que Faster RCNN, pour effectuer la détection 2D, notre méthode est basée sur le détecteur à étage unique YOLOv3 [14] pour effectuer les prédictions de boîte de délimitation 2D. Notre méthode partagera la même base de réseau que Yolov3 afin d'extraire les caractéristiques de l'image, et ce dans le but d'économiser la charge de calcul. Notre architecture de réseau permet de réduire considérablement la consommation de mémoire, tout en surpassant les autres méthodes de l'état de l'art en termes de temps de calcul, au coût d'une précision inférieure. Enfin, notre méthode est entraînée de bout en bout, alors que d'autres modèles exigent que le détecteur 2D soit entraîné séparément, ce qui réduit le temps et le coût d'entraînement de notre méthode. Notre pipeline de détection est décrit dans la Figure 6.1.

6.3.2 Estimation de la boîte englobante en 3D

La boîte englobante 3D d'un objet peut être décomposée en plusieurs éléments: la position de son centre en coordonnées 3D vis-à-vis de la caméra $\mathbf{T} = [x\ y\ z]^\top$ ses dimensions $\mathbf{D} = [w, h, l]$ et son orientation $\mathbf{R}(\phi, \theta, \psi)$ caractérisée par le tangage, le lacet et le roulis. Soit $\mathbf{K}$ la matrice des paramètres intrinsèques de la caméra et $\mathbf{X_o} = [x_o\ y_o\ z_o\ 1]^\top$ un point 3D dans le repère de l'objet, la projection de ce point sur le plan

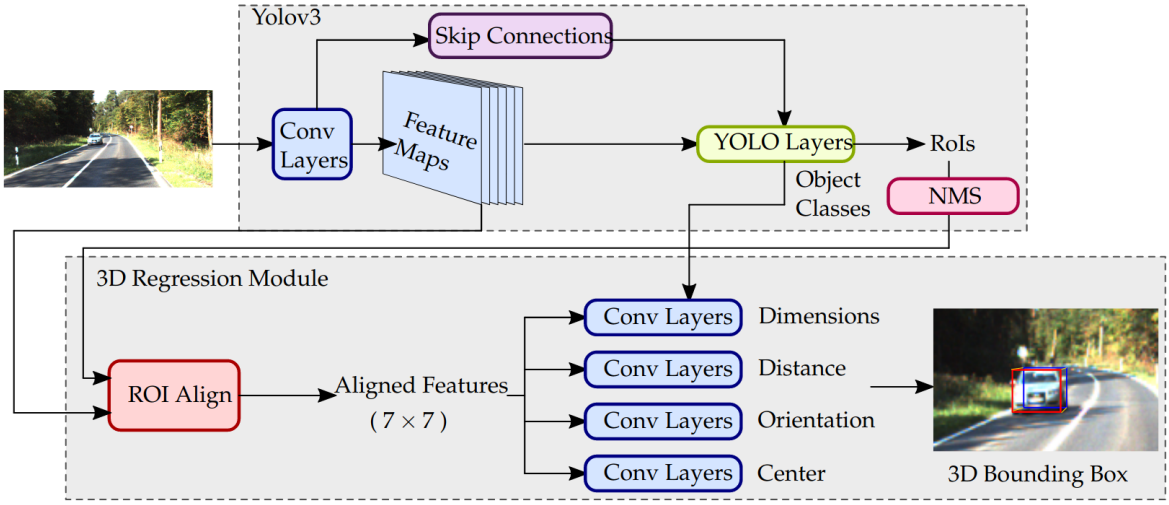

FIGURE 6.1: Illustration de notre détecteur d'objets en 3D. Une seule image RVB est utilisée comme entrée pour notre méthode ; les caractéristiques convolutionnelles partagées sont ensuite extraites par la base du réseau, ici Yolov3. Nous utilisons le détecteur d'objets 2D éprouvé, YOLOv3, pour effectuer la prédiction du RoIs (Régions d'intérêt) et de la classe des objets. Nous extrayons ensuite les caractéristiques des RoIs (Régions d'intérêt) en utilisant l'alignement des caractéristiques utilisé dans [96]. Les paramètres de la boîte de délimitation 3D sont prédits par la prédiction des paramètres de notre CNN, et enfin la boîte de délimitation 3D est dessinée sur l'image.

de l'image $\mathbf{x_{im}} = [u\ v\ 1]^\top$ est donnée par la relation ci-dessous:

$$\mathbf{x_{im}} = \mathbf{K}.\begin{bmatrix}\mathbf{R} & \mathbf{T}\end{bmatrix}.\mathbf{X_o} \tag{6.1}$$

En considérant que l'origine des coordonnées de l'objet est le centre de la boîte englobante 3D, les coordonnées de la boîte englobante 3D sont les suivantes $\mathbf{X_1} = \begin{bmatrix} w/2 & h/2 & l/2\end{bmatrix}$, $\mathbf{X_2} = \begin{bmatrix} w/2 & -h/2 & l/2\end{bmatrix}$, ..., $\mathbf{X_8} = \begin{bmatrix} -w/2 & -h/2 & -l/2\end{bmatrix}$. Les coordonnées de la boîte de délimitation 3D dans l'image peuvent ensuite être obtenues à l'aide de l'équation (6.1).

6.3.3 Paramètres prédits

Détection d'objets en 2D. Dans notre travail, nous utilisons YOLOv3 pour effectuer la détection d'objets en 2D. YOLOv3 effectue la prédiction des classes d'objets *cls* ainsi que les paramètres de la boîte englobante **b** (position et dimension). Les prédictions de la boîte englobante sont ensuite utilisées comme RoIs pour l'alignement des caractéristiques du réseau (Feature Align) afin d'extraire les caractéristiques de chaque RoI, qui sont ensuite transmises au reste du réseau pour prédire les paramètres de la boîte englobante 3D.

Estimation de la distance de l'objet. Afin de déterminer le centre de la boîte englobante 3D, il est nécessaire de déterminer sa position sur l'axe Z des coordonnées de la caméra. Pour chaque RoI du détecteur d'objet, nous prédisons la distance du centre de l'objet $\tilde{z_o}$ en mètres.

Prédiction du centre de l'objet. Dans ce travail, nous supposons que le centre de la boîte englobante 3D est le centre 3D de l'objet. Prédire le centre des boîtes englobantes 3D des objets équivaut donc à prédire leur position: $\mathbf{X_o} = [x_o\ y_o\ z_o\ 1]^\top$. Afin d'augmenter la précision de cette prédiction, nous cherchons à prédire le centre 3D projeté sur le plan image. Au lieu de prédire directement les coordonnées du centre de l'objet sur le plan de l'image, nous utilisons les indices de la prédiction de la boîte englobante

2D et nous prédisons la position décalée en pixels du centre de l'objet par rapport au centre de la boîte englobante 2D $\tilde{\mathbf{c}} = \begin{bmatrix} \tilde{cx} & \tilde{cy} \end{bmatrix}$. Nous réduisons donc la variance de la prédiction, ce qui facilite l'apprentissage de l'algorithme. Le centre de l'objet $\mathbf{X_o}$ peut ensuite être calculé en utilisant l'estimation de la distance de l'objet sur l'axe Z et la matrice de calibration inversée $\mathbf{K^{-1}}$ comme décris dans l'équation (6.2):

$$\begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix} = \mathbf{K}^{-1} \begin{bmatrix} \widetilde{cx} \times z_o \\ \widetilde{cy} \times z_o \\ z_o \\ 1 \end{bmatrix} \tag{6.2}$$

Dimensions de l'objet. Au lieu de prédire directement les dimensions de l'objet en mètres, nous utilisons le fait que les dimensions des objets ont une très faible variance au sein d'une même classe (voiture, camion, etc.). Par conséquent, nous choisissons d'utiliser les dimensions moyennes pour chaque classe d'objets comme une forte priorité pour la prédiction des dimensions. **Orientation de l'objet.** Dans notre travail, nous supposons que seul le lacet de l'objet (noté θ) importe pour une application sur l'environnement routier, par conséquent nous ne prédisons pas l'orientation caractérisée par le tangage et le roulis de l'objet et nous fixons le tangage $\phi = 0$ et le roulis $\psi = 0$. Étant donné qu'un même lacet peut conduire à plusieurs orientations observées du point de vue de la caméra (voir la Figure 6.2), nous ne pouvons pas prédire directement l'angle θ. Au lieu de cela, nous prédisons l'angle observé α et récupérons l'orientation globale θ en utilisant l'équation (6.3):

$$\theta = \alpha + arctan(\frac{x_o}{z_o}) \tag{6.3}$$

En suivant les travaux de [85], au lieu de considérer la prédiction de l'angle comme un problème de régression, nous adoptons une approche hybride classification/régression. Nous divisons les angles possibles en 2 intervalles ; nous effectuons ensuite une tâche de classification pour prédire dans quel bac se trouve l'angle de l'objet. Ensuite, nous régressons la différence entre le centre du bin et α.

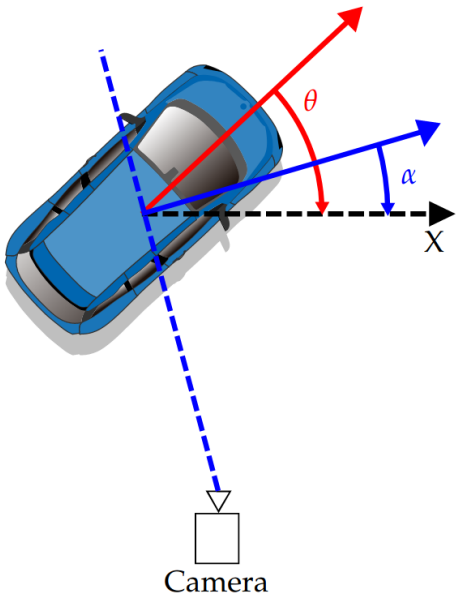

Figure 6.2: Illustration du lacet de l'objet θ et de son orientation observée α. L'orientation observée est obtenue en calculant l'angle entre la normale entre la caméra et le centre de l'objet et l'axe X de la caméra. Étant donné que nous utilisons un repère main gauche, la rotation se fait dans le sens des aiguilles d'une montre. Notre méthode estime l'orientation observée et θ peut être obtenue en utilisant l'équation (6.3).

6.3.4 Fonctions perte

Dans cette sous-section, nous présentons le calcul de la fonction perte de notre méthode. La perte est spécifiée dans l'équation (6.4):

$$L = L_{yolo} + k_1 \times L_{center} + k_2 \times L_{distance} + k_3 \times L_{dim} + k_4 \times L_{orient}. \tag{6.4}$$

Dans nos calculs, nous utilisons la même perte que YOLOv3 pour la détection 2D et la prédiction de classe. Nous utilisons également $k_{i\in[1,...,4]}$ comme facteurs de pondération pour les pertes. Avec $\mathbf{c_k} = \begin{bmatrix} cx^k & cy^k \end{bmatrix}^T$ le centre de vérité de l'objet k en pixels, $\mathbf{Rc_k} = \begin{bmatrix} Rcx_k & Rcy_k \end{bmatrix}^T$ le centre de la RoI pour l'objet k, N le nombre d'objets, et $\tilde{\mathbf{Oc_k}}$ le décalage entre la boite englobante prédite par le détecteur d'objets et la position de l'objet à déterminer, la perte de centre s'écrit dans l'équation (6.5):

$$L_{center} = Moyenne(\frac{1}{N}\sum_{k=1}^{N}\left|\mathbf{c_k} - \mathbf{Rc_k} - \tilde{\mathbf{Oc_k}}\right|). \tag{6.5}$$

En supposant que z_k est la vérité de terrain de la distance d'un objet k sur l'axe Z du repère caméra, N le nombre total d'objets, et $\tilde{z}_k$ la prédiction de distance de notre méthode, la perte de distance est alors calculée en utilisant la perte L1 et est détaillée dans l'équation (6.6):

$$L_{distance} = \frac{1}{N}\sum_{k=1}^{N}|z_k - \tilde{z}_k|. \tag{6.6}$$

Avec $\mathbf{d_k} = \begin{bmatrix} dx & dy & dz \end{bmatrix}^T$ les dimensions réelles d'un objet k en mètres, $\mathbf{dd}_k$ la dimension moyenne pour la classe d'objet k, N le nombre d'objets, et $\tilde{d_k}$ la prédiction de notre méthode, la perte de dimensions est détaillée dans l'équation (6.7):

$$L_{dim} = Moyenne(\frac{1}{N}\sum_{k=1}^{N}\left|\mathbf{d_k} - \tilde{\mathbf{d}}_\mathbf{k} - \mathbf{dd_k}\right|). \tag{6.7}$$

Enfin, en supposant que α_k est l'angle observé de l'objet k, N le nombre total d'objets, et $\tilde{\alpha_k}$ la prédiction, nous utilisons la perte Smooth L1 comme perte d'orientation et la perte s'écrit dans l'équation (6.8):

$$L_{orient} = \frac{1}{N}\sum_{k=1}^{N} e_k, \tag{6.8}$$

où

$$e_k = \begin{cases} 0.5(\alpha_k - \tilde{\alpha_k})^2, & \text{si } |\alpha_k - \tilde{\alpha_k}| < 1 \\ |\alpha_k - \tilde{\alpha_k}| - 0.5, & \text{sinon} \end{cases}$$

6.4 Résultats expérimentaux

6.4.1 Détails de l'entrainement

KITTI. Nous avons entraîné notre méthode sur le jeu de données KITTI dédié à la détection 3D sur le même split d'entraînement que celui utilisé par les auteurs de [85] contenant la moitié des échantillons et nous avons effectué l'évaluation sur l'autre moitié des échantillons. Ce jeu de données offre plus de

7000 images annotées pour l'entraînement avec des données sur les boîtes de délimitation 2D, la position de l'objet (XYZ), les dimensions de l'objet, l'angle de lacet de l'objet, et l'orientation observée pour 3 classes d'objets différentes (voiture, vélo, personne). Nous avons transféré les poids d'un modèle Yolov3 déjà entrainé sur la base de données de détection d'objets 2D COCO [42] (Transfert learning) sur notre nouveau modèle afin de raccourcir le temps d'entrainement mais aussi d'augmenter significativement la précision de notre modèle. Ainsi, les résultats optimaux pour notre méthode ont été atteints après 130 époques.

GTA. Afin de surmonter le manque de bases de données ferroviaires, nous avons utilisé notre nouvelle base de données hybride routière/ferrovaire présentée dans le chapitre précedent pour entrainer et évaluer notre nouvelle méthode. Cette base comporte une vérité de terrain riche permettant l'apprentissage des méthodes de détections d'objets en 3D. Parmi les classes annotées nous avons les voitures, les camions, les piétons et les motos. L'entraînement de notre méthode a été réalisé sur un split contenant $107K$ images routières et ferroviaires. L'évaluation a ensuite été réalisée sur la partie de la base de données contenant $11K$ images. L'entraînement sur ce jeu de données a été effectué sur 50 époques.

Entrainement. Pour estimer la boîte de délimitation 3D, nous avons utilisé un modèle YOLOv3 pré-entraîné sur le jeu de données COCO afin de réduire le temps d'apprentissage. L'apprentissage sur les deux ensembles de données (KITTI et GTAV) a été effectué avec une résolution d'image de 512 pixels avec une taille de lot (batch size) de 64. Nous avons utilisé l'optimiseur ADAM combiné avec un planificateur de taux d'apprentissage cyclique tel que proposé dans [97] pour contrôler le momentum et le taux d'apprentissage pendant l'entraînement. Le taux d'apprentissage maximal optimal a été déterminé à l'aide de la méthode décrite dans le même article [97]. Pour notre méthode, nous avons utilisé un taux d'apprentissage maximal de 7.10^{-4} avec un weight decay de 1.10^{-3}. Par essais et erreurs, nous avons également déterminé les poids de perte et nous avons fixé $k_1 = 1$, $k_2 = 5.1$, $k_3 = 70$, $k_4 = 110$.

6.4.2 Évaluation

Détection 2D. Pour évaluer la performance de la détection 2D, nous avons utilisé les métriques décrites par les auteurs de YOLOv3 sur chaque classe du jeu de données. Étant donné que la régression des paramètres de la boîte de délimitation 3D repose sur des RoIs et des classes précises pour les objets, l'évaluation de la précision du détecteur 2D est une nécessité. Les mesures d'erreur sont la moyenne de la précision (AP), le rappel (R), la précision moyenne (mAP) et le score F1.

Estimation de la distance. Pour l'évaluation de notre estimation de distance, nous avons utilisé la même évaluation que celle utilisée pour les méthodes d'estimation de la profondeur au niveau de l'image comme Monodepth2 [98] ou MadNet [15]. Les métriques utilisées sont les mêmes que celles décrites dans le Chapitre 4. On y retrouve l'erreur relative absolue (Abs Rel), l'erreur relative quadratique (SRE), l'erreur quadratique moyenne (RMSE), la RMSE logarithmique (log RMSE), et le pourcentage de mauvaise correspondance des pixels (BMP). Soient z_{gt} et z_{pd}, respectivement, la distance réelle et prédite de l'objet i, calculée à l'aide de l'équation (6.9), où $\delta = 1.25^k$:

$$\alpha_{k=[1...3]} = max(\frac{z_{gt}}{z_{pd}}, \frac{z_{pd}}{z_{gt}}) < \delta^k. \tag{6.9}$$

Dimensions. L'évaluation de la prédiction de la dimension est effectuée à l'aide du score de dimension (DS) décrit par les auteurs de [76]. Avec V_{pd} et V_{gt} le volume prédit et le volume de vérité terrain de l'objet,

le DS est calculé en utilisant l'équation (6.10):

$$DS = min(\frac{V_{pd}}{V_{gt}}, \frac{V_{gt}}{V_{pd}}). \tag{6.10}$$

Centre de l'objet. Les prédictions du centre de l'objet sont évaluées à l'aide du score du centre (CS) comme décrit dans [76]. En supposant que x et y sont les coordonnées du centre projeté en pixels et w et h la largeur et la hauteur de la boîte de délimitation 2D, CS est calculé avec l'équation (6.11):

$$CS = (2 + cos(\frac{x_{gt} - x_{pd}}{w_{pd}}) + cos(\frac{y_{gt} - y_{pd}}{h_{pd}}))/4. \tag{6.11}$$

Orientation. Pour évaluer les prédictions d'orientation, nous utilisons le score d'orientation (OS) tel que décrit dans le benchmark KITTI. On pose α l'angle observé, L'OS est calculé en utilisant l'équation (6.12):

$$OS = (1 + cos(\alpha_{gt} - \alpha_{pd}))/2. \tag{6.12}$$

6.4.3 Résultats

Les résultats quantitatifs de notre méthode sur les jeux de données KITTI et GTAV sont présentés dans le tableau 6.1 et les Figures 6.3 et 6.4. Nous pouvons constater que, bien que notre méthode offre des résultats inférieurs aux méthodes de l'état de l'art, l'architecture légère de notre réseau nous permet d'effectuer une détection 3D en temps réel, ce qui n'est pas possible avec les autres méthodes de l'état de l'art. Nous avons également ajouté les résultats qualitatifs de notre méthode pour les jeux de données GTAV et KITTI dans la Figure 6.10.

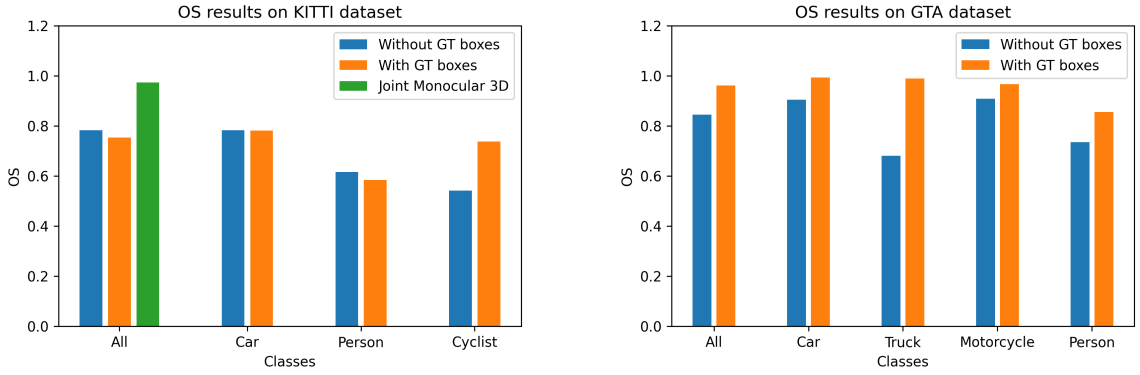

FIGURE 6.3: Dans ces graphiques, nous comparons le score d'orientation obtenu par notre méthode (avec ou sans boîte de vérité de terrain) sur les différentes classes de jeux de données ; nous incluons également les résultats de la méthode "3D joint monocular" [76] (qui utilise également des boîtes de vérité de terrain). Nous pouvons constater que notre méthode a un score d'orientation plus faible lorsque nous n'utilisons pas les boîtes de vérité terrain (GT). Cela peut s'expliquer par le fait que les boîtes utilisées pour l'alignement des caractéristiques (RoI Align) pendant l'inférence sont les mêmes que celles utilisées pendant l'entraînement.

Afin d'effectuer une détection d'objet 3D, notre méthode nécessite 2 étapes:

(1) Le premier niveau permet l'extraction des caractéristiques de l'image pour prédire les RoIs des objets et leurs classes.

(2) Le deuxième niveau de notre réseau CNN utilise à la fois les caractéristiques extraites et les RoIs pour aligner les caractéristiques et prédire les paramètres 3D des objets.

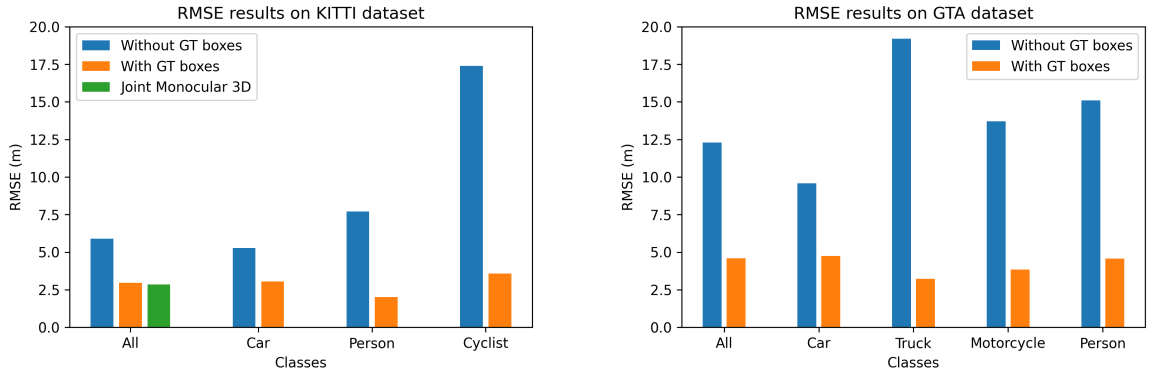

FIGURE 6.4: Dans ces graphiques, nous comparons les RMSE de profondeur obtenus par notre méthode (avec ou sans boîte de vérité au sol) sur les différentes classes de jeux de données ; nous incluons également les résultats de la méthode "3D joint monocular" [76] (qui utilise également des boîtes de vérité au sol). Nous pouvons constater que notre méthode obtient une erreur RMSE plus élevée lorsque nous n'utilisons pas de boîtes de délimitation GT. Cela peut s'expliquer par le fait que les boîtes utilisées pour l'alignement des caractéristiques pendant l'inférence sont les mêmes que celles utilisées pendant l'entraînement. Nous pouvons également constater qu'il y a une perte significative de précision sur les petites classes telles que les bicyclettes ou les personnes lorsque notre méthode prédit les ROIs en utilisant YOLOv3 au lieu d'utiliser la vérité terrain. Cela peut s'expliquer par le fait que les variations dans la prédiction des RoIs ont un impact plus important que pour les classes plus grandes comme les voitures.

Quant à la méthode présentée dans [85], elle présente une approche pour améliorer la prédiction de l'orientation lors de la détection 3D alors que notre approche prédit non seulement l'orientation des objets, mais aussi leur distance, leurs centres 3D et leur dimension pour une détection complète des objets 3D. L'architecture du réseau présenté dans [85] ne comporte qu'un seul étage (le deuxième étage dans notre approche) pour la prédiction 3D et nécessite d'ajouter le premier étage en utilisant un détecteur d'objets 2D (RCNN plus rapide, YOLO, Single-Shot Detector (SSD), etc.). De plus, les résultats présentés dans [85] se concentrent sur l'évaluation de l'orientation sous le jeu de données KITTI et ne comprennent pas d'évaluation dédiée à la détection d'objets 3D. Notre approche prédit non seulement la détection des objets 2D, mais aussi la distance des objets par rapport à la caméra, leur centre 3D, leur orientation et leur dimension. Toutes ces prédictions sont intégrées dans un réseau "tout-en-un" pour prédire les objets 3D. Pour cette raison, nous avons présenté dans le tableau 6.1 les résultats quantitatifs de notre méthode sur les ensembles de données KITTI et GTAV par rapport à [76], et dans le tableau 6.2 quelques résultats expérimentaux pour différentes classes d'objets sur KITTI et GTAV par rapport à [76].

Base	Méthode	Détection 2D				Distance							Dimensions	Centre	Orientation	Utilisation mémoire	Temps d'inférence
		AP	R	mAP	F1	Abs Rel	SRE	RMSE (m)	log RMSE	a_1	a_2	a_3	DS	CS	OS		
KITTI	Yolov3-3d (w GT RoIs)	-	-	-	-	0.096	**0.307**	2.96	0.175	0.941	**0.980**	**0.988**	0.847	**0.983**	0.753	**3.22 GB**	**10ms**
	Yolov3-3d (w/o GT RoIs)	0.469	0.589	0.5	0.517	0.199	2.32	5.89	0.310	0.823	0.934	0.960	0.853	0.951	0.765	**3.22 GB**	**10ms**
	Joint Monocular 3D [76]	-	-	-	-	**0.074**	0.449	**2.847**	**0.126**	**0.954**	**0.980**	0.987	**0.962**	0.918	**0.974**	5.64 GB	97ms
GTA	Yolov3-3d (w GT RoIs)	-	-	-	-	**0.069**	**0.420**	**4.60**	**0.100**	**0.965**	**0.992**	**0.999**	**0.886**	**0.999**	**0.961**	**3.22 GB**	**10ms**
	Yolov3-3d (w/o GT RoIs)	0.533	0.763	0.632	0.61	0.207	4.92	12.3	0.319	0.781	0.897	0.942	0.853	0.846	0.845	**3.22 GB**	**10ms**

Tableau 6.1: Résultats quantitatifs de notre méthode sur les jeux de données KITTI et GTA. L'évaluation a été réalisée sur la partie validée des jeux de données. Comme les mesures de précision pour la détection 2D (ROI) n'étaient pas disponibles pour [76], seules les nôtres sont affichées.

Nous avons mené nos expériences sur les deux jeux de données en utilisant notre méthode avec les RoIs provenant soit de YOLOv3, soit directement de la vérité terrain. Les résultats montrent que la précision de notre méthode, en particulier pour l'estimation de la distance, dépend de la précision de la prédiction de la RoI. Ainsi, nous pouvons constater que notre méthode, lorsque les RoIs de la vérité terrain sont utilisés, a une précision proche des méthodes de l'état de l'art. Cependant, lorsque nous utilisons les ROIs prédites par YOLOv3, la précision est significativement plus faible, ce qui peut être expliqué par le fait

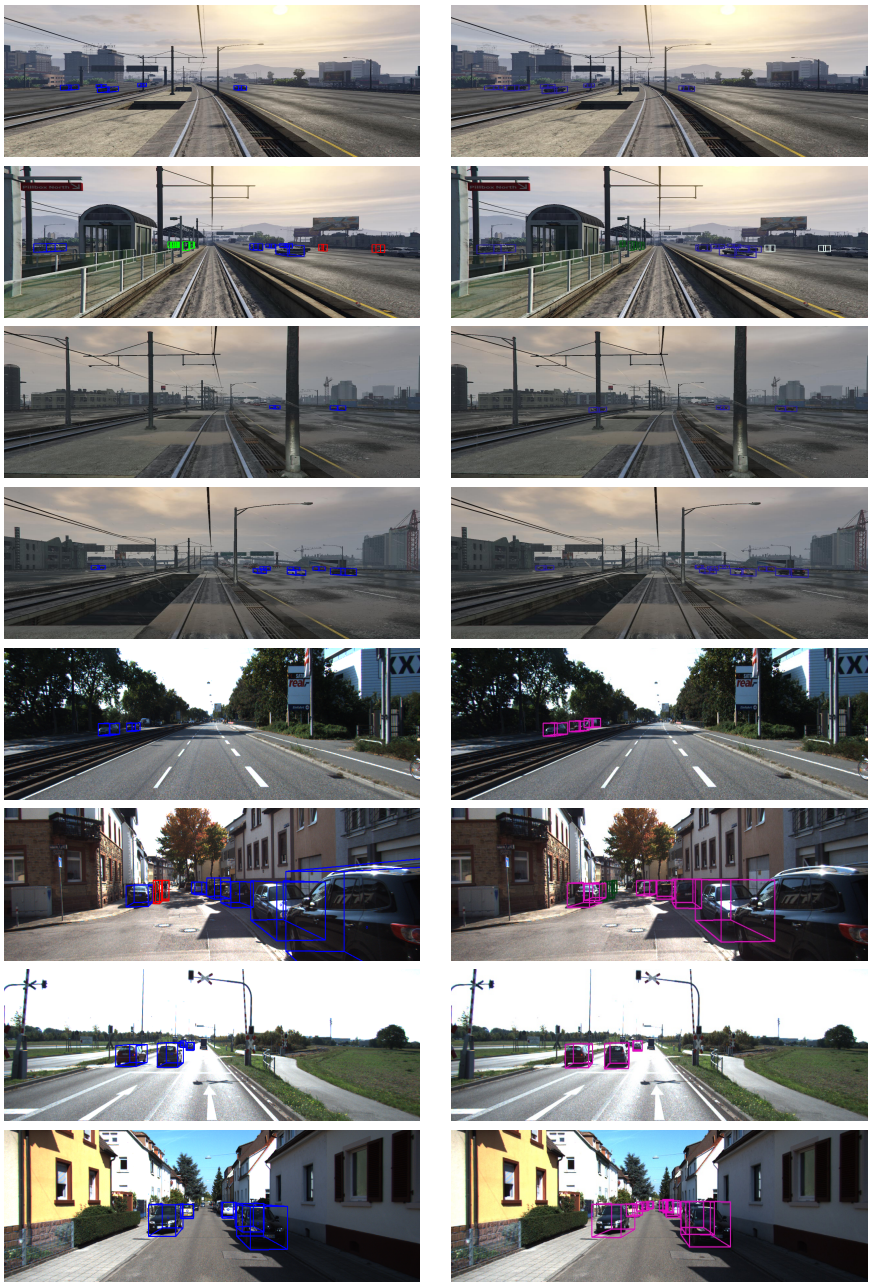

Figure 6.5: Les résultats qualitatifs de notre méthode ont été obtenus grâce aux jeux de données KITTI et GTAV. Ces images ont été extraites du split de validation de chaque jeu de données. Les RoIs utilisés pour prédire les paramètres de la boîte englobante 3D ont été calculés par YOLOv3. 4 lignes du haut: résultats obtenus pour le jeu de données GTAV (colonne de gauche: vérité terrain ; colonne de droite: prédiction), 4 lignes du bas: résultats obtenus pour le jeu de données KITTI (colonne de gauche: vérité terrain ; colonne de droite: prédiction).

que les RoIs pour la même cible peuvent varier en taille et en forme, ce qui rend l'apprentissage de la distance plus difficile. Ce problème est atténué lorsque la méthode est évaluée en utilisant les mêmes RoIs fixes que ceux utilisés pour l'apprentissage, ce qui explique les bonnes performances des méthodes de l'état de l'art et de notre méthode avec les RoIs de la vérité terrain. Nous remarquons également à travers nos résultats que l'OS est relativement faible ; cela peut s'expliquer par le fait que notre méthode peine à distinguer l'avant de l'arrière des objets détectés. Cependant, en traçant les boîtes de délimitation 3D sur l'image, ce problème est atténué. Nous avons déduit, grâce à ces observations, que le problème vient de l'alignement des caractéristiques du réseau lorsque les RoIs varient. En effet, notre module de

Bases	Classes	Méthode	Détection 2D				Distance							Dimensions	Centre	Orientation
			AP	R	mAP	F1	Abs Rel	SRE	RMSE (m)	log RMSE	α_1	α_2	α_3	DS	CS	OS
KITTI	Voiture	Yolov3-3d (w GT RoIs)	-	-	-	-	**0.0957**	0.312	3.05	0.18	**0.941**	0.980	0.988	**0.872**	0.981	0.781
		Yolov3-3d (w/o GT RoIs)	0.558	0.879	0.783	0.682	0.163	1.29	5.28	0.271	0.839	0.948	0.971	0.867	0.962	**0.783**
	Personne	Yolov3-3d (w GT RoIs)	-	-	-	-	0.0979	**0.254**	**2.01**	0.154	0.935	**0.983**	**0.992**	0.705	0.993	0.584
		Yolov3-3d (w/o GT RoIs)	0.508	0.518	0.464	0.513	0.424	7.92	7.70	0.485	0.729	0.844	0.892	0.719	0.885	0.616
	Cycliste	Yolov3-3d (w GT RoIs)	-	-	-	-	0.0979	0.359	3.57	**0.150**	0.940	0.979	0.988	0.809	**0.997**	0.738
		Yolov3-3d (w/o GT RoIs)	0.342	0.371	0.253	0.356	1.04	30.9	17.4	0.797	0.415	0.622	0.719	0.812	0.678	0.542
GTA	Voiture	Yolov3-3d (w GT RoIs)	-	-	-	-	0.0552	0.385	4.74	0.0761	0.990	0.999	0.999	0.860	**0.999**	**0.993**
		Yolov3-3d (w/o GT RoIs)	0.477	0.915	0.778	0.627	0.129	2.28	9.59	0.217	0.873	0.938	0.965	0.812	0.894	0.905
	Camion	Yolov3-3d (w GT RoIs)	-	-	-	-	**0.0454**	**0.178**	**3.22**	**0.0576**	0.994	1.00	1.00	0.871	**0.999**	0.989
		Yolov3-3d (w/o GT RoIs)	0.312	0.880	0.617	0.461	0.259	6.99	19.2	0.455	0.642	0.78	0.868	0.736	0.622	0.681
	Motocyclette	OuYolov3-3drs (w GT RoIs)	-	-	-	-	0.0623	0.266	3.84	0.0753	**1.00**	**1.00**	**1.00**	0.918	1.00	0.967
		Yolov3-3d (w/o GT RoIs)	0.614	0.764	0.725	0.681	0.186	3.58	13.7	0.281	0.691	0.878	0.967	0.901	0.689	0.909
	Personne	Yolov3-3d (w GT RoIs)	-	-	-	-	0.116	0.612	4.56	0.159	0.877	0.966	1.00	**0.963**	0.997	0.856
		Yolov3-3d (w/o GT RoIs)	0.263	0.659	0.488	0.375	0.372	10.5	15.1	0.453	0.620	0.834	0.903	0.961	0.812	0.735

Tableau 6.2: Nos résultats expérimentaux pour différentes classes d'objets sur les jeux de données KITTI et GTA. L'évaluation a été réalisée sur la partie validée des jeux de données. Comme les mesures de précision pour la détection 2D (ROI) n'étaient pas disponibles pour [76], seules les nôtres sont affichées.

prédiction 3D, entraîné sur les RoIs provenant directement de la vérité de terrain, arrive difficilement à prédire avec précision les paramètres 3D lorsque les RoIs utilisés diffèrent de la vérité de terrain.

Notre méthode de détection d'objets 3D, bien qu'étant l'une des plus rapides, n'atteint pas encore le niveau de précision des méthodes de pointe. Cela est dû au fait que la variation des RoIs pendant la phase de prédiction 2D entraîne une perte de précision lors de la prédiction des paramètres 3D. La prédiction des paramètres 3D d'un objet à partir d'une image 2D est un problème complexe, il est donc nécessaire de limiter autant que possible la prédiction directe des paramètres 3D. Par exemple, au lieu de prédire directement la position du centroïde de l'objet en mètres (sur l'axe XYZ), nous prédisons le centre 3D projeté sur l'image 2D, permettant ainsi une meilleure prédiction du centre 3D. Notre réseau peut alors utiliser les informations liées à l'apparence de l'objet pour déduire la position du centre 3D de l'objet sur l'image. En combinant ces informations avec la prédiction de la distance de l'objet par rapport à la caméra et la matrice de calibration de l'objet, nous pouvons obtenir une prédiction du centre 3D de l'objet (voir équation (6.2)) . Cette approche permet d'augmenter la précision de la prédiction 3D de l'objet, mais elle n'est toujours pas aussi précise que l'utilisation des données LiDAR. La précision est également réduite lorsque l'objet est partiellement caché par tout autre obstacle présent dans la scène.

Des évaluations supplémentaires de nos méthodes ont été réalisées pour les différentes classes d'objets présentes sur les deux jeux de données (voir tableau 6.2). Ces résultats montrent que sur les jeux de données KITTI et GTAV, la classe d'objets ayant la plus grande précision est la classe des voitures. Ceci s'explique par le fait que la classe Voiture est la classe majoritaire sur les deux jeux de données et par le fait qu'une voiture représente une cible relativement grande (contrairement à une personne ou un cycliste) rendant la détection plus facile.

Nous avons utilisé YOLOv3 comme première étape car au moment de la conception de notre réseau, YOLOv3 était le détecteur d'objets 2D en temps réel le plus réactif/précis (meilleur que Faster RCNN, SSD, etc.). YOLOv4, qui a été publié en 2020, a apporté des améliorations à l'architecture de base de YOLOv3 (passant de Darknet53 à Darknet53CSP) en raison d'améliorations dans l'augmentation des données pendant le processus d'entraînement. Au cours du développement de notre réseau, nous avons testé les différentes améliorations liées à l'augmentation des données sur notre réseau et, bien qu'elles améliorent la précision de la détection des objets en 2D, elles détériorent et diminuent la qualité de la détection en 3D. Par conséquent, nous avons choisi de ne pas apporter ces améliorations à notre réseau et de conserver YOLOv3 comme approche principale dans la première phase.

Enfin, nous avons mené des expériences sur le temps de calcul et la consommation de mémoire des différents modèles, que l'on peut trouver dans le tableau 6.1. Comme la méthode proposée par [76] est séparée en trois modules (prédiction de la RoI, régression des paramètres 3D et suivi), nous n'avons pu obtenir un temps de calcul théorique qu'en additionnant le temps de calcul de la prédiction de

la RoI et des paramètres 3D. La consommation de mémoire a été obtenue de la même manière. Les résultats de cette expérience montrent que l'architecture à réseau unique de notre méthode nous permet de réduire considérablement le temps de calcul ainsi que l'empreinte mémoire, ce qui convient aux applications temps réel des systèmes embarqués. L'expérience a été menée sur un Nvidia RTX 3080. Notre approche est innovante pour au moins deux raisons importantes. Premièrement, notre nouvelle approche de détection d'objets en 3D est adaptée en temps réel aux conditions réelles de navigation et de trafic. La détection d'objets 3D en temps réel permet d'améliorer la qualité de la perception de l'environnement, ce qui a amélioré la qualité des décisions et des actions (évitement d'obstacles, détection de piétons, maintien d'une distance de sécurité, etc.) Notre approche permet la détection 2D, la prédiction du centre de l'objet, la prédiction de la distance de l'objet par rapport à la caméra, la prédiction du centre de l'objet 3D, la dimension de l'objet 3D et l'orientation de l'objet 3D. Cela signifie que notre méthode est basée sur un nouveau réseau qui peut être entraîné de bout en bout, ce qui facilite le processus d'entraînement. Notre approche est l'une des méthodes les plus rapides et les plus légères par rapport aux méthodes de l'état de l'art. Cependant le prix de cette rapidité d'exécution est une perte en terme de précision assez importante. Il est donc devenu nécessaire de concevoir une seconde méthode pour la détection d'objets en 3D qui ne souffre pas des problèmes liées à la variation des RoIs et de l'alignement des caractéristiques du réseau.

6.5 Détection d'objets en 3D par approche à étage unique

6.5.1 Vue d'ensemble

Afin de résoudre les problèmes liés à l'extraction des caractéristiques dans les RoIs présents dans notre approche précédente, nous avons choisi de développer une nouvelle méthode de détection d'objets 3D qui résoudrait ce problème. Pour ce faire, nous avons supprimé l'étape d'alignement des caractéristiques et le module de régression 3D afin de le remplacer par un réseau à architecture unique. Le réseau va donc, dans le même module, prédire les paramètres 3D et la boîte de délimitation 2D grâce à une architecture à une seule étape basée sur le détecteur 2D Yolov5 [99]. Une illustration de notre nouveau réseau peut être trouvée dans la Figure 6.6.

6.5.2 Détection d'objets en un seul étage

Notre méthode est un détecteur d'objets 3D en une étape basé sur le réseau neuronal de détection 2D YOLOv5. En effet, il s'agit d'un réseau convolutif léger qui offre une bonne précision, permettant une utilisation en temps réel même sur des systèmes embarqués tels que le Nvidia Jetson $TX2$. YOLOv5 offre un gain de performance significatif par rapport à YOLOv3 grâce à une architecture de réseau améliorée ainsi qu'à des innovations sur l'augmentation des données pendant l'entraînement. La conception à une étape de notre réseau permet de meilleurs temps de calcul par rapport aux méthodes à plusieurs étapes et il peut être entraîné de bout en bout. Un autre problème avec les méthodes à plusieurs étapes était la dégradation de la précision de la régression des paramètres 3D lorsque les RoIs prédites utilisées pour l'alignement des caractéristiques varient de celles utilisées dans l'entraînement, comme le montre notre approche précédente. Notre méthode en une étape utilise des boîtes d'ancrage hybrides pour régresser directement la boîte de délimitation 2D ainsi que les paramètres 3D. De cette façon, nous évitons le problème de la fluctuation de la précision lorsque les ROIs ne sont pas fixes. Cette nouvelle approche est la méthode la plus rapide pour la prédiction 3D à partir d'images sans sacrifier la précision. La Figure 6.7

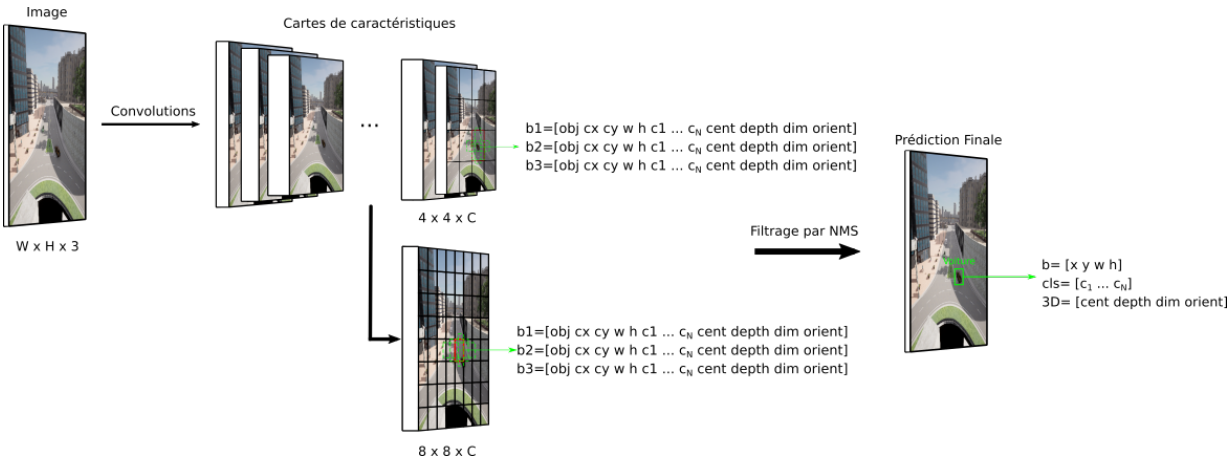

FIGURE 6.6: Architecture de notre nouveau détecteur 3D basé sur l'architecture Yolov5. Nous avons modifié les "boîtes ancrées" pour prédire les paramètres 3D des objets. Une image de dimensions $(W \times H \times 3)$ est utilisée comme entrée pour le réseau. Des couches convolutionnelles successives transforment cette image en cartes de caractéristiques de différentes dimensions. Les cartes de caractéristiques de différentes dimensions (ici 4×4 et 8×8) sont utilisées pour obtenir les "boîtes par défaut" (ou "boîtes ancrées"). Les prédictions des "boîtes par défaut" (à la fois pour la détection 2D, la classification et la détection 3D) pour chacune des cartes de caractéristiques sont combinées et filtrées à l'aide d'un filtre de Non-Maximum-Supression (NMS) pour obtenir la prédiction finale.

montre une vue d'ensemble de notre méthode de détection multi-objets 3D.

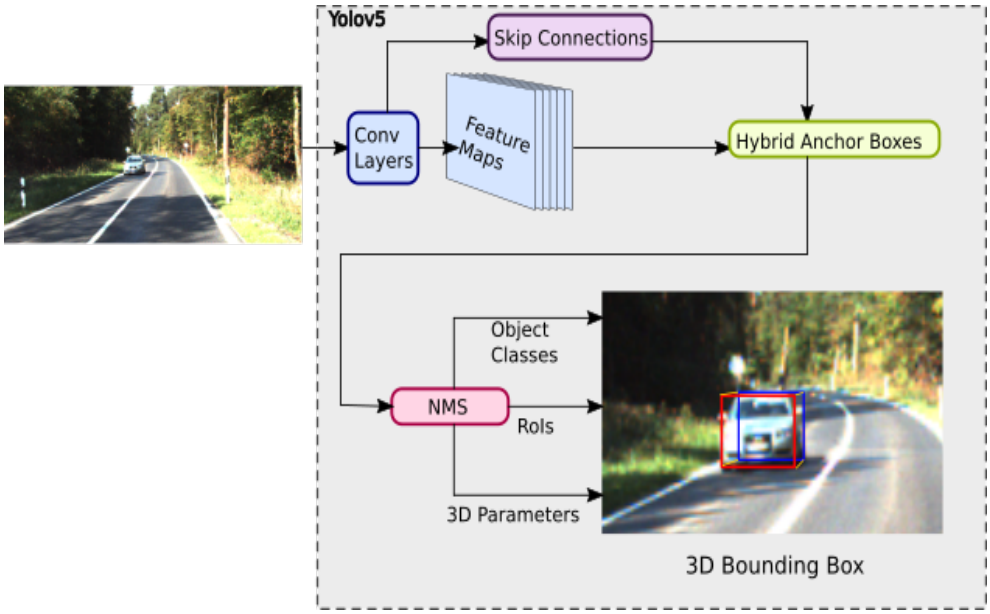

FIGURE 6.7: Vue d'ensemble de notre méthode de détection multi-objets en 3D. Une seule image RVB est utilisée comme entrée. Nous utilisons nos boîtes d'ancrage hybrides pour prédire les boîtes de délimitation 2D et 3D. La Suppression Non-Maximale est utilisée pour filtrer les prédictions. Parmi les paramètres 3D que nous prédisons, nous avons le centre 3D projeté sur le plan de l'image, la distance de l'objet, ses dimensions, et enfin son orientation.

6.5.3 Paramètres predits et fonctions pertes

Les prédiction qui sont renvoyées par cette nouvelle méthode sont exactement les même que celles pour la méthode précédente. On y retrouve la détection d'objets en 2D, la prédiction du centre projeté de

l'objet, l'estimation de distance de l'objet, les dimensions de l'objet et enfin son orientation. Plus de détails sont donnés dans la partie 6.3.3.

La perte utilisée lors de l'apprentissage de notre méthode est détaillée dans l'équation (6.13):

$$L = L_{yolo} + k_1.L_{center} + k_2.L_{distance} + k_3.L_{dim} + k_4.L_{orient}. \tag{6.13}$$

où L_{yolo} est la perte pour la détection 2D et la prédiction de classe, et L_{center}, $L_{distance}$, L_{dim} et L_{orient} sont les pertes pour le centre, la distance, les dimensions et l'orientation de l'objet 3D, respectivement. $k_{i\in[1,etc.,4]}$ sont les poids des différentes pertes. Pour la perte L_{yolo}, nous utilisons la même chose que pour YOLOv5. Les autres fonctions pertes sont identiques à celles utilisées dans la méthode précédente et sont décrites dans la partie 6.3.4.

6.5.4 Boite d'ancrage hybride 2D/3D

La principale contribution de cette nouvelle méthode porte sur l'utilisation de nouvelles boites d'ancrages hybride afin de prédire à la fois les paramètres 2D (Boite englobante, classe) et 3D (Centre, Distance, Dimensions, Orientation). Ces nouvelles boites d'ancrage hybrides proviennent de boites d'ancrage de Yolov5 modifiée. Nous utilisons trois grilles de caractéristiques provenant de notre réseau avec des tailles différentes. Dans chaque grille, trois boîtes d'ancrage sont prédites. Ces boîtes contiennent les prédictions des classes d'objets, la position relative de la boîte englobante par rapport au centre de la grille, la probabilité qu'un objet soit présent, et enfin les paramètres de la boîte englobante 3D (centre projeté, distance, dimensions, orientation). La sortie de nos boîtes d'ancrage hybrides pour notre détecteur d'objets 3D est détaillée dans l'équation (6.14). Soit $\mathbf{y}$ la valeur de sortie de l'une de nos nouvelles boîtes d'ancrage:

$$\mathbf{y} = \begin{bmatrix} \mathbf{b_{off}} & obj & \mathbf{cls} & \mathbf{c_{off}} & Z & \mathbf{D} & \mathbf{O} \end{bmatrix} \tag{6.14}$$

Avec $\mathbf{b_{off}} = \begin{bmatrix} x & y & w & h \end{bmatrix}$ la prédiction du décalage de la boîte englobante 2D (avec x et y la position du centre de la boite sur les 2 axes de l'image et w et h la hauteur et la largeur de la boite en pixels. obj représente la prédiction de la présence de l'objet dans la boite, $\mathbf{cls} = \begin{bmatrix} cls_1 & etc. & cls_N \end{bmatrix}$ le score pour les N classes cls, $\mathbf{c_{off}} = \begin{bmatrix} x_c & y_c \end{bmatrix}$ la prédiction du décalage entre le centre 3D projeté et le centre de la boîte englobante en pixels, Z est la position sur l'axe Z du centre 3D de l'objet en mètres, $\mathbf{D} = \begin{bmatrix} W_1 & H_1 & L_1 & etc. & W_N & H_N & L_N \end{bmatrix}$ est le décalage entre les dimensions moyennes d'un objet selon sa classe N et la dimension de l'objet réelle prédite en mètres. Enfin, nous avons la prédiction d'orientation $\mathbf{O} = \begin{bmatrix} bin_1 & \overline{bin_1} & sin_1 & cos_1 & bin_2 & \overline{bin_2} & sin_2 & cos_2 \end{bmatrix}$, où bin_1 et bin_2 représentent la probabilité prédite que $\alpha \in \begin{bmatrix} -195 & 15 \end{bmatrix}$ et $\alpha \in \begin{bmatrix} -15 & 105 \end{bmatrix}$ respectivement avec α l'orientation observée de l'objet. $\overline{bin_k}$ représente la probabilité que l'angle ne se situe pas dans l'intervalle k. Enfin $sin_{i\in\{1,2\}}$ $cos_{i\in\{1,2\}}$ sont le sinus et le cosinus du décalage de l'angle observé par rapport au centre du i-ème bac. Une illustration des boites d'ancrages hybrides peut être retrouvé dans la Figure 6.6.

6.5.5 Détails de l'entrainement

Taille du modèle

Comme proposé dans le code original YOLOv5, notre code peut être décliné en différentes versions avec une profondeur (nombre de couches dans le réseau) et une largeur (nombre de filtres dans chaque couche) différentes afin de faire varier le nombre de paramètres et donc la taille du réseau. Un réseau

avec un plus grand nombre de paramètres offrira une meilleure précision, mais au prix d'une forte augmentation du temps de calcul et de la consommation de mémoire. Afin de tester l'adaptabilité de notre méthode en fonction du domaine d'application (temps réel sur un système embarqué ou utilisation sur un PC puissant) nous avons choisi d'entraîner des modèles à différentes tailles. Ainsi, notre approche est déclinée en 3 versions: Large, Medium et Small. Afin d'analyser les performances en termes de précision de détection 3D et de temps de calcul de chaque version, nous avons entraîné et évalué chacune d'entre elles sur les jeux de données KITTI et GTA.

Modèle à attention partagée

Parmi les blocs convolutifs composant notre réseau, le goulot d'étranglement (*Bottleneck*) introduit dans [100] est largement utilisé dans YOLOv5. Ce bloc est constitué de plusieurs couches convolutives organisées en blocs et qui vont progressivement réduire les dimensions du volume d'entrée. Des améliorations de ces blocs convolutifs ont été récemment introduites dans [101] avec les convolutions à attention partagée (*Split Attention*). Ces nouveaux blocs convolutifs à attention partagée augmentent les performances des réseaux au prix d'une légère augmentation du temps de calcul. Pour créer un modèle offrant un bon compromis entre précision et temps de calcul, nous avons modifié le modèle "Small" en remplaçant les goulots d'étranglement par des goulots d'étranglement d'attention partagée. Ce nouveau modèle, appelé Small SA (Split Attention), est dédié aux applications sur des systèmes aux ressources de calcul limitées, comme les cartes embarquées Jetson $TX2$, sans sacrifier la précision des prédictions.

Entraînement sur KITTI

Nous avons entraîné notre méthode sur le jeu de données KITTI dédié à la détection 3D sur la même répartition d'entraînement et de validation définie dans [102]. Le segment d'apprentissage contient 3712 images et le segment de validation 3769 images. Pour accélérer le processus d'apprentissage et améliorer la précision, nous avons entraîné nos modèles avec les poids pré-entraînés de YOLOv5 ou les poids pré-entraînés de notre modèle après un entraînement de 15 époques sur notre jeu de données GTAV. Nous démontrons que le pré-entraînement de notre modèle sur notre jeu de données améliore significativement la précision de notre méthode sur le jeu de données KITTI. De plus, comme dans [92], nous effectuons l'entraînement avec les images de gauche et de droite du split d'entraînement de KITTI, doublant ainsi le nombre d'images disponibles pour l'entraînement, ce qui améliore les performances de notre méthode lors de l'évaluation. Pour rendre notre méthode compatible avec les cartes embarquées telles que la Nvidia Jetson $TX2$, nous avons effectué l'entraînement avec une résolution réduite de 672×224. Nous avons également entraîné un modèle avec une résolution de 1312×416, plus proche de la résolution originale, afin de comparer la précision de notre méthode avec les méthodes de pointe sur le jeu de données KITTI.

Entraînement sur GTA

L'apprentissage de notre méthode a été effectué sur un échantillon contenant des images routières et ferroviaires (107 000 images au total). L'évaluation a ensuite été réalisée sur la partie validation du jeu de données contenant 11630 d'images. L'apprentissage sur ce jeu de données a été effectué sur des époques de 50 avec les couches de base de notre réseau provenant d'un modèle pré-entraîné de Yolov5 sur la base de données de détection de COCO 2D [42] afin de réduire le temps d'apprentissage et d'améliorer la précision.

Augmentation des données

L'augmentation des données est très utile pour améliorer les performances des modèles CNN via la création de nouveaux échantillons d'images en appliquant des transformations sur les images déjà présentes. Même si KITTI et notre jeu de données GTA présentent une variété d'environnements et de conditions (dans le jeu de données GTA, par exemple, nous avons différentes situations comme une route par temps clair, une route de nuit, une route par temps de pluie et une voie ferrée par temps clair, comme le montre la Figure 5.2), le risque de sur-entraînement de nos modèles est toujours présent. L'augmentation des données offre de multiples avantages:

(1) Réduire le sur-entraînement

(2) augmenter les capacités de généralisation des modèles entraînés, ce qui permet d'obtenir de meilleurs résultats dans un environnement réel. C'est un bon moyen d'introduire de la "non-linéarité" dans le modèle de l'ensemble de données, le rendant plus proche du monde réel.

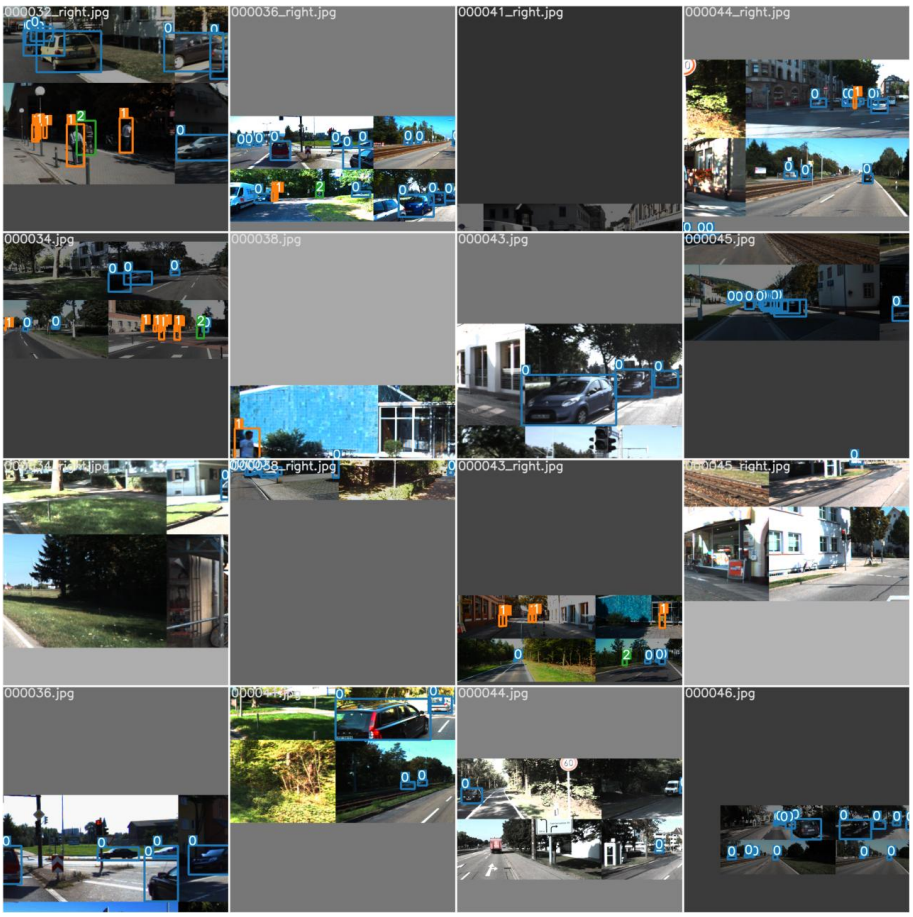

FIGURE 6.8: Augmentation des données utilisées lors de l'entraînement de nos modèles.

Nous utilisons l'augmentation des données en mosaïque telle que définie dans YOLOv4 [103] qui consiste à mélanger 4 images d'entraînement pour améliorer la compréhension du contexte spatial des objets pour notre méthode. Nous avons également appliqué des transformations de translation, de teinte, de saturation, de contraste ainsi que de mise à l'échelle aux images pour améliorer encore la compréhension. Un exemple des images augmentées sont présentées dans la Figure 6.8.

Hyperparamètres

Les hyperparamètres de nos modèles déterminés après de nombreuses itérations de courtes sessions d'entraînement. Nous utilisons l'optimiseur Adam avec un planificateur de taux d'apprentissage cyclique tel que décrit dans [97]. Le taux d'apprentissage maximal a été fixé à $9,4e-4$ avec un taux d'apprentissage final de $1,8e-5$. Enfin, nous utilisons l'accumulation de gradient pour effectuer l'apprentissage avec une taille de lot effective de 64.

Pondération de la fonction de perte

Comme notre méthode est entraînée de bout en bout, notre fonction de perte combine la perte de chaque tâche. Afin d'obtenir des résultats optimaux, il est nécessaire de trouver le poids $(k1, etc., k4)$ de chaque perte. Grâce à de multiples essais, nous avons trouvé $k1 = 0.005$, $k2 = 0.2$, $k3 = 0.0176$, $k4 = 0.01$. Afin d'éviter le sur-entraînement pour la détection 2D, nous avons imposé la condition $L_{yolo} < 0,1$ qui, lorsqu'elle est satisfaite, L_{yolo} est ignorée dans la boucle d'optimisation.

6.5.6 Évaluation

Dans cette sous-section, nous présentons les métriques qui seront utilisées dans l'évaluation de nos modèles.

Précision moyenne 3D

Pour l'évaluation sur le jeu de données KITTI, nous avons utilisé les métriques officielles du benchmark telles que la précision moyenne (AP) 3D avec 40 positions de rappel telles que présentées par [104]. Le benchmark officiel KITTI utilise un seuil de reconnaissance de 0,7 pour la classe voiture et un seuil de 0,5 pour les classes personne et vélo. Nous avons également ajouté des résultats d'évaluation pour la classe des voitures avec un seuil de 0,5.

Détection 2D

Pour évaluer la performance du détecteur 2D, nous avons utilisé les métriques décrites par les auteurs de YOLOv5 sur chaque classe du jeu de données. Puisque la régression de la métrique de la boîte englobante 3D repose sur une RoI et des classes précis pour les objets, l'évaluation de la précision du détecteur 2D est une nécessité. Les mesures d'erreur sont la précision moyenne (AP), le rappel (R) et la précision moyenne (mAP_{50}).

Estimation de la distance

Pour l'évaluation de notre estimateur de distance, nous avons utilisé la même évaluation que celle utilisée pour les méthodes d'estimation de la profondeur au niveau de l'image. Les métriques utilisées sont l'erreur relative absolue (Abs Rel), l'erreur relative au carré (SRE), l'erreur quadratique moyenne (RMSE), la logarithmique RMSE (log RMSE) et le pourcentage de pixels non concordants. Soit z_{gt} et z_{pd} la distance réelle et prédite de l'objet i respectivement, elle est calculée en utilisant l'équation (6.15), où $\delta = 1.25^k$:

$$\alpha_{k=[1..3]} = max(\frac{z_{gt}}{z_{pd}}, \frac{z_{pd}}{z_{gt}}) < \delta^k. \tag{6.15}$$

Dimensions

L'évaluation de la prédiction de la dimension est effectuée à l'aide du score de dimension (DS) décrit par les auteurs de [76]. Avec V_{pd} et V_{gt} le volume prédit et le volume de vérité terrain de l'objet, le DS est calculé à l'aide de l'équation (6.16) en faisant la moyenne sur tous les exemples:

$$DS = min(\frac{V_{pd}}{V_{gt}}, \frac{V_{gt}}{V_{pd}}). \tag{6.16}$$

Centre de l'objet

Les prédictions du centre de l'objet ont été évaluées à l'aide du Center Score (CS) tel que décrit dans [76]. En supposant que x et y sont les coordonnées du centre projeté en pixels et w et h la largeur et la hauteur de la boîte de délimitation 2D, le CS est calculé avec l'équation (6.17):

$$CS = (2 + cos(\frac{x_{gt} - x_{pd}}{w_{pd}}) + cos(\frac{y_{gt} - y_{pd}}{h_{pd}}))/4. \tag{6.17}$$

Orientation

Pour évaluer les prédictions d'orientation, nous utilisons le score d'orientation (OS), moyenné sur tous les exemples. L'OS est calculé à l'aide de l'équation (6.18):

$$OS = (1 + cos(\alpha_{gt} - \alpha_{pd}))/2. \tag{6.18}$$

Temps de calcul

Pour évaluer le temps de calcul de nos modèles, nous calculons le temps moyen nécessaire pour effectuer une prédiction (le filtrage par suppression non maximale (NMS) est inclus) pour une seule image pendant l'inférence. Nous avons testé toutes les méthodes présentées sur un GPU Nvidia RTX à 3080. Nous avons également inclus le temps de calcul sur une carte Nvidia Jetson $TX2$ embarquée comme indiqué dans le tableau 6.6.

Méthode	Pré-entrainement sur GTA	Résolution	[val] IOU 0.7			[val] IOU 0.5			Temps/img (ms)	Consommation mémoire
			Easy	Mod	Hard	Easy	Mod	Hard		
Mono3D [84]		-	2.53	2.31	2.31	25.19	18.20	15.52	-	-
Multi-Fusion [88]		1696x512	10.53	5.69	5.39	47.88	29.48	26.44	-	-
M3D-RPN [91]		1696x512	20.27	17.06	15.21	48.96	39.57	33.01	78.3	5 GB
GAM3D ¯\cite{liu2021ground}		1280x288	**23.63**	16.16	12.06	60.92	42.18	32.02	34.2	2.1 GB
Yolov5-3d (small)		672x224	7.29	5.48	5.34	44.35	31.80	29.91	1.5	1.6 GB
		1312x416	15.52	13.27	13.19	48.24	36.14	31.78	4.4	1.7 GB
	x	1312x416	15.59	13.50	13.14	49.71	38.02	32.70	4.4	1.7 GB
Yolov5-3d (small SA)		672x224	13.79	11.72	11.39	44.51	32.47	27.26	3.1	**1.5 GB**
		1312x416	10.34	10.07	9.85	23.22	20.22	19.56	6.1	1.6 GB
	x	1312x416	15.57	13.32	13.39	51.33	38.09	33.31	6.1	1.6 GB
Yolov5-3d (medium)		672x224	9.42	7.96	6.52	46.22	34.66	33.01	2.7	1.7 GB
		1312x416	17.96	14.27	13.73	51.36	38.28	37.41	8.2	1.9 GB
	x	1312x416	17.63	15.04	14.68	53.95	44.73	39.86	8.2	1.9 GB
Yolov5-3d (large)		672x224	12.52	10.52	9.75	53.27	40.58	35.34	4	2 GB
		1312x416	21.07	16.07	15.68	55.69	41.89	40.46	11.2	2.15 GB
	x	1312x416	23.26	**18.46**	**16.16**	**60.93**	**48.45**	**42.72**	11.2	2.15 GB

Tableau 6.3: Résultats quantitatifs pour la classe Voiture de notre évaluation sur le jeu de données KITTI. Nous pouvons voir que notre méthode bénéficie d'un pré-entraînement sur notre jeu de données GTA et parvient à surpasser les autres méthodes de pointe, en particulier pour les cibles modérées et difficiles. La classe voiture dans le jeu de données KITTI représente ~ 82% de tous les objets du jeu de données.

Méthode	Pré-entrainement sur GTA	Résolution	[val] IOU 0.5 Easy	Mod	Hard	Temps/img (ms)	Consommation mémoire	# Paramètres
Yolov5-3d (small)		672x224	5.30	4.53	4.59	**1.5**	1.6 GB	
		1312x416	4.66	4.23	3.95	4.4	1.7 GB	7.3M
	x	1312x416	6.91	5.45	5.03	4.4	1.7 GB	
Yolov5-3d (small SA)		672x224	9.96	9.09	9.09	3.1	**1.5 GB**	
		1312x416	4.55	4.55	4.55	6.1	1.6 GB	9.7M
	x	1312x416	11.75	10.85	10.56	6.1	1.6 GB	
Yolov5-3d (medium)		672x224	6.51	6.18	5.90	2.7	1.7 GB	
		1312x416	7.94	5.93	5.44	8.2	1.9 GB	21.6M
	x	1312x416	8.66	6.19	6.01	8.2	1.9 GB	
Yolov5-3d (large)		672x224	**11.76**	**11.36**	**10.62**	4	2 GB	
		1312x416	8.55	7.00	6.77	11.2	2.15 GB	47.5M
	x	1312x416	11.58	8.59	7.99	11.2	2.15 GB	

Tableau 6.4: Résultats quantitatifs pour la classe Personne de notre évaluation sur le jeu de données KITTI. Nous pouvons constater que notre méthode bénéficie d'un pré-entraînement sur notre jeu de données GTA. La classe Personne du jeu de données KITTI représente ~ 13% de tous les objets du jeu de données.

Méthode	Pré-entrainement sur GTA	Résolution	[val] IOU 0.5 Easy	Mod	Hard	Temps/img (ms)	Consommation Mémoire	# Parametres
Yolov5-3d (small)		672x224	3.68	3.12	2.27	**1.5**	1.6 GB	
		1312x416	4.66	4.23	3.95	4.4	1.7 GB	7.3M
	x	1312x416	14.56	11.83	11.64	4.4	1.7 GB	
Yolov5-3d (small SA)		672x224	4.39	2.65	2.20	3.1	**1.5 GB**	
		1312x416	0.98	0.88	0.82	6.1	1.6 GB	9.7M
	x	1312x416	9.39	6.39	6.53	6.1	1.6 GB	
Yolov5-3d (medium)		672x224	13.34	10.37	10.39	2.7	1.7 GB	
		1312x416	10.75	7.65	7.53	8.2	1.9 GB	21.6M
	x	1312x416	18.12	**13.58**	**12.02**	8.2	1.9 GB	
Yolov5-3d (large)		672x224	12.88	10.26	10.31	4	2 GB	
		1312x416	**14.88**	10.33	8.21	11.2	2.15 GB	47.5M
	x	1312x416	11.08	5.94	5.71	11.2	2.15 GB	

Tableau 6.5: Résultats quantitatifs pour la classe Cycle de notre évaluation sur le jeu de données KITTI. Nous pouvons voir que notre méthode bénéficie d'un pré-entraînement sur notre jeu de données GTA. La classe Cycle du jeu de données KITTI représente ~ 5% de tous les objets du jeu de données.

Methode	Résolution	Temps/img (ms)	# Paramètres
Yolov5-3d (small)	672x224	70	7.3 M
	1312x416	130	
Yolov5-3d (small SA)	672x224	**60**	9.7M
	1312x416	200	
Yolov5-3d (medium)	672x224	143	21.6M
	1312x416	359	
Yolov5-3d (large)	672x224	199	47.5M
	1312x416	613	

Tableau 6.6: Temps de calcul sur la carte embarquée Nvidia Jetson $TX2$. Nos résultats montrent que notre nouveau modèle Small SA est le plus adapté aux applications embarquées en temps réel avec $60 ms/img$ (17 fps).

6.5.7 Analyse des résultats

Nous présentons les résultats quantitatifs de nos modèles sur le jeu de données KITTI en utilisant la métrique officielle 3D AP dans le Tableau 6.3 pour la classe Voiture, le Tableau 6.4 pour la classe Personne, et le Tableau 6.5 pour la classe Bicyclette. Nous présentons nos résultats ainsi que les méthodes les plus récentes à des fins de comparaison. Ces résultats montrent que notre approche, bien que moins complexe que les méthodes de l'état de l'art, a une précision comparable ou même supérieure à celles-ci. Notre modèle est également beaucoup plus rapide que toutes les autres approches testées, avec un temps de calcul de 11,2ms par image sur notre modèle Large lors de l'inférence sur un GPU RTX à

3080. Notre méthode est même capable d'atteindre une plus grande précision en utilisant des poids pré-entraînés sur notre jeu de données GTAV, montrant le potentiel des jeux de données provenant de jeux vidéos pour améliorer les méthodes de détection d'objets 3D sur des jeux de données réels. Ainsi, notre modèle "Large" peut surpasser les méthodes de pointe dans des scénarios modérés et difficiles. Enfin, nous avons également mené des expériences sur une carte embarquée Jetson $TX2$ pour confirmer l'applicabilité de notre petit modèle aux tâches en temps réel pour les applications embarquées. Nous présentons les résultats dans le tableau 6.6.

Les résultats de notre nouveau modèle SA Small montrent que ses performances sont les mêmes, voire pires lorsqu'il n'utilise pas les poids pré-entraînés, que celles du modèle Small lors de l'apprentissage à pleine résolution (1312×416). Cependant, lorsqu'il utilise une résolution plus petite (672×224), il surpasse significativement le modèle Small. Il s'agit d'un modèle particulièrement intéressant pour les applications sur des systèmes où les capacités de calcul sont limitées, car ses besoins en mémoire sont moindres et il fonctionne à $3,1 ms/img$ tout en ayant une précision supérieure à celle des modèles Small et Medium entraînés à la même résolution. Nos tests avec le Jetson $TX2$ montrent également qu'il est le plus rapide de nos méthodes sur cette plateforme. Nous pouvons constater que nous avons obtenu des gains de précision significatifs en passant d'une approche multi-étapes basée sur YOLOv3 [14] à une approche basée sur un seul étage en utilisant notre version de YOLOv5. Nous avons mené les expériences en utilisant soit les ROIs des prédictions YOLOv3, soit la vérité terrain pour l'alignement des caractéristiques nécessaires à la régression des paramètres de la boîte de délimitation 3D. Ceci afin de mettre en évidence le problème posé par la variation des ROIs dans la régression des paramètres 3D dans les approches multi-étapes. Nous montrons que notre nouvelle approche en une étape est nettement plus performante que notre méthode précédente dans les deux cas, en particulier pour l'estimation de la profondeur et de l'orientation.

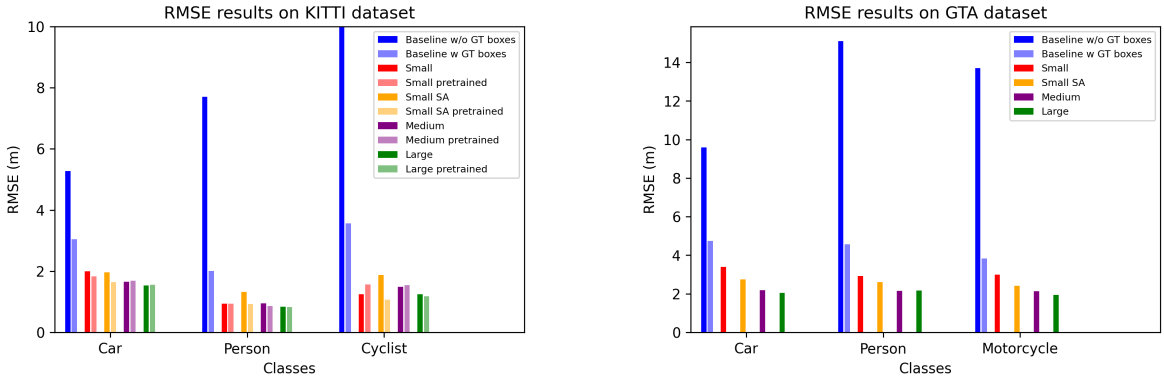

Figure 6.9: RMSE de la profondeur obtenue par nos différents modèles par rapport à notre précédente méthode multi-étapes basée sur Yolov3 [11] pour chaque classe sur les jeux de données KITTI et GTA. Notre nouvelle méthode améliore la précision sur le jeu de données KITTI de manière significative lorsque nous utilisons des poids pré-entraînés sur notre nouveau jeu de données GTA. Avec ces poids, notre nouveau modèle avec convolutions d'attention partagée (SA) est capable d'atteindre les mêmes performances que notre modèle de taille moyenne.

Nous présentons également les différents modèles de notre approche actuelle (Small, SA Small, Medium, Large) et nous pouvons voir que la précision entre les différents modèles pour la détection 2D, les dimensions, le centre et l'orientation ne bénéficie pas significativement des modèles plus lourds alors que l'estimation de la distance s'améliore en utilisant des modèles plus grands. Les résultats qualitatifs sur les jeux de données KITTI et GTA sont présentés dans la Figure 6.10.

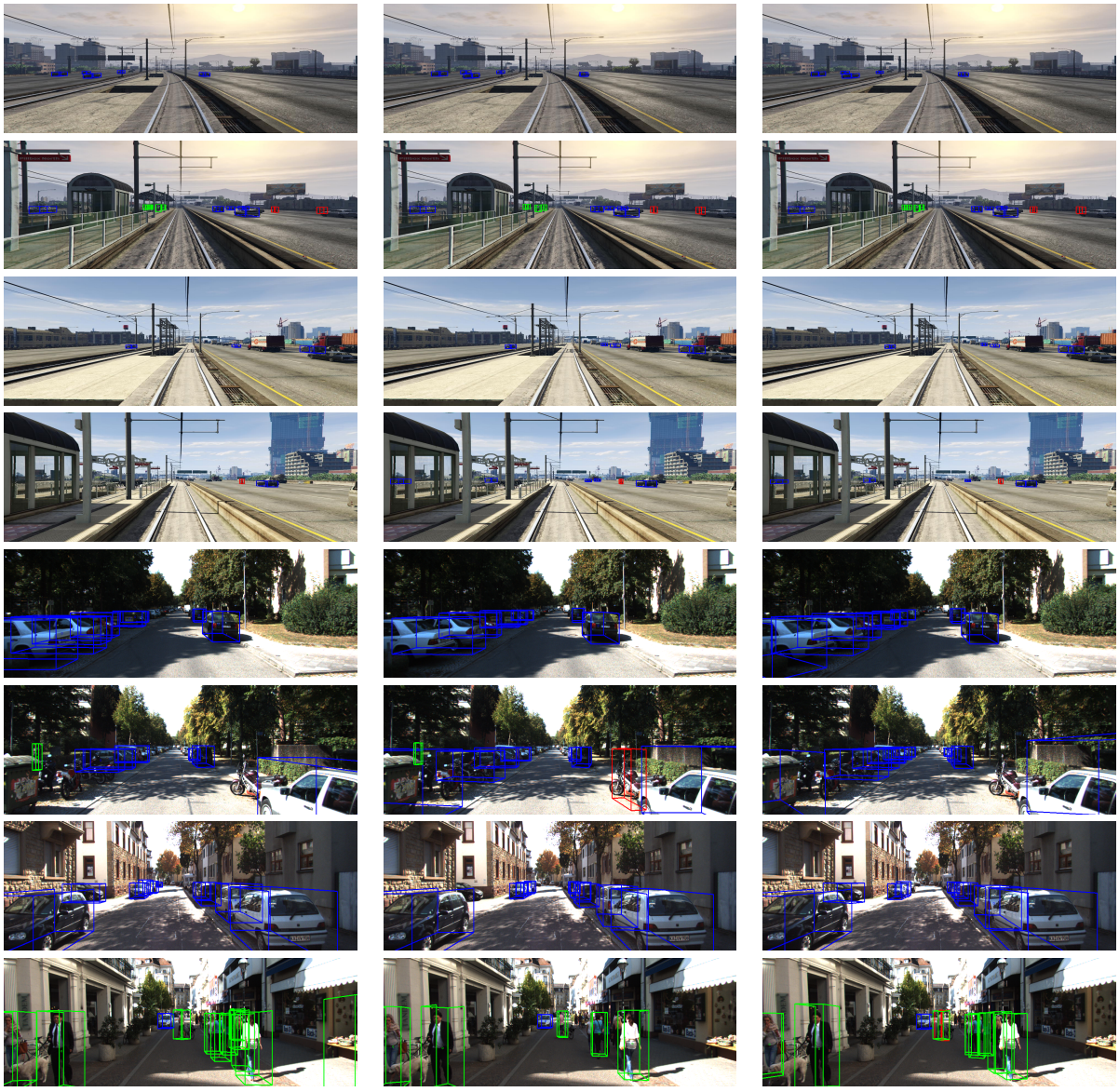

Figure 6.10: Résultats qualitatifs de nos méthodes sur les jeux de données KITTI et GTA. Ces images sont extraites du split de validation de chaque jeu de données. Nous affichons les résultats de notre meilleur modèle (large pré-entraîné sur GTAV) et de notre nouveau modèle léger pour les plateformes embarquées (Small SA pré-entraîné sur GTAV). 4 lignes du haut: résultats obtenus sur le jeu de données GTAV (colonne de gauche: vérité terrain, colonne du milieu: Small SA, colonne de droite: Large), 4 lignes du bas: résultats obtenus sur le jeu de données KITTI (colonne de gauche: vérité terrain, colonne du milieu: Small SA, colonne de droite: Large).

Bases	Classes	Méthode	Détection 2D			Distance							Dimensions	Centre	Orientation
			AP	R	mAP@0.5	Abs Rel	SRE	RMSE (m)	log RMSE	α_1	α_2	α_3	DS	CS	OS
KITTI	Voiture	Yolov3-3d (w GT RoIs)	-	-	-	0.096	0.312	3.05	0.1800	0.941	0.980	0.988	0.872	**0.981**	0.781
		Yolov3-3d (w/o GT RoIs)	0.558	**0.879**	0.783	0.163	1.290	5.28	0.2710	0.839	0.948	0.971	0.867	0.962	0.783
		Yolov5-3d (small)	0.886	0.814	0.790	0.051	0.109	2.00	0.0782	0.987	0.997	**0.999**	0.879	0.960	0.907
		Yolov5-3d (small + GTA)	0.882	0.818	**0.796**	0.050	0.099	1.83	0.0757	0.988	0.997	**0.999**	0.882	0.961	0.935
		Yolov5-3d (small SA)	0.880	0.764	0.739	0.062	0.133	1.97	0.0901	0.981	0.997	**0.999**	0.874	0.956	0.859
		Yolov5-3d (small SA + GTA)	0.901	0.777	0.755	0.047	0.086	1.65	0.0733	0.991	**0.998**	**0.999**	0.883	0.960	0.908
		Yolov5-3d (medium)	0.892	0.800	0.777	0.047	0.085	1.66	0.0755	0.988	0.997	**0.999**	0.883	0.963	0.929
		Yolov5-3d (medium + GTA)	0.887	0.822	0.798	0.045	0.083	1.69	0.0704	0.989	**0.998**	**0.999**	0.883	0.963	0.938
		Yolov5-3d (large)	**0.908**	0.787	0.787	0.042	0.072	**1.53**	0.0670	**0.992**	**0.998**	**0.999**	**0.885**	0.964	**0.947**
		Yolov5-3d (large + GTA)	0.903	0.798	0.777	**0.041**	**0.070**	1.56	**0.0657**	**0.992**	**0.998**	**0.999**	**0.885**	0.964	0.946
	Personne	Yolov3-3d (w GT RoIs)	-	-	-	0.098	0.254	2.01	0.1540	0.935	0.983	0.992	0.705	**0.993**	0.584
		Yolov3-3d (w/o GT RoIs)	0.508	0.518	0.464	0.424	7.920	7.70	0.4850	0.729	0.844	0.892	0.719	0.885	0.616
		Yolov5-3d (small)	0.817	0.528	0.514	0.057	0.068	0.94	0.0800	0.979	0.998	**1**	0.729	0.949	0.750
		Yolov5-3d (small + GTA)	0.740	0.554	0.525	0.054	0.062	0.94	0.0762	0.980	0.999	**1**	0.736	0.948	0.745
		Yolov5-3d (small SA)	0.773	0.435	0.419	0.086	0.136	1.33	0.1050	0.964	0.999	**1**	0.682	0.945	0.581
		Yolov5-3d (small SA + GTA)	0.821	0.436	0.427	0.059	0.067	0.92	0.0803	0.981	0.999	**1**	0.723	0.944	0.716
		Yolov5-3d (medium)	0.828	0.527	**0.527**	0.049	0.065	0.95	0.0745	0.981	0.996	**1**	0.709	0.949	0.815
		Yolov5-3d (medium + GTA)	0.825	0.532	0.517	0.051	0.055	0.86	0.0731	0.982	**1**	**1**	0.733	0.950	**0.818**
		Yolov5-3d (large)	**0.851**	0.514	0.501	0.048	0.050	0.83	0.0689	0.987	0.999	**1**	0.722	0.951	0.812
		Yolov5-3d (large + GTA)	0.817	**0.539**	0.519	**0.046**	**0.049**	**0.82**	**0.0667**	**0.993**	**1**	**1**	**0.736**	0.950	0.772
	Cycliste	Yolov3-3d (w GT RoIs)	-	-	-	0.098	0.359	3.57	0.1500	0.940	0.979	0.988	0.809	**0.997**	0.738
		Yolov3-3d (w/o GT RoIs)	0.342	0.371	0.253	1.040	30.900	17.40	0.7970	0.415	0.622	0.719	0.812	0.678	0.542
		Yolov5-3d (small)	0.568	0.400	0.342	0.058	0.075	1.25	0.0755	0.989	**1**	**1**	0.787	0.960	0.851
		Yolov5-3d (small + GTA)	0.589	0.403	0.364	0.056	0.099	1.57	0.0780	0.986	0.997	**1**	**0.827**	0.964	0.857
		Yolov5-3d (small SA)	0.490	0.274	0.215	0.099	0.205	1.88	0.1140	0.959	**1**	**1**	0.800	0.952	0.868
		Yolov5-3d (small SA + GTA)	0.655	0.348	0.321	0.053	0.072	1.07	0.0746	0.987	0.997	**1**	0.819	0.961	0.919
		Yolov5-3d (medium)	0.692	0.476	0.436	0.051	0.083	1.49	0.0664	0.995	**1**	**1**	0.824	0.962	0.865
		Yolov5-3d (medium + GTA)	0.654	**0.492**	**0.460**	0.049	0.083	1.54	0.0665	0.995	**1**	**1**	0.822	0.961	0.888
		Yolov5-3d (large)	**0.780**	0.429	0.407	**0.042**	0.059	1.25	**0.0571**	**1**	**1**	**1**	0.825	0.963	0.878
		Yolov5-3d (large + GTA)	0.629	0.448	0.404	0.046	**0.058**	**1.18**	0.0622	0.998	0.998	**1**	0.808	0.965	**0.913**
GTA	Voiture	Yolov3-3d (w GT RoIs)	-	-	-	0.055	0.385	4.74	0.0761	0.990	0.999	0.999	0.860	**0.999**	0.993
		Yolov3-3d (w/o GT RoIs)	0.477	0.915	0.778	0.129	2.280	9.59	0.2170	0.873	0.938	0.965	0.812	0.894	0.905
		Yolov5-3d (small)	0.921	0.942	0.957	0.062	0.222	3.40	0.0749	0.998	**1**	**1**	**0.876**	0.982	**0.997**
		Yolov5-3d (small SA)	0.906	0.937	0.947	0.041	0.134	2.75	0.0542	0.998	**1**	**1**	0.873	0.980	0.995
		Yolov5-3d (medium)	**0.930**	0.951	**0.964**	0.033	0.086	2.18	0.0440	**0.999**	**1**	**1**	0.865	0.984	**0.997**
		Yolov5-3d (large)	**0.930**	0.905	0.919	**0.029**	**0.075**	**2.05**	**0.0402**	**0.999**	**1**	**1**	0.867	0.984	**0.997**
	Motocyclette	Yolov3-3d (w GT RoIs)	-	-	-	0.062	0.266	3.84	0.0753	**1**	**1**	**1**	**0.918**	**1.00**	0.967
		Yolov3-3d (w/o GT RoIs)	0.614	0.764	0.725	0.186	3.580	13.70	0.2810	0.691	0.878	0.967	0.901	0.689	0.909
		Yolov5-3d (small)	0.937	0.925	0.935	0.061	0.190	2.99	0.0755	0.998	**1**	**1**	0.877	0.976	0.993
		Yolov5-3d (small SA)	0.935	0.911	0.918	0.041	0.116	2.42	0.0548	0.998	**1**	**1**	0.871	0.975	0.993
		Yolov5-3d (medium)	0.934	**0.938**	**0.944**	0.034	0.086	2.14	0.0463	0.998	**1**	**1**	0.876	0.980	**0.995**
		Yolov5-3d (large)	**0.944**	0.933	0.943	**0.029**	**0.070**	**1.95**	**0.0414**	**0.999**	**1**	**1**	0.876	0.981	0.994
	Personne	Yolov3-3d (w GT RoIs)	-	-	-	0.116	0.612	4.56	0.1590	0.877	0.966	1.00	**0.963**	**0.997**	0.856
		Yolov3-3d (w/o GT RoIs)	0.263	0.659	0.488	0.372	10.500	15.10	0.4530	0.620	0.834	0.903	0.961	0.812	0.735
		Yolov5-3d (small)	**0.932**	0.814	0.834	0.049	0.170	2.92	0.0674	0.996	**1**	**1**	0.872	0.970	**0.957**
		Yolov5-3d (small SA)	0.913	0.790	0.798	0.038	0.127	2.60	0.0538	**0.998**	**1**	**1**	0.868	0.970	0.920
		Yolov5-3d (medium)	0.931	0.832	0.844	0.030	0.088	**2.16**	0.0452	0.997	**1**	**1**	0.878	0.974	0.955
		Yolov5-3d (large)	0.929	**0.833**	**0.847**	**0.028**	**0.087**	2.17	**0.0435**	0.997	**1**	**1**	0.872	0.975	0.953

Tableau 6.7: Résultats quantitatifs de notre évaluation sur les jeux de données KITTI et GTA. Les résultats de notre méthode précédente sont affichés avec ceux de notre nouvelle méthode. Notre nouvelle méthode est nettement plus performante que la précédente, notamment en termes de précision d'estimation de la profondeur.

6.5.8 Limitations

Les résultats de cette nouvelle approche sont très prometteurs, nous avons donc mené de nouvelles expériences pour explorer les limites de cette approche. Dans les résultats que nous avons présentés dans le chapitre précédent, l'évaluation a été réalisée sur le même jeu de données que celui de l'entraînement et avec la même résolution d'image. Il s'agit d'une pratique courante car elle nous permet d'obtenir une évaluation représentative de la performance de nos méthodes. Cependant, ces évaluations ne nous permettent pas d'évaluer nos méthodes dans des conditions réelles où les images sont différentes de celles utilisées lors de l'entraînement. Nous avons donc essayé d'évaluer la capacité de généralisation de notre réseau à des environnements inconnus avec différentes résolutions d'images.

Pour ce faire, nous avons utilisé un modèle "Large" entraîné sur un jeu de données particulier et nous l'avons évalué sur un autre. Nous avons également fait varier la résolution de l'image entre l'entraînement et la validation afin de mesurer la perte éventuelle de précision. Nos résultats sont présentés dans le tableau 6.8.

Ces résultats nous montrent que lorsque nous changeons la résolution de l'image, nous obtenons une perte significative sur la précision de l'estimation de la profondeur tout en gardant une précision légèrement inférieure pour les autres paramètres. Nous pouvons expliquer cela par le fait que lorsqu'une image change de résolution, l'apparence des objets reste relativement similaire, ce qui par conséquent ne conduit pas nécessairement à une dégradation significative de la détection, de la classification, de la

prédiction du centre et de l'orientation. De même, nous n'observons pas d'augmentation drastique de l'erreur de prédiction de la dimension, ce qui est certainement dû à notre approche consistant à utiliser des hypothèses tels que la dimension moyenne selon la classe de l'objet pour obtenir notre prédiction finale.

La perte importante de précision sur l'estimation de la profondeur est due au fait que notre algorithme nous donne directement la prédiction en mètres et qu'il ne prend pas en compte la matrice intrinsèque de la caméra. Ainsi, lorsque la résolution change, la taille de l'objet sur l'image change également. Si notre méthode est entraînée sur une résolution spécifique, elle aura appris à prédire la distance des objets en fonction de leur taille sur l'image et les prédictions seront faussées.

Le même raisonnement peut être appliqué lorsque nous changeons de base de données, la matrice intrinsèque des caméras utilisées étant différente, les images elles-mêmes se retrouvent avec un aspect différent. Cela conduit inévitablement à une perte de précision.

6.5.9 Normalisation des images

Afin de surmonter les problèmes décrits dans la section ci-dessus, nous avons appliqué une méthode de "normalisation" des images qui vise à les transformer afin d'imiter les images prises par une caméra similaire à celle utilisée lors de l'entraînement. Il s'agit d'un problème particulièrement important à résoudre car nous souhaitons que notre approche soit applicable à tout type de caméra sans nécessiter de ré-entraînement.

Nous avons utilisé les fonctions fournies par OpenCV [57] et notamment la fonction *undistord* afin de transformer nos images utilisées lors de l'inférence en images similaires utilisées lors de l'entraînement. Les seules informations nécessaires sont la résolution utilisée pendant l'entraînement, la matrice intrinsèque utilisée pendant l'entraînement et enfin la matrice intrinsèque de la caméra qui a acquis les images utilisées pendant l'inférence. Une comparaison des résultats est présentée dans le tableau 6.8 et un exemple des images dans la Figure 6.11.

Figure 6.11: Exemple d'une image provenant du jeu de donnée KITTI transformé en image similaire à celles présentes dans la base de donnée NuScenes. L'image du haut est l'image original de KITTI, celle d'en bas à droite est une image tirée de NuScenes et celle d'en bas à gauche est l'image de KITTI transformée.

Base de donnée d'entrainement	Résolution d'entrainement	Résolution d'inférence	Détection 2D			Distance							Dimensions	Centre	Orientation
			AP	R	mAP@0.5	RE	SRE	RMSE	log RMSE	α_1	α_2	α_3	DS	CS	OS
KITTI	672x224	672x224	0.897	0.752	0.732	0.0483	0.0987	1.82	0.0733	0.989	0.998	0.999	0.883	0.959	0.906
	672x224	1312x416	0.835	0.821	0.779	0.178	1.15	6.61	0.218	0.6	0.997	0.999	0.866	0.949	0.928
GTA	1312x768	1312x416	0.839	0.603	0.584	0.140	0.784	5.21	0.157	0.846	0.999	1.0	0.851	0.967	0.955
	1312x768	1312x768	0.823	0.593	0.571	0.114	0.584	4.63	0.134	0.913	1.0	1.0	0.848	0.962	0.955
NuScenes	1312x768	1312x416	0.857	0.678	0.697	0.455	5.97	14.7	0.378	0.0766	0.945	0.999	0.651	0.965	0.977
	1312x768	1312x768	0.843	0.642	0.635	0.0654	0.198	2.72	0.0776	0.994	1.0	1.0	0.654	0.964	0.986

Tableau 6.8: Résultats quantitatifs sur la base de données KITTI de plusieurs modèles *Large* entraînés sur différentes bases et avec différentes résolutions. Seule la classe des voitures est évaluée. La normalisation des images pour imiter la résolution d'entraînement atténue la perte de précision due au changement de base de données ou de résolution.

Nous pouvons voir que grâce à cette "normalisation" nous avons réduit drastiquement l'erreur de l'estimation de la profondeur lorsque nous changeons de base de données si celle-ci est aussi une base d'images réelles. Le gain en précision pour le modèle entraîné sur la base GTAV lorsque nous utilisons la "normalisation" est moindre car nous supposons que l'apparence des images entre le domaine virtuel et le domaine réel est trop importante. Les gains en précision sur l'estimation de la profondeur sont donc minimes.

Le faible score du modèle entraîné sur Nuscenes pour la prédiction de la dimension est dû au fait que NuScenes n'utilise pas la même taille de boîte de délimitation 3D pour les objets que KITTI. En effet, sur Nuscenes, les objets sont souvent donnés avec des dimensions beaucoup plus grandes que dans la réalité.

6.5.10 Création d'un modèle optimal

Pour approfondir notre travail, nous avons également mené des recherches pour créer un modèle optimal en prenant en compte le nombre de paramètres et ses performances en termes de précision. Un réseau qui a trop de paramètres aura un temps de calcul beaucoup plus élevé tout en ayant une plus grande tendance à se surentraîner, ce qui dégrade considérablement la précision du modèle. Un modèle qui n'a pas assez de paramètres n'apprendra pas bien et aura une précision plus faible.

Dans notre réseau, le nombre de paramètres définissant la complexité de notre modèle dépend largement du nombre de couches (profondeur) et du nombre de filtres (largeur). Le temps de calcul et la précision de notre modèle dépendront également de la résolution des images d'entrée. Un travail a déjà été effectué pour trouver un ensemble optimal (profondeur, largeur, résolution de l'image) dans EfficientNet [105]. Dans ce travail, les auteurs ont déterminé que la complexité du modèle, mesurée en Flops, était liée à la largeur (w), la profondeur (d) et la résolution de l'image (r) comme décrit dans la relation ci-dessous:

$$Flops \sim w^2 \times r^2 \times d \tag{6.19}$$

Pour une valeur de complexité fixe, les auteurs ont effectué une recherche sur grille pour trouver la largeur, la hauteur et la résolution d'image qui donneraient la meilleure précision. Afin de créer un modèle encore plus léger pour notre approche, nous avons choisi de mener une démarche similaire. Tout d'abord, nous avons défini la complexité de notre modèle avec l'objectif d'obtenir un temps de calcul de $33ms/image$ sur la Jetson TX2. La complexité choisie est de $\tilde{1}0$ GFlops. Nous sélectionnons ensuite toutes les combinaisons (largeur, profondeur, résolution) qui nous permettent d'obtenir cette même complexité de réseau. Enfin, nous entraînons chacun de ces modèles pendant 20 époques sur notre jeu de données GTAV.

Puisque nous créons de nouveaux modèles, nous ne pouvons pas faire un transfert de connaissances avec des modèles pré-entraînés sur COCO comme dans les entraînements précédents. L'entraînement se fait à partir de zéro, ce qui nécessite une base de données d'entraînement beaucoup plus importante, nous avons donc choisi d'utiliser notre base de données GTAV. Au total, nous avons effectué 30 sessions

Base de donnée	Résolution	Détection 2D			Distance							Dimensions	Centre	Orientation	# Paramètres	temps/img
		AP	R	mAP@0.5	RE	SRE	RMSE	log RMSE	α_1	α_2	α_3	DS	CS	OS		
KITTI	608x192	0.642	0.518	0.469	0.056	0.133	2.15	0.081	0.985	0.998	0.999	0.871	0.953	0.863	6.0M	28ms
Nuscenes	608x352	0.527	0.431	0.374	0.054	0.205	3.11	0.074	0.985	1	1	0.838	0.962	0.920	6.0M	28ms

Tableau 6.9: Résultats quantitatifs de notre nouveau modèle optimal évalué sur une Nvidia Jetson TX2. Les modèles testés ont été entraînés sur la base de KITTI et NuScenes.

d'entraînement, qui ont ensuite été comparées dans une évaluation pour déterminer le modèle optimal. Si nous définissons $w = 1$ et $d = 1$ comme la largeur et la hauteur du modèle *Large* et $r = 1$ pour définir une image de résolution 1280×720, le modèle optimal que nous avons trouvé a comme combinaison ($w = 0.4$, $d = 0.5$, $r = \frac{576}{1280} = 0.45$).

Nous avons ensuite suivi la même procédure pour entraîner les autres modèles (*Large*, *Medium*, etc.). Nous avons d'abord entraîné Yolov5 avec les mêmes dimensions que ce nouveau modèle sur la base de données de détection d'objets 2D COCO pendant 300 époques. Les poids pré-entraînés des premières couches sont ensuite transférés à notre modèle de détection 3D pour 15 époques d'entraînement sur le jeu de données GTAV et enfin nous entraînons ce modèle sur la base de données KITTI ou NuScenes.

Ce modèle a ensuite été implémenté en utilisant la bibliothèque TensorRT de Nvidia afin de réduire considérablement le temps d'inférence.

Ce nouveau modèle correspond au meilleur compromis entre faible temps de calcul et précision, comme l'illustre le tableau 6.9.

6.6 Suivi d'objets en 3D

Afin de prévenir les risques de collisions et d'éviter les dangers de la circulation, tant sur la route que sur les rails, la détection et la localisation des objets ne suffisent pas. En effet, il est nécessaire de prédire la position future des objets et de déterminer s'ils peuvent représenter un risque de collision. Dans le chapitre 2, nous avions déjà créé une méthode pour suivre des objets en 3D afin de prédire leurs trajectoires. Cette méthode était basée sur le filtre de Kalman pour prédire les positions futures des objets dans les images suivantes, et ce dans l'espace 3D. Les positions des objets étaient obtenues à partir d'un algorithme de détection 2D (Yolov3) combiné à un second algorithme d'estimation de distance (MadNet). Les prédictions obtenues, bien que prometteuses, n'étaient pas au niveau de notre dernière méthode de détection 3D.

La qualité du suivi d'objet dépend largement de la qualité des prédictions de position pour fournir des résultats précis. Prenant également en compte le fait que le filtre de Kalman est léger et rapide, nous avons choisi de conserver une approche similaire pour le suivi d'objets 3D. Plusieurs méthodes de suivi d'objets 3D utilisent déjà une approche similaire, combinant un détecteur 3D et un filtre de Kalman [76], mais aucune d'entre elles ne peut être considérée comme compatible avec une application en temps réel. Nous appelons tracklet les instances d'objets qui sont suivis par notre algorithme. Notre suivi d'objets peut être séparé en trois parties:

(1) Association entre les détections et les tracklets

(2) Création de tracklets pour les détections non-associées

(3) Prédiction de la position des tracklets sur la frame suivante

6.6.1 Association détection/tracklet

Afin de suivre avec précision les objets détectés, il est nécessaire d'associer les détections du temps t aux tracklets de t-1 dont la position a a été prédite pour le temps t. Pour ce faire, nous calculons un

score d'affinité entre la détection et le tracklet avec la relation (6.20). C'est-à-dire un score permettant de caractériser la probabilité qu'un objet soit le même entre deux images consécutives.

$$score = 0.5 \times Score_{coord} + 0.5 \times Score_{Feat} \tag{6.20}$$

avec $Score_{coord}$ l'affinité des distances entre la position de détection et la position prédite du tracklet. $Score_{Feat}$ représente l'affinité des caractéristiques du réseau de la région où se trouve l'objet au temps t et celles des tracklets au temps t-1.

Soit $(X_{det}, Y_{det}, Z_{det})$ et $(X_{trk}, Y_{trk}, Z_{trk})$ les positions du centre de l'objet détecté et la position prédite du tracklet, le score d'affinité des coordonnées $Score_{coord}$ est calculé en utilisant la distance euclidienne entre ces deux points 3D comme détaillé dans la relation (6.21):

$$Score_{coord} = exp\left(-\sqrt{(X_{det} - X_{trk})^2 + (Y_{det} - Y_{trk})^2 + (Z_{det} - Z_{trk})^2}\right) \tag{6.21}$$

En suivant les travaux proposés dans Deep Sort [106], nous utilisons une nouvelle métrique basée sur l'apparence des objets pour améliorer l'association entre les détections et les tracklets. Le calcul de cette métrique est possible car notre réseau de détection est capable de détecter des objets et de les classer, il a donc appris à reconnaître ces objets en fonction de leur apparence. Par conséquent, nous pouvons utiliser les couches intermédiaires de notre réseau et la fonction d'alignement des caractéristiques pour obtenir une matrice de caractéristiques pour chacun de nos objets, qui devient notre descripteur d'apparence d'objet.

Soit $\mathbf{Feat}_{det}$ et $\mathbf{Feat}_{trk}$ les cartes de caractéristiques extraites à l'aide de la fonction d'alignement de caractéristiques sur les régions contenant la détection à l'image t et le tracklet au temps t-1. Ces cartes de caractéristiques se présentent sous la forme d'une matrice de dimensions (N_{filter},7,7) avec N_{filter} le nombre de filtres dans la couche réseau. Le score d'affinité des caractéristiques $Score_{Feat}$ est ensuite calculé en utilisant la distance euclidienne entre ces deux matrices. L'objectif de ce score est d'extraire un score de similarité d'apparence entre l'objet détecté et l'objet qui est suivi (tracklet). Le calcul de ce score est détaillé dans l'équation (6.22):

$$Score_{Feat} = exp\left(-\sqrt{\sum_{n}^{N_{filtre}} \sum_{k=1}^{7} \sum_{j=1}^{7} \frac{(\hat{f}_{n,k,j} - f_{n,k,j})^2}{500}}\right) \tag{6.22}$$

avec f et $\hat{f}$ éléments des matrices $\mathbf{Feat}_{trk}$ et $\mathbf{Feat}_{det}$ respectivement.

Pour chaque détection, nous calculons le score d'affinité *score* de la détection avec tous les tracklets. A la fin, nous pouvons créer une matrice contenant tous les scores entre les détections et les tracklets. Cette matrice de dimensions (N_{det},N_{trk}) représente le nombre de détections et le nombre de tracklets. L'association devient alors un problème d'optimisation combinatoire qui peut être résolu en utilisant l'algorithme hongrois [107] afin d'associer les détections les plus appropriées aux tracklets.

Si aucun tracklet n'est associé à une détection, un nouveau tracklet est créé pour celle-ci. De la même manière, si aucune détection n'est associée à un tracklet, celui-ci est supprimé et le suivi est terminé.

6.6.2 Prédiction

Le mouvement des objets suivis dans l'espace 3D est modélisé à l'aide d'un filtre de Kalman. Nous utilisons l'hypothèse d'une vélocité constante pour ce modèle [108]. Ainsi, lorsqu'un objet détecté est initialisé dans notre approche et devient un tracklet, nous supposons que sa vitesse par rapport à la

caméra est nulle.

Nous pouvons utiliser cette hypothèse car notre fréquence d'échantillonnage (>30hz) est suffisamment élevée pour que l'accélération d'un objet entre deux images de notre caméra soit très faible, ce qui nous permet de l'approximer à 0. Nous pouvons formuler ce modèle comme décrit dans (6.23):

$$\begin{bmatrix} X_t \\ Y_t \\ Z_t \\ \dot{X}_t \\ \dot{Y}_t \\ \dot{Z}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{t-1} \\ Y_{t-1} \\ Z_{t-1} \\ \dot{X}_{t-1} \\ \dot{Y}_{t-1} \\ \dot{Z}_{t-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \Delta\dot{X} \\ \Delta\dot{Y} \\ \Delta\dot{Z} \end{bmatrix} \tag{6.23}$$

Avec (X,Y,Z) la position dans l'espace 3D et $(\dot{X},\dot{Y},\dot{Z})$ la différence de position entre le temps t et $t-1$ (vélocité). Enfin $(\Delta\dot{X}, \Delta\dot{Y},\Delta\dot{Z})$ est la variation de la vitesse qui peut être modélisée par une distribution gaussienne $N(0,\sigma_v)$. Avec σ_v fixé en fonction du degré de notre certitude sur l'estimation a priori de la vélocité. Afin de modéliser ce modèle avec le filtre de Kalman, nous définissons d'abord le vecteur de mesure z_t donné par l'équation (6.24):

$$\mathbf{z}_t = \mathbf{H} \begin{bmatrix} X_t \\ Y_t \\ Z_t \\ \dot{X}_t \\ \dot{Y}_t \\ \dot{Z}_t \end{bmatrix} + \gamma_t \tag{6.24}$$

Avec

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{6.25}$$

Avec γ l'erreur de mesure que nous supposons suivre une distribution gaussienne de moyenne nulle avec une covariance σ_γ. La phase de prédiction du filtre de Kalman s'écrit comme suit:

$$\begin{bmatrix} \bar{X} \\ \bar{Y} \\ \bar{Z} \\ \bar{\dot{X}} \\ \bar{\dot{Y}} \\ \bar{\dot{Z}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \\ \hat{\dot{X}} \\ \hat{\dot{Y}} \\ \hat{\dot{Z}} \end{bmatrix} \tag{6.26}$$

$$\bar{\Sigma}_t = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \hat{\Sigma}_{t-1} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_v & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_v & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_v \end{bmatrix} \tag{6.27}$$

Lorsqu'un objet est suivi pour la première fois et qu'un tracklet est créé, σ_v est initialisé à 0, les estimations de position initiale $(\hat{X}_{t=0},\hat{Y}_{t=0}, \hat{Z}_{t=0})$ sont initialisées aux valeurs de position de la détection et les vitesses initiales $(\hat{\dot{X}}_{t=0},\hat{\dot{Y}}_{t=0},\hat{\dot{Z}}_{t=0})$ sont nulles. On donne à la matrice de covariance $\hat{\Sigma}_{t=0}$ une valeur très

élevée pour modéliser notre incertitude sur la première prédiction.

Lorsqu'une détection est associée à un objet précédemment suivi, nous mettons à jour notre filtre de Kalman pour améliorer la prédiction. Cette opération est décrite ci-dessous:

$$\boldsymbol{K} = \bar{\Sigma}_t \boldsymbol{H}^T \left(\boldsymbol{H} \bar{\Sigma} \boldsymbol{H}^T + \sigma_\gamma \right)^{-1} \tag{6.28}$$

$$\begin{bmatrix} \hat{X} \\ \hat{\dot{X}} \\ \hat{\ddot{X}} \\ \hat{Y} \\ \hat{\dot{Y}} \\ \hat{Z} \end{bmatrix} = \begin{bmatrix} \bar{X} \\ \bar{\dot{X}} \\ \bar{\ddot{X}} \\ \bar{Y} \\ \bar{\dot{Y}} \\ \bar{Z} \end{bmatrix} \boldsymbol{K} \left(\boldsymbol{z}_t - \boldsymbol{H} \begin{bmatrix} \bar{X} \\ \bar{\dot{X}} \\ \bar{\ddot{X}} \\ \bar{Y} \\ \bar{\dot{Y}} \\ \bar{Z} \end{bmatrix} \right) \tag{6.29}$$

$$\hat{\Sigma}_t = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}) \bar{\Sigma}_t \tag{6.30}$$

6.6.3 Résultats

Dans cette section, nous allons présenter les résultats obtenus par notre nouvelle approche de suivi d'objets 3D. Malgrés les nombreux travaux qui ont pu être menés, il n'a pas été possible de réaliser des évaluations quantitatives complètes par manque de temps, cependant, nous avons pu réaliser de nombreux tests qualitatifs. Ceux-ci ont été réalisés à la fois sur des jeux de données routières publics tels que KITTI et NuScenes (Figure 6.12 et Figure 6.13), mais aussi sur notre propre base de données routières/ferroviaires ESORAD (voir Figure 6.14)

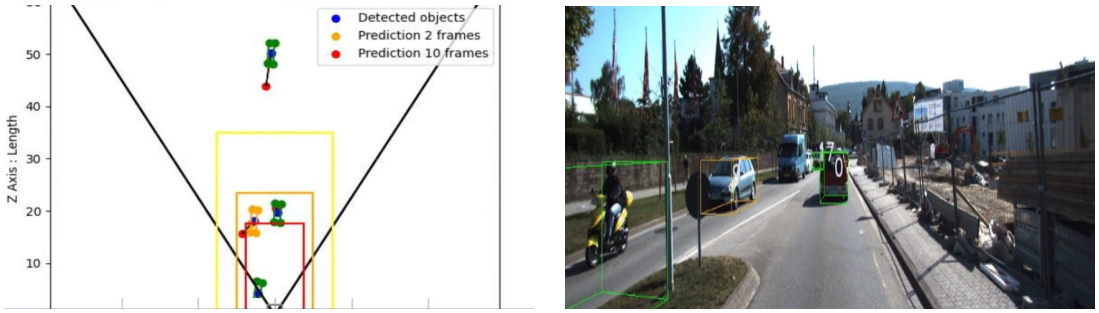

Figure 6.12: Résultats sur la base de données KITTI. A gauche sont tracées les trajectoires des objets suivis et à droite l'image avec les boîtes 3D des objets suivis dessinées.

FIGURE 6.13: Résultats obtenus par notre méthode de suivi d'objets 3D. Le modèle "Large" entraîné sur la base de données Nuscenes a été utilisé pour détecter les objets 3D. Les images présentées dans la Figure font partie d'une séquence de validation fractionnée. Dans la colonne de gauche sont présentés les résultats du suivi d'objet avec l'ID de l'objet au centre des boîtes. La lettre L indique qu'un objet est "perdu", aucune détection n'a été associée au tracklet mais nous continuons à le suivre quelques images avant de l'effacer. Dans la colonne de droite sont présentés les résultats de notre détecteur d'objets 3D.

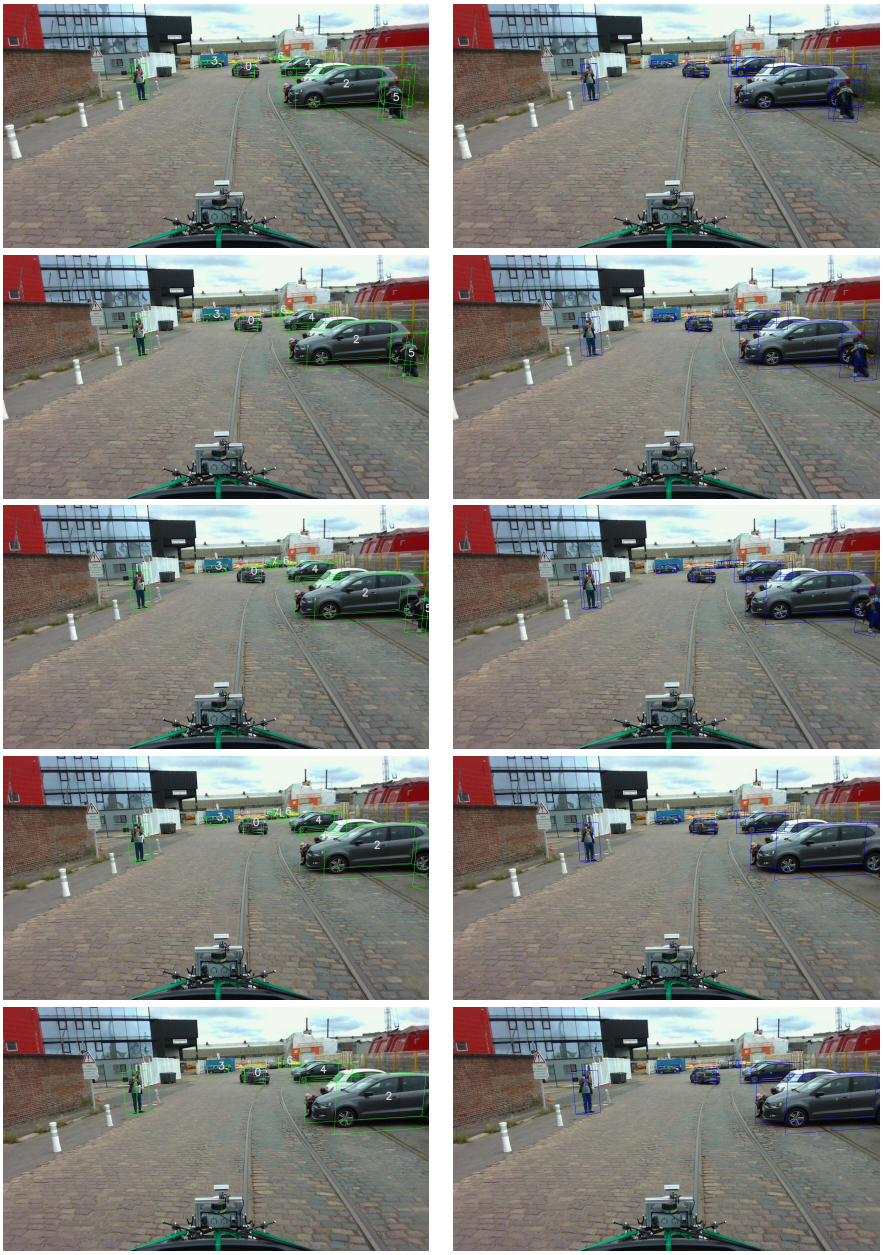

FIGURE 6.14: Résultats obtenus par notre méthode de suivi d'objets 3D sur notre base de données ESRO-RAD. Le modèle "Large" entraîné sur la base de données Nuscenes a été utilisé pour la détection d'objets 3D. Dans la colonne de gauche sont présentés les résultats du suivi d'objet avec l'ID de l'objet au centre des cases. La lettre L indique qu'un objet est "perdu", aucune détection n'a été associée au tracklet mais nous continuons à le suivre quelques images avant de l'effacer. Dans la colonne de droite sont présentés les résultats de notre détecteur d'objets 3D.

6.7 Conclusion

Dans ce chapitre, nous avons présenté la contribution principale de cette thèse qui est la création de deux nouveaux algorithmes pour la détection d'objets 3D (qui permet de les détecter et de les localiser simultanément) à partir d'images. Notre première méthode était basée sur une architecture à deux étages avec deux modules, le premier étant le détecteur d'objets 2D en temps réel Yolov3 et le second étage étant dédié à la prédiction des paramètres 3D (centre, dimensions, orientation, distance). Si cette première méthode était prometteuse, notamment grâce à son temps de calcul très faible par rapport à l'état de l'art, elle souffrait d'un manque de précision dû à la variation des régions d'intérêt (RoI) où se trouvent les objets détectés par rapport aux RoIs utilisées lors de l'entraînement de la deuxième étape.

Nous avons donc décidé de créer une deuxième méthode de détection 3D, cette fois-ci basée sur une architecture à un seul étage pour la prédiction simultanée des paramètres 3D et 2D (classe, boîte 2D, centre, dimensions, orientation, distance). Cette nouvelle approche présente également un temps de calcul bien meilleur par rapport à l'état de l'art, cependant la précision est cette fois identique voire supérieure aux méthodes existantes de l'état de l'art. Cette méthode, lorsqu'elle est pré-entraînée sur notre base de données virtuelle GTA, offre un gain de précision significatif lorsqu'elle est entraînée sur des bases de données réelles comme KITTI. Nous avons également observé que la précision de l'estimation de la distance de notre méthode se dégrade de manière significative lorsque la résolution utilisée pendant l'inférence diffère de la résolution d'entraînement. La solution que nous avons trouvée à ce problème est d'utiliser les fonctions fournies par OpenCV et plus particulièrement la fonction *undistord* afin de " reproduire " la matrice intrinsèque des images utilisées lors de l'entraînement et ainsi réduire l'erreur sur la profondeur. Cette méthode nous permet d'utiliser des modèles entraînés sur différentes bases et de les appliquer sur une autre caméra. Enfin, nous avons développé un algorithme de suivi des objets détectés afin de prédire leurs trajectoires dans l'espace 3D et ainsi anticiper un risque de collision entre un objet détecté et le véhicule.

De nouvelles expériences sont prévues prochainement pour valider notre nouvelle approche en conditions réelles à l'aide d'un véhicule équipé de caméras et d'un PC embarqué (Jetson TX2). Nous avons déjà créé un nouveau modèle, encore plus léger sans sacrifier la précision, pour réaliser ces tests. Si ces tests sont concluants, notre objectif de détection et de suivi d'objets en 3D et en temps réel sera atteint et nos travaux pourront être utilisés pour approfondir la recherche dans le domaine de l'intelligence artificielle pour la sécurité des transports routiers et ferroviaires.

Chapitre 7

Conclusion Perspectives

Dans cette thèse, nous avons abordé le problème de la détection de collision à la fois pour la voiture autonome et le train autonome en utilisant des images provenant de caméras. Nous avons d'abord décomposé ce problème en trois objectifs distincts, le premier est la détection d'objets pouvant causer un risque de collision avec le véhicule (piétons, voitures, camions, etc.) ; le deuxième objectif est de localiser l'objet détecté dans l'espace par rapport au véhicule ; et enfin le dernier objectif est de prédire la trajectoire de cet objet afin de déterminer s'il présente un risque de collision ou non.

Avec l'avènement de l'apprentissage profond, atteindre ces objectifs qui nécessitaient auparavant une multitude de capteurs différents (caméra, RADAR, LiDAR, IMU/GPS, etc.) peut désormais être théoriquement possible avec beaucoup moins de modalités de capteurs. De nombreuses méthodes existent pour détecter des objets dans l'image ou même en 3D, pour la prédiction de la profondeur, le suivi d'objets, etc. Cependant, il existe un réel manque de méthodes basées sur l'apprentissage profond dédiées aux systèmes embarqués en temps réel, en raison des coûts de calcul élevés qu'implique l'apprentissage profond. De plus, de grands acteurs industriels comme Uber, Google et Facebook ont déjà beaucoup travaillé sur les véhicules autonomes, mais le domaine des trains autonomes n'en est qu'à ses débuts. Cette thèse propose d'étudier et de concevoir des approches basées sur l'apprentissage profond, légères et compatibles avec l'embarqué pour atteindre ces objectifs à la fois pour la voiture autonome et le train autonome.

Tout d'abord, nous avons étudié la possibilité d'utiliser différentes méthodes d'apprentissage profond pour atteindre cet objectif. Par exemple, nous utilisons un détecteur d'objets 2D appelé Yolov3 [14] pour détecter l'objet, puis nous utilisons la méthode d'estimation de la profondeur pour les images stéréo MadNet [15] pour extraire une image de profondeur qui, combinée aux informations de position des objets dans l'image, nous permettra de les localiser dans l'espace par rapport au véhicule. La prédiction des trajectoires de ces objets peut ensuite être calculée à l'aide d'un algorithme de suivi d'objets 3D basé sur un filtre de Kalman. Cette première approche a rempli les objectifs de rendre possible la prédiction de collision, cependant ce type d'approche qui utilise plusieurs algorithmes d'apprentissage profond différents est coûteux en temps et ne peut être utilisé en temps réel. De plus, une étude quantitative des performances de cette approche est difficile car les évaluations de l'estimation de la profondeur ne sont pas adaptées à la localisation d'objets spécifiquement.

Afin de résoudre ce problème, un nouveau protocole d'évaluation de l'estimation de la profondeur a été conçu spécifiquement pour fournir une meilleure évaluation pour la localisation d'objets. Ce nouveau protocole permet d'obtenir une étude quantitative de la précision d'une méthode d'estimation de la profondeur en fonction de la classe de l'objet mais aussi en fonction de la distance à laquelle il est localisé.

Le manque de bases de données dédiées aux chemins de fer est également un problème majeur pour la création d'algorithmes d'apprentissage, dépendants des données pour leur entraînement mais aussi pour évaluer leurs performances en conditions réelles. L'acquisition de jeux de données contenant des scènes ferroviaires a donc été entreprise. En profitant de la haute fidélité visuelle offerte par les jeux vidéo, nous avons créé un jeu de données virtuel contenant plus de 240 000 échantillons (images, images de profondeur, objets présents, etc.) pris à la fois du point de vue d'un métro et d'une voiture. Cette nouvelle base de données permettra d'enrichir les bases d'images réelles afin d'augmenter significativement la précision des méthodes entraînées. Une base d'images réelles contenant des images ferroviaires a également été créée (ESORAD) afin de fournir une base de données ferroviaire pour l'évaluation des méthodes de perception basées sur l'apprentissage profond.

Enfin, nous avons créé deux méthodes de détection d'objets 3D différentes, qui offrent l'avantage de

détecter l'objet ainsi que sa localisation grâce à un seul réseau, ce qui permet de maintenir le temps de calcul comparativement faible par rapport à notre première approche. Notre première méthode de détection 3D était basée sur deux étapes:

- La première étape était le détecteur d'objets Yolov3 que nous avons modifié afin d'extraire à la fois la classe des objets, les régions d'intérêt (RoI) où les objets sont situés mais aussi les caractéristiques du réseau pour cette même région qui ont été obtenues en utilisant la fonction d'alignement des caractéristiques.
- La deuxième étape est un réseau convolutif que nous avons créé afin d'effectuer la prédiction des paramètres 3D (position du centre de l'objet, distance, orientation, dimensions). Cette étape prend en entrée les différentes cartes de caractéristiques et la classe d'objets fournies par l'étape 1.

Bien que cette première approche ait fourni des résultats encourageants, notamment en termes de faible temps de calcul, elle était loin d'atteindre la précision des méthodes de pointe. Après avoir identifié l'extraction de caractéristiques comme le problème principal de cette approche, nous avons commencé la création d'un deuxième réseau CNN pour la détection 3D, mais cette fois-ci, il ne comportera qu'une seule étape. Basé sur le détecteur d'objets Yolov5[99], plus récent que Yolov3, nous l'avons modifié (notamment ses Anchor Boxes utilisées pour la prédiction) afin de prédire, en plus des classes et des régions d'intérêt, les différents paramètres 3D. Ce nouveau réseau est aussi précis que les méthodes de l'état de l'art tout en étant suffisamment léger pour permettre une exécution en temps réel sur un système embarqué comme le Nvidia Jetson TX2. En combinant cette méthode avec un filtre de Kalman, nous avons pu concevoir un algorithme de suivi d'objets 3D qui permet enfin, en temps réel, de prédire la trajectoire des objets afin d'avertir en cas de risque de collision. Afin de pallier la perte de précision causée par un changement d'aspect des images par rapport à celles utilisées lors de l'entraînement (changement de base de données, changement de résolution, etc.), nous avons également établi une nouvelle méthode utilisant la bibliothèque de traitement d'images OpenCV afin d'imiter l'aspect des images d'entraînement. Enfin, nous avons mené d'autres expériences afin de créer un modèle léger avec une précision optimale, spécifiquement pour une utilisation en conditions réelles.

Les travaux réalisés au cours de cette thèse nous ont permis de faire un grand pas en avant pour une utilisation dans le monde réel, notamment avec notre nouvelle approche qui nous permet d'atteindre tous nos objectifs fixés au début de cette thèse. Cependant, il reste encore quelques problèmes à résoudre avant d'atteindre cette étape:

- Tester l'algorithme sur une carte embarquée dans un véhicule équipé d'une caméra afin de confirmer le temps d'exécution en temps réel et la précision sur des images différentes du jeu de données d'entraînement.
- Évaluer quantitativement notre approche pour le suivi des objets et la prédiction de leur trajectoire.
- Fusionner et normaliser les données provenant de nombreux jeux de données (KITTI, NuScenes, etc.) afin de créer une base de données plus importante et d'obtenir un modèle encore plus précis.

Nous pouvons également envisager d'utiliser des indices fournis par l'environnement, par exemple nous savons qu'une voiture a toujours un contact avec le sol ce qui peut être utilisé pour la prédiction de la position du véhicule en 3D (ce qui a été expérimenté dans [92]). Nous prévoyons également de mener des expériences de distillation de connaissances, c'est-à-dire d'utiliser deux modèles, un déjà entraîné et qui a plus de paramètres qui jouera le rôle de "professeur" et un autre beaucoup plus léger que nous cherchons à entraîner ("élève"). Pour chaque étape de l'apprentissage, le réseau "professeur" fournira ses cartes caractéristiques de plusieurs parties du réseau qui seront utilisées, en plus de celles déjà existantes, comme vérité terrain pour le réseau "élève" lors du calcul de la perte. Le but de cette technique est d'augmenter la précision du réseau "élève" en profitant de la "connaissance" du réseau "professeur",

mais elle nous permettra aussi d'avoir une convergence plus rapide pendant l'apprentissage.

Bibliographie

[1] "Accidents de la route." [Online]. Available: https://www.who.int/fr/news-room/fact-sheets/detail/road-traffic-injuries

[2] "The 6 levels of vehicle autonomy explained." [Online]. Available: https://www.synopsys.com/automotive/autonomous-driving-levels.html

[3] B. Shahian Jahromi, T. Tulabandhula, and S. Cetin, "Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles," *Sensors (Basel, Switzerland)*, vol. 19, no. 20, p. E4357, October 2019. [Online]. Available: https://europepmc.org/articles/PMC6833089

[4] M. Zhao, A. Mammeri, and A. Boukerche, "Distance measurement system for smart vehicles," in *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2015, pp. 1–5.

[5] J. Palacín, T. Pallejà, M. Tresanchez, R. Sanz, J. Llorens, M. Ribes-Dasi, J. Masip, J. Arno, A. Escola, and J. R. Rosell, "Real-time tree-foliage surface estimation using a ground laser scanner," *IEEE transactions on instrumentation and measurement*, vol. 56, no. 4, pp. 1377–1383, 2007.

[6] B. Kang, S.-J. Kim, S. Lee, K. Lee, J. D. Kim, and C.-Y. Kim, "Harmonic distortion free distance estimation in tof camera," in *Three-Dimensional Imaging, Interaction, and Measurement*, vol. 7864. International Society for Optics and Photonics, 2011, p. 786403.

[7] A. Mauri, R. Khemmar, B. Decoux, N. Ragot, R. Rossi, R. Trabelsi, R. Boutteau, J.-Y. Ertaud, and X. Savatier, "Deep learning for real-time 3d multi-object detection, localisation, and tracking: Application to smart mobility," *Sensors*, vol. 20, no. 2, p. 532, 2020.

[8] A. Mauri, R. Khemmar, R. Boutteau, B. Decoux, J.-Y. Ertaud, and M. Haddad, "A new evaluation approach for deep learning-based monocular depth estimation methods," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[9] A. Mauri, R. Khemmar, B. Decoux, J.-y. Ertaud, M. Haddad, and R. Boutteau, "Une nouvelle approche pour l'évaluation des méthodes monoculaires d'estimation de la profondeur basées sur l'apprentissage profond," in *ORASIS 2021*. Saint Ferréol, France: Centre National de la Recherche Scientifique [CNRS], Sep. 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03339671

[10] A. Mauri, R. Khemmar, B. Decoux, T. Benmoumen, M. Haddad, and R. Boutteau, "A comparative study of deep learning-based depth estimation approaches: Application to smart mobility," in *2021 8th International Conference on Smart Computing and Communications (ICSCC)*, 2021, pp. 80–84.

[11] A. Mauri, R. Khemmar, B. Decoux, M. Haddad, and R. Boutteau, "Real-time 3d multi-object detection and localization based on deep learning for road and railway smart mobility," *Journal of imaging*, vol. 7, no. 8, p. 145, 2021.

[12] ——, "Lightweight convolutional neural network for real-time 3d object detection in road and railway environments," *Journal of Real-Time Image Processing*, pp. 1–18, 2022.

[13] R. Khemmar, A. Mauri, C. Dulompont, J. Gajula, V. Vauchey, M. Haddad, and R. Boutteau, "Road and railway smart mobility: a high-definition ground truth hybrid dataset," *Sensors*, vol. 22, no. 10, p. 3922, 2022.

[14] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[15] A. Tonioni, F. Tosi, M. Poggi, S. Mattoccia, and L. D. Stefano, "Real-time self-adaptive deep stereo," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 195–204.

[16] Kalman, "A new approach to linear filtering and prediction problems," in *Transactions of the ASME Journal of Basic Engineering*, 1960, pp. 35–45.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[18] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "Textboxes: A fast text detector with a single deep neural network," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[21] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3642–3649.

[22] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine learning*, vol. 37, no. 3, pp. 277–296, 1999.

[23] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[24] "Cs231n convolutional neural networks for visual recognition." [Online]. Available: https://cs231n.github.io/

[25] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[26] Sadgfasdf, "Convolutional neural network demo gif by sadgfasdf," Jun 2022. [Online]. Available: https://gfycat.com/fr/anguishedpastaidi-machine-learning-neural-networks-mnist

[27] S. Saha, "A comprehensive guide to convolutional neural networks-the eli5 way," Dec 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[28] "Max-pooling / pooling." [Online]. Available: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[30] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[31] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *CoRR*, vol. abs/1710.05941, 2017. [Online]. Available: http://arxiv.org/abs/1710.05941

[32] D. Misra, "Mish: A self regularized non-monotonic neural activation function," *CoRR*, vol. abs/1908.08681, 2019. [Online]. Available: http://arxiv.org/abs/1908.08681

[33] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[34] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[35] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2017.

[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[37] "Understanding region of interest - part 2 (roi align)." [Online]. Available: https://erdem.pl/2020/02/understanding-region-of-interest-part-2-ro-i-align

[38] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[39] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[40] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset," in *CVPR Workshop on the Future of Datasets in Vision*, vol. 2, 2015.

[41] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[42] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[44] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov *et al.*, "The open images dataset v4," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.

[45] N. Ragot, R. Khemmar, A. Pokala, R. Rossi, and J.-Y. Ertaud, "Benchmark of visual slam algorithms: Orb-slam2 vs rtab-map," in *2019 Eighth International Conference on Emerging Security Technologies (EST)*. IEEE, 2019, pp. 1–6.

[46] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[47] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[48] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

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

[50] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[51] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[52] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *2016 Fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 239–248.

[53] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[54] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[55] C. Godard, O. Mac Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," *arXiv preprint arXiv:1806.01260*, 2018.

[56] F. Tosi, F. Aleotti, M. Poggi, and S. Mattoccia, "Learning monocular depth estimation infusing traditional stereo knowledge," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9799–9809.

[57] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[58] "Depth map from stereo images." [Online]. Available: https://docs.opencv.org/4.x/dd/d53/tutorial_py_depthmap.html

[59] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4040–4048.

[60] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61–88, 2020.

[61] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[62] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," 2008.

[63] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 270–279.

[64] D. Min and K. Sohn, "Cost aggregation and occlusion handling with wls in stereo matching," *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1431–1442, 2008.

[65] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," *arXiv preprint arXiv:1907.10326*, 2019.

[66] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5410–5418.

[67] X. Guo, K. Yang, W. Yang, X. Wang, and H. Li, "Group-wise correlation stereo network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3273–3282.

[68] H. Laga, L. V. Jospin, F. Boussaid, and M. Bennamoun, "A survey on deep learning techniques for stereo-based depth estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[69] T. Koch, L. Liebel, F. Fraundorfer, and M. Korner, "Evaluation of cnn-based single-image depth estimation methods," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[70] M. Aladem, S. Chennupati, Z. El-Shair, and S. A. Rawashdeh, "A comparative study of different cnn encoders for monocular depth prediction," in *2019 IEEE National Aerospace and Electronics Conference (NAECON)*. IEEE, 2019, pp. 328–331.

[71] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2650–2658.

[72] T. Koch, L. Liebel, F. Fraundorfer, and M. Körner, "Evaluation of cnn-based single-image depth estimation methods," *CoRR*, vol. abs/1805.01328, 2018. [Online]. Available: http://arxiv.org/abs/1805.01328

[73] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[74] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3234–3243.

[75] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision (ECCV)*, ser. LNCS, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9906. Springer International Publishing, 2016, pp. 102–118.

[76] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krahenbuhl, T. Darrell, and F. Yu, "Joint monocular 3d vehicle detection and tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5390–5399.

[77] UM and F. C. for Autonomous Vehicles. (2021) umautobots/gtavisionexport. URL:https://github.com/umautobots/GTAVisionExport,Online; Accessed July, 2021.

[78] M. Račinský. (2018) Gtavisionexport. URL:https://github.com/racinmat/GTAVisionExport/tree/master.

[79] Jotrius. (2015) Railroad engineer. URL:https://www.gta5-mods.com/scripts/railroad-engineer, Online; Accessed July, 2021.

[80] S. Yang and M. Baum, "Extended kalman filter for extended object tracking," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4386–4390.

[81] "Supervisely Homepage," https://supervise.ly/lidar-3d-cloud/, [Online; accessed 06 April 2022].

[82] "Scale Homepage," https://scale.com/, [Online; accessed 06 April 2022].

[83] W. Zimmer, A. Rangesh, and M. Trivedi, "3d bat: A semi-automatic, web-based 3d annotation toolbox for full-surround, multi-modal data streams," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1816–1821.

[84] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2147–2156.

[85] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 7074–7082.

[86] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *European conference on computer vision*. Springer, 2016, pp. 354–370.

[87] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau, "Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2040–2049.

[88] B. Xu and Z. Chen, "Multi-level fusion based 3d object detection from monocular images," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2345–2353.

[89] Z. Liu, Z. Wu, and R. Tóth, "Smoke: Single-stage monocular 3d object detection via keypoint estimation," 2020.

[90] Z. Qin, J. Wang, and Y. Lu, "Monogrnet: A general framework for monocular 3d object detection," 2021.

[91] G. Brazil and X. Liu, "M3d-rpn: Monocular 3d region proposal network for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9287–9296.

[92] Y. Liu, Y. Yixuan, and M. Liu, "Ground-aware monocular 3d object detection for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 919–926, 2021.

[93] P. Li, H. Zhao, P. Liu, and F. Cao, "Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving," 2020.

[94] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[95] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," 2019.

[96] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[97] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. International Society for Optics and Photonics, 2019, p. 1100612.

[98] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3828–3838.

[99] G. Jocher, A. Stoken, and J. B. et al., "ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration," Jan. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4418161

[100] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[101] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. J. Smola, "Resnest: Split-attention networks," *CoRR*, vol. abs/2004.08955, 2020. [Online]. Available: https://arxiv.org/abs/2004.08955

[102] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in *Advances in Neural Information Processing Systems*. Citeseer, 2015, pp. 424–432.

[103] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[104] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder, "Disentangling monocular 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1991–1999.

[105] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

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

[107] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[108] H. Li, "A Brief Tutorial On Recursive Estimation With Examples From Intelligent Vehicle Applications (Part II): System Models," Jul. 2014, working paper or preprint. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01018124

Prédiction et évitement d'obstacles basés deep learning. Application à la mobilité ferroviaire et routière

Résumé

Afin d'améliorer la sécurité des transports et rendre la conduite plus autonome, les véhicules doivent disposer d'une meilleure perception de leur environnement. Bien que le développement de la voiture autonome fasse l'objet d'une grande attention, le transport ferroviaire suit aussi la même voie. L'enjeu est à la fois d'améliorer le confort de conduite grâce à l'aide à la navigation et d'accroître la sécurité pendant la navigation.

Les travaux présentés dans cette thèse visent à créer un système de perception fiable utilisant une seule modalité, en l'occurrence les images d'une caméra, afin de détecter des obstacles mettant en danger la vie des passagers. Le système doit être générique afin de pouvoir être utilisé sur tout type de caméra dans un contexte de trafic sur route mais aussi sur rail. Nous utiliserons l'apprentissage profond pour atteindre cet objectif et nous présentons quatre contributions. La première est une approche basée sur la combinaison d'un détecteur d'objets (Yolov3), d'un estimateur de profondeur (MadNet) et d'un filtre de Kalman pour détecter, localiser et suivre des objets sur la voie du véhicule. La deuxième contribution est basée sur un nouveau protocole d'évaluation de l'estimation de la profondeur plus adapté aux tâches de localisation d'objets. La troisième contribution est basée sur la création de deux nouvelles bases de données, une virtuelle basée sur le jeu vidéo *Grand Theft Auto* et une réelle (ESRORAD) pour le train autonome. Enfin, notre dernière contribution est une approche pour la détection d'objets 3D basée sur Yolov5 et leur suivi basé sur un filtre de Kalman.

Les résultats obtenus par cette dernière approche montrent une réelle amélioration du temps de calcul et permettent une utilisation sur des systèmes embarqués tout en étant aussi précis que les méthodes de l'état de l'art.

Mots clés: Estimation de la boîte englobante 3D, détection multi-objets 3D, ensemble de données multimodales, apprentissage profond, mobilité intelligente.

Abstract

To improve transportation safety and make driving more autonomous, vehicles must have a better perception of their environment. Although the development of the autonomous car is receiving a lot of attention, rail transport is also following the same path. The challenge is both to improve driving comfort through navigation assistance and to increase safety during navigation.

The work presented in this thesis aims at creating a reliable perception system using a single modality, in this case, images from a camera, to detect life-threatening obstacles. The system must be generic to be used on any type of camera in a traffic context on road but also rail. We will use deep learning to achieve this goal and we present four contributions. The first one is an approach based on the combination of an object detector (Yolov3), a depth estimator (MadNet), and a Kalman filter to detect, locate and track objects on the vehicle track. The second contribution is based on a new evaluation protocol for depth estimation more suitable for object localization tasks. The third contribution is based on the creation of two new databases, a virtual one based on the video game Grand Theft Auto and a real one (ESRORAD) for the autonomous train. Finally, our last contribution is an approach to the detection of 3D objects based on Yolov5 and their tracking based on a Kalman filter.

The results obtained by this last approach show a real improvement in the computation time and allows use on embedded systems while being as accurate as the state-of-the-art methods.

Keywords: 3D bounding box estimation, 3D multi-object detection, Multi-modal dataset, Deep learning, Smart mobility.