Fusion Caméra-LiDAR à l'aide de réseaux de neurones artificiels pour la détection d'obstacles à bord de véhicules autonomes

Ruddy Théodose

Numéro d'ordre :

UNIVERSITÉ CLERMONT AUVERGNE

Ecole Doctorale Sciences Pour l'Ingénieur

Institut Pascal, UMR 6602 CNRS, Université Clermont Auvergne,
Axe ISPR : Image, Systèmes de Perception pour la Robotique

Thèse présentée pour obtenir le grade de :

DOCTEUR D'UNIVERSITÉ

Spécialité : Vision pour la robotique

par

Ruddy Théodose

Fusion Caméra-LiDAR à l'aide de réseaux de neurones artificiels pour la détection d'obstacles à bord de véhicules autonomes

Soutenue publiquement le 1er décembre 2021 devant le jury composé de :

Véronique Cherfaoui	Professeur, Université de Technologie de Compiègne	**Présidente**
Jean-Philippe Lauffenburger	Professeur, Université Haute-Alsace	**Rapporteur**
Renaud Marlet	Directeur de Recherche, École des Ponts ParisTech	**Rapporteur**
Jean-Emmanuel Deschaud	Maitre assistant HDR, École des Mines ParisTech	**Examinateur**
Thierry Chateau	Professeur, Université Clermont Auvergne	**Examinateur**
Paul Checchin	Professeur, Université Clermont Auvergne	**Directeur de thèse**
Vincent Frémont	Professeur, École Centrale de Nantes	**Co-directeur de thèse**
Dieumet Denis	Docteur, Sherpa Engineering	**Invité**

Remerciements

Je souhaite tout d'abord exprimer ma gratitude profonde à mes encadrants universitaires Paul Checchin et Vincent Frémont d'avoir accepté de m'encadrer sur ces travaux. Chacun d'entre eux m'a apporté de précieux conseils, autant sur la technique que la méthodologie et a fait preuve de patience à mon égard. J'exprime également ma reconnaissance à Thierry Chateau qui m'a prodigué de précieux conseils tout au long de cette thèse.

Je remercie également l'entreprise Sherpa Engineering qui m'a offert cette opportunité de thèse. Des remerciements particuliers à Dieumet Denis qui, par sa bienveillance, a permis un environnement de travail agréable et des correspondances facilitées avec l'entreprise malgré la distance et Xavier Dauptin avec qui j'ai pu échanger grandement.

Je remercie particulièrement les membres des équipes Comsee et Persyst avec qui j'ai pu discuter et travailler. Merci à Gauthier, Rémi, Mathieu, Yohann, Antoine, Élie, Jeremy, Thomas, Julien, Charles-Antoine, Aigerim, Vaishali, Priyanka, Anthony, Yongzhe avec qui j'ai pu partager idées, ressources scientifiques et techniques ainsi que quelques fous-rires.

Enfin, un immense merci à ma mère Claude et mes frères Cédric et Frédéric qui m'ont soutenu et réconforté durant les moments difficiles, et ce, malgré la distance qui nous sépare.

Résumé

Propulsé par les avancées technologiques, en particulier sur les capteurs LiDAR, et les réseaux de neurones profonds, le domaine de détection d'obstacles dans une scène 3D est devenu un domaine d'importance majeure dans le développement de véhicules autonomes. Néanmoins, la grande majorité des méthodes de l'état de l'art sont davantage sujettes aux défaillances lorsqu'elles sont utilisées avec des capteurs différents de ceux utilisés pour leur entraînement.

Cette thèse porte sur l'étude de solutions pour accroître la portabilité de détecteurs d'obstacles sur différentes configurations de capteurs, caméra et LiDAR. En effet, un système de détection pré-entraîné pouvant opérer sur des données produites par plusieurs modèles de capteurs peut facilement être intégré à un véhicule sans adaptation majeure ni entraînement supplémentaire sur un domaine cible. En outre, ce domaine cible n'est en général pas annoté. Un détecteur capable de traiter des entrées diverses peut ainsi contribuer à l'annotation initiale d'un ensemble d'acquisitions vierges enregistré à l'aide d'un capteur différent du modèle utilisé pour l'entraînement.

D'abord, le cadre de la thèse est défini ainsi que les jeux de données utilisés durant les travaux. Les principales métriques de performances utilisées dans ce manuscrit sont également décrites. Puis, l'évolution des méthodes reposant sur l'apprentissage profond pour les tâches de détection d'objets est retracée. Les avancées sur les images RGB ayant grandement influencé le domaine, les méthodes de détections font d'abord l'objet d'un historique. Un état de l'art sur les techniques de détection 3D est ensuite réalisé. Le sujet ayant été vastement étudié, les méthodes de la littérature sont synthétisées et regroupées en fonction des capteurs utilisés pour réaliser la tâche.

Le chapitre suivant est consacré à notre première contribution, l'élaboration d'un détecteur LiDAR dit *anchor-free* afin de réduire le nombre d'hyper-paramètres nécessaires à l'apprentissage. Le réseau utilisé est un réseau de segmentation dont l'objectif est de réaliser une pseudo carte d'occupation représentant la scène vue de dessus. La valeur de chaque pixel de cette carte donne une estimation de l'occupation de sa région correspondante dans l'espace 3D. Les obstacles sont représentés sous la forme de disques elliptiques afin de permettre l'extraction de leurs paramètres à l'aide d'algorithmes de traitement d'images et d'optimisation.

Notre seconde contribution, présentée dans le chapitre subséquent, est une étude concernant la portabilité de réseaux sur plusieurs jeux de données construits autour d'équipements variés. L'objectif est ici d'évaluer ce qu'un réseau entraîné avec un unique jeu de données peut accomplir dans des contextes très différents. Cette étude est centrée autour d'un réseau LiDAR et de deux réseaux caméra−LiDAR. Afin d'améliorer la résilience aux variations de domaine, nous avons développé deux techniques. La première consiste en la réduction aléatoire du nombre de nappes dans le nuage de points 3D utilisé en entrée du réseau. Elle permet ainsi de diversifier les distributions de points observées par le réseau au cours de l'entraînement et de devenir plus versatile vis-à-vis du capteur utilisé pour générer le nuage de points. La seconde technique consiste en une représentation des obstacles sous la forme d'une fonction gaussienne permettant l'usage de distances statistiques comme fonctions de coût. Ces deux techniques permettent une amélioration des performances sur des nuages de points bien plus épars. De plus, le détecteur LiDAR, testé sur plusieurs jeux de données, s'exécute en temps réel et fournit des détections pertinentes sur flux de données non annotés.

Mots-clés— Réseaux de neurones artificiels, détection d'objets, fusion de données, LiDAR

Abstract

Boosted by technological advances on sensors, especially on LiDAR sensors, and deep learning techniques, object detection on 3D scenes has become a major stake for the deployment of autonomous vehicles. However, most of the state-of-the-art methods tend to produce more errors when they exploit data from sensors that were not used for the training stage.

In this PhD thesis, we investigate solutions to improve the portability of detectors across various LiDAR models. In fact, a generalizable detector can be easily embedded in new vehicles. Moreover, such a detector can ease the annotation process by providing an initial set of labels on recorded data instead of annotating from scratch.

We first introduce the context of the thesis by defining the studied tasks as well as the datasets and the metrics that are used in this work.

Thereafter, we trace the evolution of deep learning techniques for object detection. Technological advances on RGB images heavily influenced the other fields. Hence, a history of 2D detection methods are introduced first and foremost. Then, a listing of state-of-the-art 3D object detection methods is developed. The topic has been greatly investigated. Consequently, these methods are summarized and grouped according to the sensor types they use.

The following chapter presents our first contribution, an anchor-free LiDAR detector in order to reduce the number of hyperparameters required for the training. The network is a segmentation network whose goal is to produce a pseudo occupancy map representing a top view of the scene. Each pixel of the map indicate an estimation of the occupancy of its corresponding 3D region. Obstacles are represented as ellipses on the map to allow the extraction of their parameters through image processing and optimization techniques.

Lastly, our second contribution deals with portability of detectors across datasets built with various setups. The goal is to determine how to improve the performances of a detector when trained on one dataset and applied on others without supplementary priors. This study revolves around one LiDAR detector and two camera−LiDAR networks. In order to improve the resistance to domain variations, two techniques were developed. The first one consists in the randomized removal of layers in the input point cloud. This technique allows to increase the variability of data observed during the training stage. The second technique consists in representing the obstacles as Gaussian functions, Statistical distances then can be used as loss functions. Both techniques improve the performances, especially on low-resolutions point clouds. Moreover, the LiDAR detector, besides running in real-time, is able to provide reliable detections on multiple datasets.

Mots-clés— Neural networks, Object detection, data fusion, LiDAR sensors

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Sommaire

1.1 Le véhicule autonome : définition et enjeux

Depuis quelques années, le sujet du véhicule autonome est devenu un sujet de recherche majeur, non seulement pour les constructeurs automobiles traditionnels mais aussi pour les grandes sociétés dites du numérique. En effet, la définition du véhicule autonome repose sur la délégation d'une partie ou de la totalité des fonctions de conduite à la machine. Des algorithmes de plus en plus complexes sont mis en place afin de garantir la sécurité des passagers. La plupart de ces algorithmes reposent sur des techniques d'intelligence artificielle, aboutissant à ce que les groupes travaillant dans le domaine occupent une place prépondérante dans le secteur.

Les enjeux du véhicule autonome sur la mobilité future sont nombreux. La sécurité routière est au cœur des problématiques sociétales, les accidents causant de nombreux décès et accidents graves. Ainsi, en 2019, l'Observatoire National Interministériel de la Sécurité Routière (ONISR) estime que les accidents ont causé en France 3498 morts et 74 165 blessés (*Sécurité Routière 2019*), la grande majorité des accidents étant causés par des facteurs humains tels que l'alcool au volant, la fatigue ou les distractions. Malgré les lois qui deviennent plus strictes en matière de sécurité routière et les messages de prévention visant à sensibiliser la population, le comportement des usagers est impossible à prédire.

Les systèmes d'aide à la conduite (ADAS) permettant d'assister et d'avertir les conducteurs sont des solutions possibles. Des systèmes tels que les avertisseurs de franchissement de lignes ou les systèmes de vitesse adaptatives (ACC pour *Adaptative Cruise Control*) existent d'ores et déjà sur des véhicules de tourisme commercialisés. La conduite autonome vise quant à elle à retirer le facteur humain. La conduite doit alors être prévisible car cadencée par un ensemble de règles définies explicitement. Cette prévisibilité rend la conduite des autres usagers ou même d'autres agents autonomes plus sûres. De plus, les temps de réaction des ordinateurs sont bien plus faibles que les réflexes humains permettant une prise de décision rapide et logique en cas de situation dangereuse.

Un autre point visé par le développement des véhicules autonomes concerne la mobilité dans les zones peu desservies, notamment en milieu rural. En effet, actuellement la grande majorité des expériences menées ont lieu dans des environnements très structurés tels que des autoroutes, des villes ou les entrepôts. Cependant, des environnements dans lesquels la signalisation est plus rare ou moins bien définie sont fréquents en province. L'adaptabilité des véhicules autonomes aura une importance capitale pour éviter l'exclusion sociale et géographique. Des initiatives telles le projet Tornado (*Tornado Mobility*) ou la navette Beti de Navya (*Navya Beti*), bien qu'elles fassent appel à des infrastructures prédéfinies et/ou des trajectoires fixes, visent l'intégration des zones faiblement desservies. Ces initiatives prennent la forme de navette communautaire plutôt que de véhicule individuel (cf. Figure 1.1).

En fonction de la proportion des tâches de conduites déléguées à la voiture et non au conducteur, la SAE international a défini en 2014 plusieurs niveaux qualifiant le degré d'autonomie du véhicule. L'échelle européenne définit six niveaux détaillés dans la figure 1.2.

Les niveaux 1 à 3 ne constituent que des aides à la conduite. De ce fait, la responsabilité revient au conducteur si un accident se produit.

- Niveau 0 : tout est manuel, le conducteur a un contrôle total sur l'ensemble des fonctions du véhicule. Les mécanismes présents constituent au mieux des dispositifs d'avertissement. Exemple de dispositif : le radar de recul ;
- Niveau 1 : le conducteur peut momentanément déléguer au véhicule une tâche de contrôle sur une seule dimension spatiale (longitudinal ou latéral). Cependant, le conducteur doit être capable de reprendre le contrôle total du véhicule. Exemple : le régulateur de vitesse (ACC ou *Adaptative Cruise Control*). Le contrôle de l'accélération est gérée par le véhicule, mais le positionnement dans la voie est toujours aux mains du conducteur ;
- Niveau 2 : le conducteur peut déléguer le contrôle sur les deux dimensions (vitesse et direction) selon certaines conditions, mais doit superviser la réalisation et reprendre la

(a) Navya Beti

(b) Easymile EZ10 utilisée lors du projet Tornado

FIGURE 1.1 – Exemples de navettes autonomes françaises. Ces navettes se déplacent le long d'une trajectoire prédéfinie, les environnements dans lesquels elles évoluent ne sont pas contrôlés.

main en cas d'échec. Exemple : stationnement automatique ;

- Niveau 3 : le conducteur peut déléguer le contrôle total du véhicule suivant certaines conditions prédéfinies. Le conducteur peut abaisser sa vigilance mais doit tout de même pouvoir reprendre le contrôle. Exemple : conduite en embouteillage ;
- Niveau 4 : le conducteur n'est plus nécessaire et le véhicule peut rouler sans contrôle extérieur dans certaines situations très précises. Exemple : conduite autonome sur autoroute ;
- Niveau 5 : le véhicule peut rouler en totale autonomie quelle que soit la route.

Les technologies déjà présentes sur le marché et accessibles au grand public visent au maximum le niveau L3 dans la mesure où le conducteur garde toujours la main en cas d'urgence. Les niveaux L4 et L5 font l'objet de très nombreuses recherches de la part des laboratoires et entreprises, occasionnant entre eux une grande concurrence. Les vidéos d'expérimentations en conditions réelles font d'ailleurs souvent office d'éléments de communication.

La conduite autonome est un exercice complexe pouvant être décomposé en de nombreuses sous-tâches, chacune nécessitant des prérequis logiciels et/ou matériels. Néanmoins, malgré la diversité des acteurs et des approches, l'ensemble des méthodes développées sont en général divisées en trois grands groupes : la perception, la planification et le contrôle. La tâche de perception est conditionnée par la nature des capteurs qui équipent le véhicule. Certains capteurs sont devenus récurrents pour la réalisation de la tâche de perception. Nous présentons dans la section suivante ces dispositifs sensoriels que l'on rencontre à bord des véhicules autonomes.

1.2 Éléments technologiques du véhicule autonome

1.2.1 Architecture globale

Les tâches qu'un véhicule autonome doit réaliser pour atteindre ses objectifs finaux peuvent être décomposées en trois grands groupes : la perception, la décision et le contrôle. La figure 1.3 expose ces différentes briques logicielles, leurs interactions entre elles et les éléments matériels.

La perception se focalise sur la capacité d'un véhicule à récupérer de l'information utile sur l'environnement dans lequel il se situe. Ce groupe se divise en deux sous groupes, la perception de l'environnement et la localisation. La perception environnementale est dirigée vers la compréhension de l'environnement telle que la position des obstacles, la reconnaissance de la signalisation ou la segmentation des éléments de la route. La localisation se concentre sur la capacité du robot à déterminer sa position dans l'environnement. Pour obtenir des informations sur l'environnement, le véhicule peut employer des capteurs pour mesurer divers phénomènes

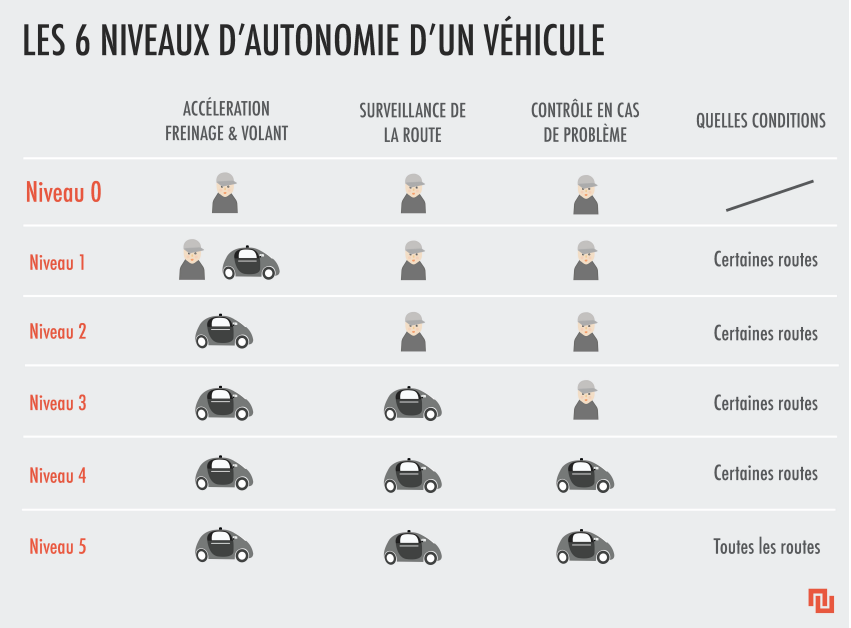


FIGURE 1.2 – Les six niveaux d'autonomie du véhicule autonome définis par la SAE International. Chaque niveau est notamment caractérisé par l'entité (le conducteur ou le véhicule) à qui l'exécution de chaque tâche est déléguée ainsi que par le contexte d'application. Le niveau 0 correspond à une conduite totalement manuelle du conducteur tandis que le niveau 5 représente une autonomie totale de la part du véhicule, les dispositifs de contrôle manuel du véhicule (par exemple un volant) ne sont plus nécessaires. Source : (CADOT, 2018).

physiques ou bien récupérer de l'information par l'intermédiaire d'autres entités communicantes telles que d'autres véhicules (*Vehicle To Vehicle* ou V2V) ou infrastructures (*Vehicle to Insfrastructure* ou V2I).

La planification ou la décision concernent le processus de choix des actions à sélectionner en vue de satisfaire les objectifs du véhicule autonome. La gestion des trajectoires d'évitement appartient notamment à cette catégorie. Les techniques appartenant à cette catégorie reposent grandement sur les résultats provenant de la partie perception.

Le contrôle se réfère à la capacité du véhicule à effectuer les actions sélectionnées. La réalisation de l'action doit prendre en compte la configuration des actionneurs au sein du véhicule ainsi que les phénomènes physiques pouvant interagir avec lui.

1.2.2 Capteurs pour l'automobile

Le terme « capteur » désigne tout dispositif dont l'objectif est de mesurer un phénomène physique. Dans le cadre de la robotique, un agent, ici un véhicule autonome, doit être capable d'estimer l'état de l'environnement dans lequel il se meut, mais également son état propre avant d'entreprendre une action. En somme, les capteurs jouent le rôle d'interface entre l'environnement libre et incertain et le choix de la décision qui sera par la suite réalisée.

Les capteurs, en fonction de ce qui est mesuré et du référentiel dans lequel ils décrivent les phénomènes, sont divisés en deux groupes qui sont décrits dans les paragraphes suivants.

Capteurs proprioceptifs

Les capteurs dits « proprioceptifs » sont chargés de mesurer l'état interne du robot. Cet état peut être décrit de multiples façons en fonction de la tâche à accomplir. Dans le cadre d'un

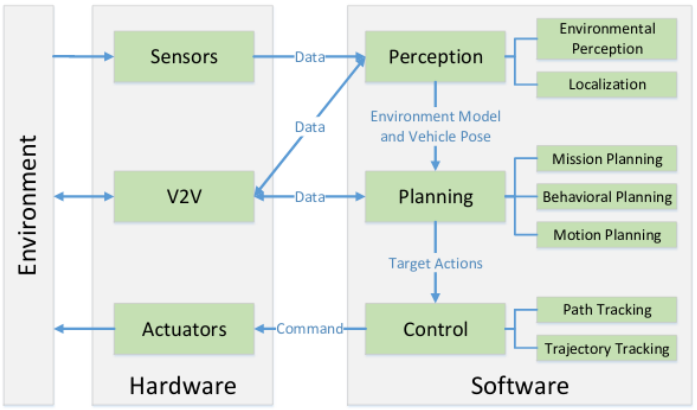


FIGURE 1.3 – Briques logicielles et matérielles d'un véhicule autonome (PENDLETON et al., 2017).

véhicule, ces capteurs peuvent notamment inclure des codeurs pour mesurer des angles (angle du volant, position angulaire des roues), des tachymètres pour la vitesse des roues ou des centrales inertielles (IMU ou *Inertial Measurement Unit*) pour évaluer le tangage et le roulis.

Capteurs extéroceptifs

Les capteurs dits « extéroceptifs », contrairement aux capteurs proprioceptifs, mesurent l'état de l'environnement externe au robot. Dans le cadre du véhicule autonome, les tâches déléguées aux capteurs extéroceptifs sont en général relatives à l'analyse de l'environnement par l'extraction des zones autorisant la navigation, l'identification et la localisation des différents obstacles pouvant porter atteinte à l'intégrité du véhicule.

Chaque capteur repose sur un phénomène physique précis. En fonction de la tâche à réaliser et des conditions de réalisation, certains capteurs sont plus adéquats que d'autres. Dans les paragraphes suivants, nous décrivons les capteurs extéroceptifs les plus communs utilisés dans le domaine.

Capteur à ultrasons Les capteurs à ultrasons mesurent le temps de vol d'ondes ultrasoniques. Le capteur dispose de deux composants principaux, un émetteur et un récepteur. L'émetteur est chargé d'émettre des ondes sonores dans une direction donnée. Si un obstacle est présent dans la direction en question, l'obstacle va réfléchir l'onde émise vers le récepteur. Connaissant la vitesse du son dans l'air (environ $344\,\mathrm{m\,s^{-1}}$) et la durée entre l'émission et la réception de l'onde, il est possible d'estimer la distance entre le capteur et l'obstacle.

La technologie est peu coûteuse. Cependant, à plus longue portée, les ondes ont plus tendance à se disperser et voient leur énergie initiale diminuer avec le carré de la distance, les rendant imperceptibles par le récepteur. De ce fait, les capteurs ultrasoniques sont utilisés à courte portée et à faible vitesse. Ces capteurs sont déjà déployés sur des véhicules de tourismes. Ils sont notamment employés en tant que dispositifs d'aide au stationnement. La figure 1.4 illustre un capteur à ultrasons disponible dans le commerce.

Radar Le terme « RADAR » est l'acronyme de « *RAdio Detection And Ranging* ». Le principe de fonctionnement repose également sur la mesure du temps de vol. La différence principale avec le sonar (capteur à ultrasons) provient du médium utilisé. Ici, des ondes radios sont utilisées à la place des ondes sonores (vitesse de propagation : $3 \times 10^8\,\mathrm{m\,s^{-1}}$). Les ondes radios utilisées ont l'avantage de ne pas être affectées par les conditions météorologiques ni les conditions d'éclairage de la scène. La figure 1.5 illustre quelques radars commerciaux disponibles.

FIGURE 1.4 – Capteurs ultrasoniques Bosch.

(a) Radar Continental ARS441

(b) Radar Bosch MRR (Mid-range Radar)

FIGURE 1.5 – Exemples de radars automobiles.

Plusieurs types de radars sont actuellement commercialisés et servent différents objectifs en fonction de leur distance d'opération comme le montre la figure 1.6.

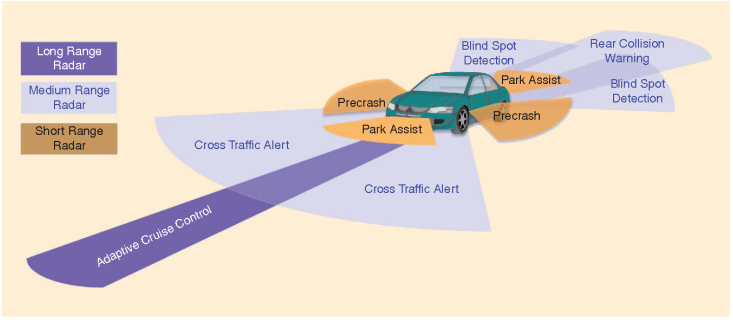


FIGURE 1. An ADAS consists of different range radars.

FIGURE 1.6 – Utilisation des radars automobiles (PATOLE et al., 2017)

Les radars sont souvent catégorisés en fonction de leur portée effective. Les radars à courte portée (environ 10 m) sont utilisés comme les capteurs à ultrasons pour les manœuvres à faible vitesse comme l'aide au stationnement. Les radars moyenne portée sont caractérisés par une portée d'environ 80 m et un angle de vision assez large (environ 80°). Disposés autour du véhicule, ces radars peuvent être utilisés pour des tâches telles que l'inspection des angles morts et l'analyse des carrefours. Les radars à longue portée ont une portée bien plus grande (environ 200 m) mais un angle de vision bien plus réduit. De ce fait, il est communément placé à l'avant du véhicule pour ce qui traite de la gestion de l'allure (*Adaptative Cruise Control*) et le freinage d'urgence sur les routes à vitesse élevée.

Caméra Les caméras RGB font partie des capteurs les plus étudiés. Peu coûteuses, elles permettent de récupérer des informations sur l'apparence des objets de la scène capturée. Il existe également des caméras sensibles à des domaines particuliers du spectre électromagnétique (par exemple les infrarouges, utiles pour détecter les sources de chaleur).

Les caméras les plus communes ont un champ de vision inférieur à 100°. Certains dispositifs tels que les caméras 360° permettent de visualiser l'intégralité d'une scène avec un seul capteur. Cependant, en raison de la faible disponibilité des données de ces capteurs et des déformations importantes présentes sur les images résultantes, ces dispositifs sont peu étudiés.

Les caméras de haute-définition permettent de distinguer des cibles sur des moyennes portées. S'agissant d'un capteur passif, c'est-à-dire n'émettant pas l'énergie qu'il mesure, le capteur est donc très sensible aux conditions lumineuses. Or, la lumière est très affectée par les conditions météorologiques telles que la pluie ou le brouillard, rendant le capteur moins efficace dans ces conditions.

LiDAR Le terme « LiDAR » est l'acronyme de « *Light Detection And Ranging* ». Contrairement aux capteurs ultrasoniques qui utilisent des ondes sonores et aux radars employant des ondes radio à faible longueur d'onde, les capteurs LiDAR utilisent comme médium des infrarouges de longueur d'onde environ 900 nm, proches de la lumière visible. Le principe reste fondé sur le temps de vol comme l'indique la figure 1.7.

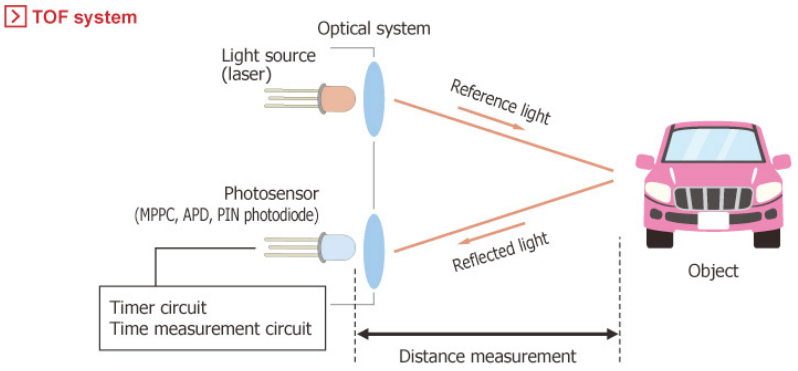


FIGURE 1.7 – Fonctionnement d'un capteur temps de vol (ici LiDAR).

En termes d'usage, le LiDAR et le radar partagent un certain nombre de points communs. Les deux capteurs se différencient sur deux points principaux :

- la résolution spatiale : les LiDAR fonctionnent à l'aide de lumière infrarouge collimatée dont la longueur d'onde est de l'ordre du nanomètre. D'une part le faisceau n'est pas diffus, d'autre part la précision du LiDAR grâce à sa longueur d'onde est très importante (erreur sur les distances de l'ordre du centimètre). Cela permet d'obtenir des représentations 3D très précises de l'environnement capturé. Les radars fonctionnent avec les ondes radios (longueur d'onde de 4 mm pour du 77 GHz), plus imprécis. Les détails dans une scène sont donc plus difficilement perceptibles par les capteurs radar ;
- la robustesse aux changements météorologiques : les performances des LiDAR sont en général dégradées sous des conditions météorologiques difficiles comme le brouillard ou la neige. Les ondes radios émises par les radars sont moins sensibles aux perturbations météorologiques.

Il existe plusieurs types de LiDAR, différenciés principalement par la manière dont ils balayent la scène pour obtenir la représentation en 3D. Le plus commun est le LiDAR à système de déflexion « mécanique ». Un faisceau laser est orienté par un système de miroirs dont la rotation est contrôlée par différents moteurs. La majorité des fabricants tels que Velodyne (*Produits Velodyne*) ou Ouster (*Produits Ouster*) proposent cette catégorie de LiDAR (Figure 1.8).

(a) Velodyne HDL-64E, HDL-32E, Puck (b) Ouster OS2, OS1, OS0

FIGURE 1.8 – Exemples de LiDAR mécaniques commerciaux.

La sensibilité des pièces mobiles au sein du capteur rendent ce dernier onéreux. Les capteurs dits « Solid-State » visent à réduire, voire à annuler, la présence de pièces mobiles. Nous pouvons notamment citer les LiDAR MEMS (Micro Electro Mechanical Systems) qui utilisent des miroirs dont l'orientation varie en fonction d'un stimulus tel qu'un courant, les LiDAR Flash qui illuminent en une seule fois la scène au lieu de la balayer ou les LiDAR OPA (Optical Phase Array). Le faisceau laser est réparti vers un ensemble de transmetteurs dont les phases sont individuellement contrôlées. L'orientation du front d'onde résultant dépend des phases sélectionnées.

La figure 1.9 explicite ces différents types de LiDAR.

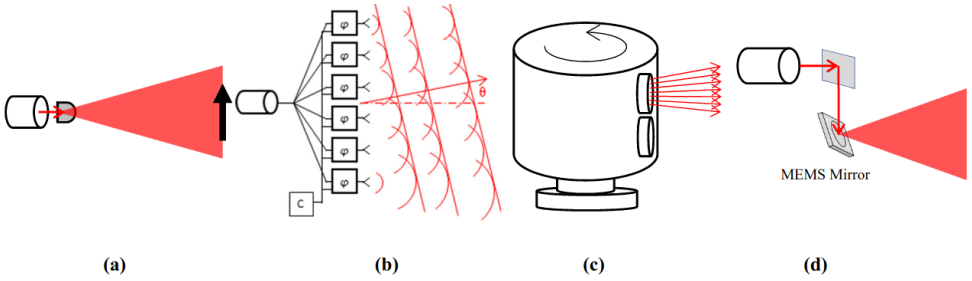


FIGURE 1.9 – Illustration des fonctionnements des différents types de LiDAR. (a) LiDAR Flash, (b) LiDAR OPA, (c) LiDAR mécanique, (d) LiDAR MEMS (image extraite de (WANG, WATKINS et XIE, 2020)).

Tableau récapitulatif des capteurs extéroceptifs Le tableau 1.1 résume les forces et faiblesses de chaque capteur sur plusieurs aspects tels que la portée, la résolution mais aussi la sensibilité aux conditions environnementales, le prix ou la nature de l'information retournée. D'autres critères relatifs à l'état du capteur, comme la présence de salissures sur les dispositifs optiques des caméras ou des LiDAR, pourraient être ajoutés. Nous n'avons retenu que des critères concernant des capteurs neufs et propres.

Aucun capteur n'est performant sur tous les aspects, d'où la nécessité de combiner plusieurs capteurs différents afin de bénéficier de leurs avantages et ainsi améliorer la perception du véhicule. Cette perception peut d'ailleurs revêtir plusieurs aspects qu'il est plus simple de traiter indépendamment.

TABLE 1.1 – Comparaison de capteurs extéroceptifs.

	Ultrasons	Radar	LiDAR	Caméra
Portée annoncée	5 m	200 m	100 m	100 m
Résolution en distance	Faible	Faible	Élevée	Faible
Sensibilité aux conditions météorologiques	Faible	Faible	Forte	Forte
Sensibilité aux conditions lumineuses	Faible	Faible	Faible	Forte
Prix	Faible	Faible	Élevé	Faible
Mesure directe de distance	Oui	Oui	Oui	Non
Sensibilité à l'apparence	Faible	Faible	Moyen	Fort

1.2.3 Aspects de perception

Le terme « perception » désigne l'ensemble des opérations réalisées à partir de données sensorielles pour obtenir une représentation des objets et de l'environnement dans lequel se situe le sujet. Lorsque l'on conduit, nous extrayons de notre vue un grand nombre d'informations permettant de prendre des décisions à la fois liées à la navigation et à la sécurité, la nôtre et celles des autres.

Simuler l'ensemble de ces informations au niveau informatique est un problème extrêmement complexe d'où la nécessité de décomposer la perception en plusieurs sous-problèmes, chacun ayant ses enjeux propres.

Nous pouvons, entre autres, citer les tâches de localisation dont l'objectif est de positionner l'agent mobile dans un contexte donné. Ce contexte peut prendre plusieurs échelles : position à l'échelle d'une carte préexistante, voie dans laquelle se situe l'ego-véhicule à l'échelle d'une route (KASMI et al., 2020), etc.

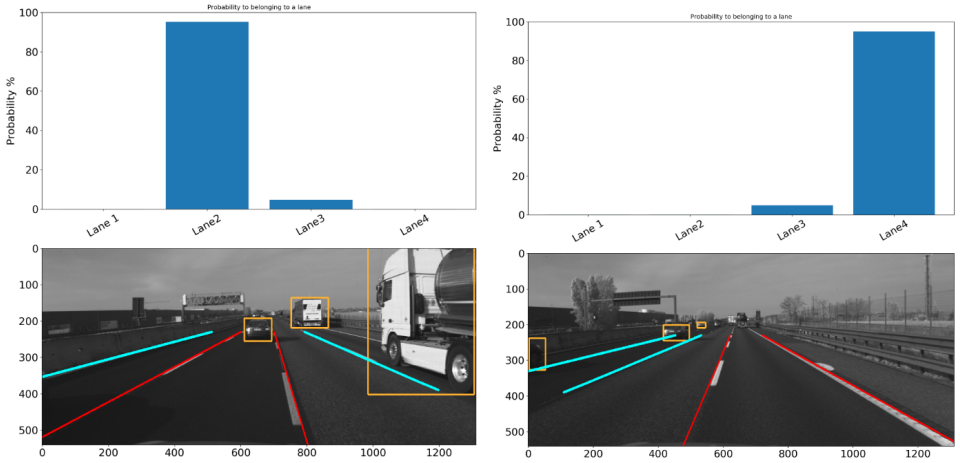


FIGURE 1.10 – Localisation au niveau des voies de circulation. En haut, les probabilités de chacune des voies d'être l'ego-voie. En bas, les boîtes orange désignent les résultats d'un détecteur 2D YOLO (REDMON et al., 2016 ; REDMON et FARHADI, 2017 ; REDMON et FARHADI, 2018), les lignes en cyan sont les lignes de marquage des autres voies, les lignes rouges les lignes de marquage de l'ego-voie. Image tirée de (KASMI et al., 2020).

En plus de connaître la position de l'ego-véhicule, la perception implique de connaître l'état de l'environnement proche. Cela inclut la segmentation sémantique, l'identification de zones

franchissables, la détection des signalisations horizontales et verticales ou encore la détection des usagers mobiles.

1.3 Contexte et objectifs de la thèse

1.3.1 Sujet et enjeux de l'étude

L'objectif des travaux menés dans cette thèse est d'étudier la complémentarité des données images provenant de caméras RGB et de nuages de points 3D issus de capteurs LiDAR pour des objectifs de détection de cibles liées à la conduite autonome. En effet, les capteurs caméras, de par leur disponibilité et leur capacité à fournir des données en couleur proche de ce que l'œil humain peut observer, ont bénéficié de nombreux travaux et études sur les tâches d'identification de cibles. Cependant, ces capteurs présentent des lacunes importantes vis-à-vis de la localisation précise de ces cibles dans l'espace. Les capteurs LiDAR, à l'inverse, fournissent des informations spatiales de haute précision, mais leur capacité à identifier des cibles décroît très rapidement avec la distance.

En raison de leurs performances, les rendant quasiment omniprésentes dans les tâches de perception, les techniques d'apprentissage machine sont au centre de l'étude. La contrepartie de ces performances repose sur l'utilisation d'un grand nombre de données pour ajuster les millions de paramètres constituant ces méthodes, mais également du matériel dédié pouvant exécuter ces implémentations.

Les algorithmes étant destinés à être implémentés sur des véhicules, l'une des contraintes principales visées dans ces travaux est la vitesse d'exécution. Un algorithme de perception trop lent est en effet inutile s'il ne suit pas le flux d'informations permettant de traiter un événement potentiellement critique et d'éviter une manœuvre dangereuse. Dans ces travaux, nous avons mis l'accent sur des architectures capables de traiter des flux capteurs, sans perte de données.

La détection 3D sur données LiDAR est un sujet fortement étudié, par les laboratoires et les entreprises travaillant sur le véhicule autonome. La grande majorité des travaux se focalisent sur la performance brute, jusqu'à atteindre des plafonds de performances dans les *benchmarks*. Les méthodes les mieux classées ne sont différenciées que par des variations de moins de 0.1% sur les scores. Aussi, les travaux se sont orientés vers la « flexibilité » du réseau, de son implémentation et de ses conditions d'exploitation plutôt que la performance pure. Cet aspect « flexibilité » se décline en deux points :

- caractère *anchor-free* : la grande majorité des méthodes de détection repose sur un fonctionnement fondé sur l'usage d'« ancres ». Il s'agit de boîtes a priori définies au préalable pour chaque classe d'obstacles visées. Le réseau doit dans un premier temps déterminer lesquelles de ces boîtes semblent être assez proches de cibles et, dans un second, appliquer une correction à ces ancres afin de les ajuster au mieux à la cible. Bien que cette approche donne de bons résultats, elle introduit un certain nombre d'hyperparamètres nécessaires qui peuvent grandement affecter l'entraînement et donc les résultats finaux. La variabilité intra-classe, c'est-à-dire les variations de formes, de couleurs, de dimensions des cibles au sein d'une même classe, est l'une des causes de l'impact de ces hyperparamètres. Parmi ces paramètres se trouvent notamment les dimensions des boîtes, leur placement dans l'espace de recherche, les seuils de recouvrement définissant si elles sont assez proches des cibles, etc. Les approches les plus récentes, qu'il s'agisse de détection 2D ou 3D, tentent de s'abstraire de ce paradigme, leur octroyant plus de souplesse ;
- adaptabilité vis-à-vis du capteur LiDAR : les capteurs LiDAR sont des capteurs encore très onéreux en comparaison des autres capteurs extéroceptifs. De plus la différence de répartition des points et la résolution peut grandement varier d'un modèle à un autre (Carballo et al., 2020b ; Carballo et al., 2020a). Si un capteur très différent de celui utilisé lors de l'entraînement est exploité, les performances du système de détection peuvent s'effondrer.

De nouvelles acquisitions et annotations, coûteuses en temps, risquent d'être nécessaire à chaque nouveau capteur. Relâcher cette contrainte permettrait d'exploiter un même système de détection à plusieurs projets.

Mes travaux de thèse se sont focalisés sur ces deux points. Au niveau industriel, ces aspects peuvent s'avérer cruciaux dans la mesure où une méthode de détection exempte des contraintes citées précédemment peut être ré-exploitée sur différents projets, réduisant ainsi les coûts ultérieurs de recherche et développement.

1.3.2 Collaboration Institut Pascal – LS2N – Sherpa Engineering

Cette thèse est le fruit d'une collaboration entre l'Institut Pascal, le laboratoire LS2N et la société Sherpa Engineering.

Sherpa Engineering

Sherpa-Engineering est une société de services basée à Nanterre spécialisée dans l'ingénierie basée modèle, la modélisation de systèmes complexes et le contrôle-commande. L'entreprise a œuvré dans un grand nombre de secteurs industriels tels que les domaines de l'énergie ou de l'aviation. L'entreprise propose diverses prestations ainsi que plusieurs produits sur étagère. Nous pouvons notamment citer les bibliothèques *PhiSim*, *PhiControl* et *PhiSystem* axées sur la modélisation et la simulation, ou encore une plateforme de simulation thermique pour la *Tesla Model 3*. Depuis quelques années, Sherpa Engineering travaille également sur les technologies d'assistance et d'aide à la conduite (ADAS : *Advanced Driver Assistance Systems*). Le sujet de recherche entre dans le contexte d'un projet de développement de l'entreprise qui souhaite se positionner sur le développement de briques technologiques à destination des véhicules autonomes, secteur actuellement en forte demande. Sherpa a déjà fait bénéficier de son expertise à plusieurs entreprises, dont des constructeurs automobiles.

Institut Pascal

L'Institut Pascal est un laboratoire pluridisciplinaire de l'Université Clermont Auvergne. Les travaux réalisés concernent trois domaines d'application principaux : l'hôpital, l'usine et les transports.

L'unité de recherche est divisées en 5 grands axes de recherche :

- Génie des Procédés, Énergétique et Bio-systèmes (GePEB) ;
- Image, Systèmes de Perception, Robotique (ISPR) ;
- Mécanique, Génie Mécanique, Génie Civil, Génie Industriel (M3G) ;
- Photonique, Ondes, Nanomatériaux (PHOTON) ;
- Thérapies Guidées par l'Image (TGI).

Les travaux de thèse entrent dans les thématiques proposées par l'axe « Image, Systèmes de Perception, Robotique » (ISPR). Cet axe se focalise sur les domaines de la perception et de la vision artificielle pour la commande de systèmes robotiques.

L'axe ISPR se divise en quatre équipes distinctes :

- l'équipe `ComSee`, focalisée sur les problématiques liées à la vision par ordinateur ;
- l'équipe `PerSyst` centrée sur les tâches de localisation, de cartographie et de navigation ;
- l'équipe `DREAM`, axant ses travaux de recherche sur le développement de capteurs intelligents ;
- l'équipe `MACCS` se focalisant sur l'aspect contrôle de systèmes robotiques complexes.

La thèse fut réalisée conjointement avec les équipes `ComSee` et `PerSyst` de l'Institut Pascal, les deux équipes ayant déjà œuvrées sur des problématiques visant la robotique mobile et le véhicule autonome. L'équipe `PerSyst` s'intéresse aux systèmes de perception et à l'exploitation de capteurs dédiés tels que les LiDAR pour des objectifs de cartographies ou de localisation. L'équipe `ComSee` concentre ses recherches sur le domaine de la vision par ordinateur, en particulier l'utilisation de capteurs non conventionnels (plénoptique, *event-based*) et les techniques d'apprentissage machine. Le sujet de recherche se situe donc à l'interface des domaines des deux équipes.

LS2N

Le Laboratoire des Sciences du Numérique de Nantes (LS2N) est une unité de recherche divisée en cinq pôles de recherches :

- Conception et Conduite des Systèmes (CCS) ;
- Robotique, Procédés, Calcul (RPC) ;
- Science des Données et de la Décision (SDD) ;
- Signaux, Images, Ergonomie et Langues (SIEL) ;
- Science du Logiciel et des Systèmes Distribués (SLS).

Au sein du pôle de recherche RPC, l'équipe ARMEN (Autonomie des Robots et Maîtrise des interactions avec l'ENvironnement) œuvre. Ses travaux englobent notamment des thématiques liées à la robotique mobile. Le véhicule autonome et ce qui se rapporte à ses capacités de perception font donc partie intégrante des thématiques traitées, d'où leur participation dans le projet.

1.4 Organisation du manuscrit

Le reste du manuscrit est organisé selon cinq autres chapitres :

- le **chapitre** 2 traite des différentes bases de données existantes dans le domaine de la détection d'obstacles mobiles pour le véhicule autonome. Les métriques utilisées pour évaluer les algorithmes de détection sont également décrites et discutées ;
- le **chapitre** 3 expose une revue de la littérature sur la détection d'objets 3D à l'aide de capteurs de distance. Les méthodes relatives à la conduite autonome constituent le cœur du chapitre ;
- le **chapitre** 4 présente la première contribution de la thèse, à savoir un détecteur de véhicules sur nuages de points LiDAR. L'originalité de la méthode provient de la manière dont sont représentés les véhicules sur la sortie du réseau, permettant une interprétation rapide de la situation de la route ;
- le **chapitre** 5 présente des techniques pour améliorer les performances de détecteurs sur des données issues de jeux de données différents de l'ensemble d'entraînement, ce sans hypothèse supplémentaire sur le domaine cible ;
- enfin, le **chapitre** 6 dévoile les conclusions et les perspectives de futurs travaux.

1.5 Contributions

Au cours de cette thèse, j'ai présenté mes activités au travers de différentes communications et publications mentionnées ci-dessous.

Conférences internationales avec comité de lecture

- Ruddy Théodose, Dieumet Denis, Christophe Blanc, Thierry Chateau, Paul Checchin « Vehicle Detection based on Deep Learning Heatmap Estimation » dans *2019 IEEE Intelligent Vehicles Symposium (IV)*
- Abderrahim Kasmi, Johann Laconte, Romuald Aufrère, Ruddy Théodose, Dieumet Denis, Roland Chapuis (2019) « An Information Driven Approach For Ego-Lane Detection Using Lidar And OpenStreetMap » dans *2020 16th IEEE International Conference on Control,Automation, Robotics and Vision (ICARCV)*

Conférences nationales avec comité de lecture

- Ruddy Théodose, Dieumet Denis, Christophe Blanc, Thierry Chateau, Paul Checchin « Détection de véhicules fondée sur l'estimation de carte de probabilités » dans *ORASIS 2019*

Revues internationales avec comité de lecture

- Ruddy Théodose, Dieumet Denis, Thierry Chateau, Vincent Frémont and Paul Checchin « A Deep Learning Approach for LiDAR Resolution-Agnostic Object Detection » dans *IEEE Transactions on Intelligent Transportation Systems (2021)*.

Après ces éléments de contexte liés à mon activité doctorale, dans le chapitre suivant, je présente la plupart des différents jeux de données que j'ai utilisés pour valider mes travaux.

Chapitre 2

Jeux de données pour la détection 3D de l'environnement à bord de véhicules et métriques pour l'évaluation des performances

Sommaire

2.1 Introduction

La détection d'obstacles à bord de véhicules est devenue un domaine très étudié notamment grâce aux techniques d'apprentissage profond. Cependant, ces algorithmes requièrent une phase d'entraînement afin d'ajuster leurs paramètres pour l'objectif visé. Cette phase d'entraînement exige un nombre conséquent de données annotées. Ainsi, de plus en plus de jeux de données contenant des acquisitions réalisées à l'aide de véhicules instrumentés sont publiées. Le nombre de méthodes proposées dans la littérature étant toujours croissant, certaines de ces bases sont accompagnées de *benchmarks* permettant de comparer les différents concurrents sur les mêmes données et les mêmes métriques.

Nous introduisons dans un premier temps les spécificités des nuages de points. Puis, la tâche de détection 3D est présentée. Nous listons ensuite les différents jeux de données utilisés au cours de nos travaux ainsi que leurs spécificités. Nous définissons également dans cette section les différentes métriques d'évaluation utilisées dans les *benchmarks*. Une discussion sur la création des annotations 3D ainsi qu'une description des métriques utilisées en détection 3D feront également l'objet de sections dédiées.

2.2 À propos des nuages de points 3D

Dans cette section, nous présentons succinctement les caractéristiques des nuages de points 3D.

Un nuage de points 3D est un ensemble fini et non ordonné de points repérés dans un espace à trois dimensions. Fournis par un capteur de distance, ces points permettent la représentation d'une scène dans l'espace euclidien 3D de $\mathbb{R}^3$.

Plus formellement, un nuage de points 3D P est donc défini par

$$P = \{p_i = (x_i, y_i, z_i)^T \in \mathbb{R}^3\}_{i<N} \tag{2.1}$$

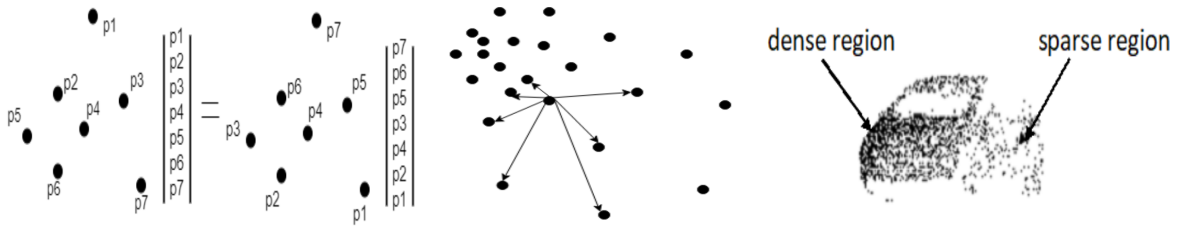
avec N étant le nombre de points dans le nuage.

Ces points 3D peuvent porter des informations supplémentaires telles que la couleur, la réflectance, la normale, etc. De ce fait, il est possible de définir des nuages de points « augmentés » :

$$P_F = \{(p_i, f_i) \mid p_i \in \mathbb{R}^3, f_i \in \mathbb{R}^D\}_{i<N} \tag{2.2}$$

avec f_i le vecteur représentant ces attributs supplémentaires, D le nombre de dimensions des informations supplémentaires transportées par les points 3D.

Malgré leur simplicité d'expression, les nuages de points disposent de propriétés qu'il convient de prendre en compte lorsque l'on définit des algorithmes de traitement de points. Ces propriétés sont illustrées sur la figure 2.1.



(a) Absence d'ordre (b) Absence de structure (c) Densité variable

FIGURE 2.1 – Propriétés des nuages de points. Images tirées de (BELLO et al., 2020).

L'une des propriétés de P est qu'il n'est pas ordonné, c'est-à-dire qu'une permutation sur les points engendre le même nuage de points du point de vue physique, mais deux nuages différents

du point de vue du stockage informatique. Par raccourci, on note souvent $P \in \mathbb{R}^{N\times 3}$. Le stockage et l'application nécessitent donc des fonctions que l'on qualifie de « symétriques » produisant les mêmes résultats quelle que soit la permutation utilisée.

La seconde propriété concerne le caractère discret et la non-uniformité des données. Les nuages de points sont irréguliers dans la mesure où les points ne sont pas distribués uniformément à travers l'espace. Certaines régions ont une densité élevée de points tandis que d'autres possèdent beaucoup moins de points, voire sont vides. Ce point soulève également la question de la représentation des espaces vides. En effet, les régions dépourvues de points sont majoritaires et pas nécessairement informatives. Les déclarer explicitement peut s'avérer lourd et coûteux.

Enfin, la troisième propriété dépend de la manière dont sont collectés les nuages de points. Si les seules informations spatiales disponibles sont les coordonnées cartésiennes des points 3D dans la scène, il est habituellement impossible d'ordonner ces points selon une structure régulière sans connaître la géométrie du dispositif d'acquisition qui peut être sphérique ou cylindrique par exemple. Contrairement à des données telles que les images qui sont des grilles de taille fixe dans lesquelles chaque pixel est localisé par une position discrète et entière, les points d'un nuage sont généralement repérés par des coordonnées réelles sans relation explicite de voisinages entre eux.

L'adaptation d'algorithmes dédiés à l'image pour des nuages de points est non triviale du fait de ces propriétés.

2.3 Description de la détection 3D

2.3.1 Réseaux de neurones en vision par ordinateur

Le domaine de la vision par ordinateur est large et complexe et difficilement représentable en un seul bloc. Aussi, la communauté scientifique a divisé la perception en plusieurs sous-catégories, parfois hiérarchiques, en fonction de l'objectif visé. Nous détaillons dans cette section les principales tâches étudiées dans la littérature dans lesquelles se sont illustrés les réseaux de neurones. La figure 2.2 donne un aperçu de différentes tâches de vision par ordinateur sur des images RGB.

(a) Classification (b) Segmentation sémantique (c) Détection 2D (d) Segmentation d'instances

FIGURE 2.2 – Principales tâches de vision par ordinateur sur images RGB (images issues de *Cours 11 de Li, Johnson et Yeung (Université de Stanford)*).

Les travaux menés au cours de cette thèse font appel à des données images et des nuages de points 3D. Les tâches décrites existent aussi bien pour les deux capteurs.

Uniquement dans cette section, nous définissons le terme *donnée* l'ensemble cohérent délivré par un capteur à un instant t et le terme *élément* le constituant atomique de la donnée. Ainsi, dans le cadre d'un capteur caméra, la donnée est l'image et l'élément est le pixel. De même, pour un

capteur LiDAR, la donnée est le nuage de points complet tandis que l'élément est le point 3D, de coordonnées (x, y, z).

Classification La classification consiste à attribuer un label à une *donnée*. Cette *donnée* représente en général un seul et unique objet. L'aspect spatial du contenu dans la donnée n'est pas considéré comme important.

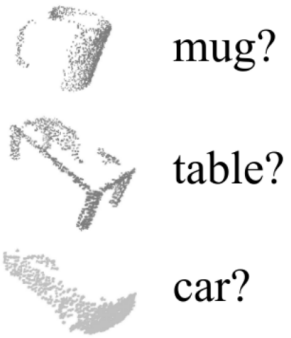
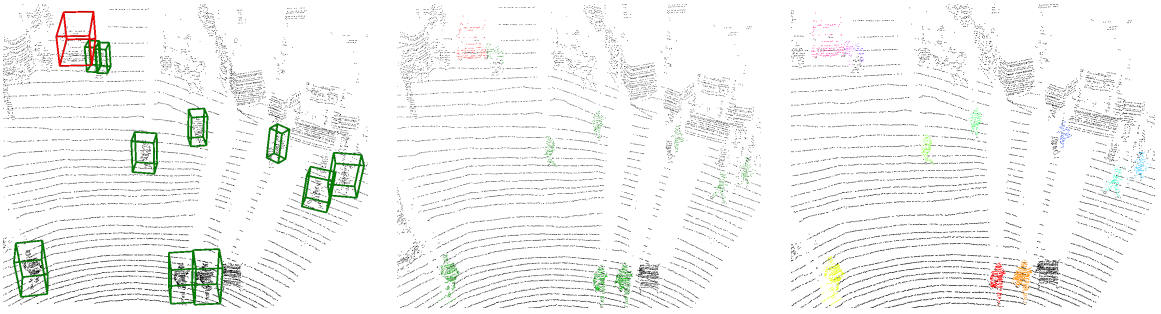


FIGURE 2.3 – Classification d'objets (QI et al., 2017a)

Détection La détection consiste à retrouver des objets dans une *donnée*. La *donnée* représente donc en général une scène complexe pouvant contenir un nombre non défini à l'avance d'objets (et peut être nul). Ces objets sont représentés par une classification sémantique (ce qu'ils représentent) et des bornes délimitant l'objet dans l'espace de recherche. En général, ces bornes sont représentées par une boîte englobante. Le contexte spatial est d'une grande importance et les objets d'une scène ne sont pas isolés les uns des autres.

Segmentation sémantique Contrairement à la classification qui vise à donner un label à une *donnée* complète, la segmentation cherche à attribuer un label à chaque *élément* de la *donnée*. On parle de segmentation sémantique lorsque le label visé concerne la catégorie sémantique de l'objet représenté par chaque *élément*.

Segmentation d'instances La segmentation d'instances est un hybride entre la détection et la segmentation. L'objectif visé est un partitionnement des *éléments* de la donnée en fonction des objets qu'ils représentent. Les labels attribués aux *éléments* peuvent ainsi être des identifiants uniques à chaque objet de la scène.



(a) Détection d'objets. Chaque couleur représente une classe

(b) Segmentation sémantique. Chaque couleur représente une classe

(c) Segmentation d'instances. Chaque couleur représente un seul élément de la scène

FIGURE 2.4 – Exemples de tâches à réaliser par traitement informatique.

Segmentation panoptique La segmentation panoptique (Kirillov et al., 2019) est un domaine récent qui mélange segmentation sémantique et segmentation d'instances. L'article originel distingue les objets dits *stuff* qui sont les objets indénombrables (ou sans limites spatiales définies tels que le ciel ou la chaussée) et les objets dits *things*, correspondant aux objets dénombrables de la scène. L'objectif est de réaliser une segmentation sémantique et, pour les *éléments* dont le label appartient aux *things*, de déterminer l'identifiant unique de la cible auquel il appartient.

Parmi toutes ces tâches, la détection d'objets fait partie des tâches les plus étudiées pour la conduite autonome. Nous détaillons dans le paragraphe suivant les raisons de cette tendance.

2.3.2 Intérêts de la détection BEV et 3D pour la conduite autonome

On distingue trois « types » de détection dans la littérature :

- la détection 2D. Il s'agit de la détection la plus connue. Les entrées sont des images issues de caméras. Les boîtes englobantes sont définies dans le plan de l'image et sont représentées comme des rectangles dont les arêtes sont alignées sur les axes (*Axis Aligned Bounding Boxes* ou AABB). Les deux représentations principales sont généralement décrites par 4 paramètres, (x_1, y_1, x_2, y_2) pour les coordonnées des coins opposés ou (x_c, y_c, w, h) pour la position du centre et les dimensions du rectangle ;
- la détection 3D. Dans la grande majorité des cas, les entrées sont des nuages de points 3D. Les boîtes englobantes sont des pavés orientés positionnés dans l'espace. La position est définie selon 3 paramètres (x, y, z) et les dimensions également (longueur, largeur, hauteur). En fonction des annotations disponibles, les orientations peuvent être libres (3 pour des angles d'Euler, 4 pour des quaternions) ou contraintes, par exemple sur l'axe vertical (1 seul paramètre) ;
- la détection BEV. Il s'agit d'une version simplifiée de la détection 3D. Les entrées sont similaires. Les boîtes englobantes sont des rectangles orientés (*Oriented Bounding Boxes* ou OBB) définies dans une vue de dessus orthographique. La principale hypothèse motivant ce type de détection est la supposition que tous les objets sont situés sur un même plan horizontal. Bien que les coordonnées x, y, z soient utilisées en entrée, les boîtes produites par le système de détection n'incluent pas de paramètres relatifs à la verticalité ; seuls cinq paramètres sont estimés : x, y, largeur, longueur et orientation selon l'axe vertical.

Les bases de données pour l'automobile contenant des données LiDAR se focalisent principalement sur la détection 3D. L'une des raisons en est que le capteur LiDAR fournit une mesure directe des distances, permettant la représentation précise de la scène dans lequel l'ego-véhicule se meut. Une détection 3D fournit directement la distance entre l'ego-véhicule et un obstacle ainsi que l'espace occupé par cet obstacle. De plus, une scène de conduite est rarement statique. Au niveau applicatif, obtenir la boîte englobante de l'obstacle permet l'application de méthodes dites de « suivi par détection » permettant l'acquisition d'informations temporelles telles que la trajectoire ou la vitesse. Ces informations sont cruciales pour des algorithmes de planification. En outre, il est préférable lors de la conduite de considérer un obstacle dans son entièreté plutôt que par les éléments individuels renvoyés par le capteur. Par exemple, il semble plus cohérent d'attribuer une vitesse identique à tous les *éléments* représentant un même objet plutôt que d'estimer les variables pour chaque point, au risque d'obtenir des estimations contradictoires sur un même objet.

La segmentation sémantique du nuage de points est importante lorsque l'on souhaite déterminer les zones dans lesquelles l'ego-véhicule peut circuler. Par exemple, bien qu'appartenant tous les deux au sol, la distinction entre le trottoir et la chaussée est primordiale à la sécurité de l'ego-véhicule, mais aussi au bien-être des autres usagers présents dans la scène.

2.4 Bases de données et métriques

Il existe actuellement un grand nombre de bases de données contenant des nuages de points 3D, et ce, pour différents usages et domaines d'activités. Certaines bases sont destinées à des domaines tels que l'analyse fine de scènes ou la topographie. Elles nécessitent un très grand nombre de points acquis avec une haute précision afin de récupérer un maximum d'informations sur la scène numérisée. Les nuages de points de ces bases sont notamment acquis par accumulation de prises de vues puis post-traitements ou bien par un capteur extrêmement précis et disposant d'un long temps d'acquisition.

Nous présentons dans cette section quelques bases uniquement dédiées aux tâches relatives à la conduite autonome qui correspondent plus aux conditions et contraintes d'exploitation recherchées. Dans le cas de la conduite autonome, les scènes sont hautement dynamiques. D'une part les données doivent être acquises à fréquence élevée, d'autre part, les algorithmes devant les traiter doivent être au moins aussi rapides que les capteurs pour détecter rapidement les événements critiques et permettre une réaction appropriée. Au niveau des données, cela se traduit par des nuages de points 3D bien plus épars et lacunaires du fait de l'absence de post-traitements comme l'illustre la figure 2.5 sur laquelle on peut observer l'analyse sémantique d'une scène numérisée en 3D de manière dense d'une part et, d'autre part, une représentation du nuage de points acquis en une révolution par un capteur Velodyne.

Dans les pages suivantes, les différents jeux de données qui ont été utilisés durant cette thèse sont introduits en commençant par la base KITTI puis nuScenes et Pandaset. Nous présentons également deux jeux de données non annotés utilisés pour la validation de nos contributions : Ford AV Dataset et UTBM Robocar. Les dispositifs d'acquisition sont rappelés ainsi que l'ensemble des données collectées ou ajoutées pour une évaluation des performances des algorithmes. D'autres bases sont également mentionnées pour établir un panorama relativement complet, mais non exhaustif.

2.4.1 KITTI

Description de la base

La base KITTI (Geiger et al., 2013) a été l'une des premières bases publiques d'envergure accessible aux chercheurs. Le premier déploiement de la base a eu lieu en 2012. La base est constituée d'enregistrements réalisés dans la ville de Karlsruhe en Allemagne.

Le véhicule instrumenté utilisé est équipé de :

- 2 caméras en nuances de gris `PointGray Flea2 FL2−14S3M−C`;
- 2 caméras couleurs `PointGray Flea2 FL2−14S3C−C`;
- 1 LiDAR Velodyne HDL-64E, LiDAR mécanique 64 nappes, cadencé à 10 Hz;
- 1 GPS RTK/Centrale Inertielle OXTS RT3003.

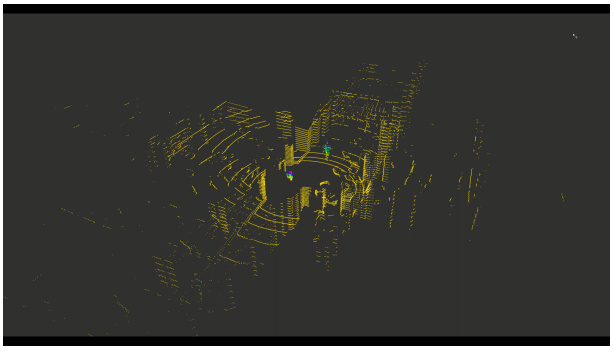
La figure 2.6 explicite le positionnement des différents capteurs sur le véhicule.

Toutes les scènes sont enregistrées de jour et par beau temps (cf. Figure 2.7). L'éclairage peut cependant fortement varier (surexposition, faible luminosité en zone urbaine, etc.).

Les données acquises ont été annotées pour plusieurs tâches relatives à la conduite autonome telle que l'odométrie, l'estimation du flot optique ou la complétion de profondeur. Chacune de ces tâches est accompagnée d'un *benchmark* permettant la comparaison de différents algorithmes selon un jeu de données de test commun.

Dans le cadre de cette thèse, nous nous focalisons sur les tâches de détection BEV. Les données utilisables pour cette tâche sont scindées en deux sous-ensembles : un ensemble d'entraînement constitué de 7481 échantillons et un ensemble de test utilisable dans le cadre d'une évaluation sur le *benchmark* officiel constitué de 7518 échantillons. Le terme « échantillon » est utilisé pour représenter un ensemble de données synchronisées à un instant t ou sur une courte période dt.

(a) Nuage de points denses post-traité issu de la base Paris-Lille 3D (Roynard, Deschaud et Goulette, 2018)



(b) Nuage de points épars issu de la base BigRedLiDAR (*Dataset BigRedLiDAR*)

Figure 2.5 – Comparaisons de densités de nuages de points.

Sur KITTI Object3D, un échantillon contient les images des caméras, le nuage de points à 360° du LiDAR et les données de calibrage. Si l'échantillon provient de l'ensemble d'entraînement, les annotations sont également présentes. Bien que le nuage représente la scène à 360°, les annotations ne sont présentes que dans le champ de vision de la caméra couleur, étiquetée « *02* », ce qui peut conduire à des incohérences car des obstacles existants dans le nuage de points LiDAR, mais absents de l'image, sont considérés comme non-existants par la base.

Représentation des obstacles

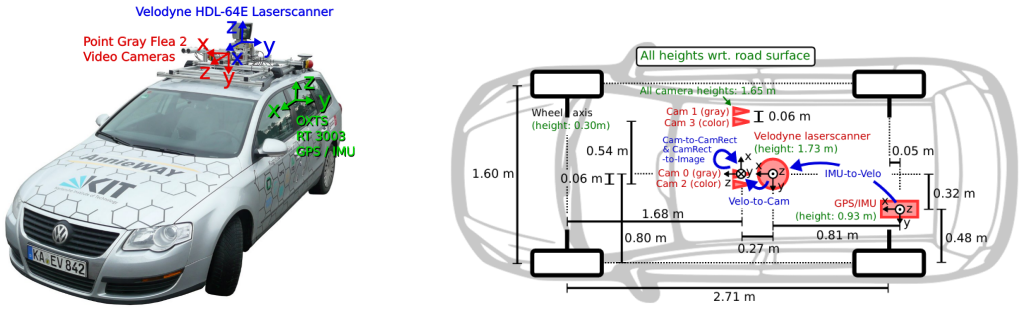
Chaque obstacle de la base est défini par la catégorie à laquelle il appartient et par une boîte englobante. Cette boîte est définie suivant sept paramètres illustrés dans la figure 2.8 :

- sa position (x, y, z) ;
- ses dimensions (h, w, l) ;
- son orientation θ. Les obstacles sont considérés comme étant posés sur un sol horizontal. θ est donc un scalaire représentant le cap de l'obstacle.

La base KITTI définit huit classes d'obstacles [1]. Cependant, le *benchmark* ne repose que sur trois d'entre elles : *Car*, *Pedestrian* et *Cyclist*.

Pour l'évaluation, les concepteurs de la base ont défini trois difficultés décrites dans le tableau 2.1. Ces dernières sont définies selon des paramètres de visibilité dans l'image : hauteur

1. Pour identifier les différentes classes, j'ai choisi de garder leur dénomination en langue anglaise et de les typographier en italique. Le lecteur, familier ou non avec ces jeux de données, peut ainsi plus facilement associer nos résultats à ceux d'autres chercheurs.



(a) Volkswagen Passat instrumentée

(b) Position et axes des capteurs

FIGURE 2.6 – Véhicule instrumenté en (a) utilisé pour les enregistrements KITTI, sur (b) la disposition des capteurs. Source : GEIGER et al., 2013

maximale de la boîte 2D en pixels, niveau d'occultation (causé par un autre objet), pourcentage de troncation (objet « coupé » car aux bords de l'image). Il s'agit donc de difficultés définies dans le repère image mais qui ne reflètent pas totalement les difficultés liées au nuage de points (distance au capteur, nombre de points appartenant à chaque annotation, etc.). Par conséquent, elles sont difficilement transposables aux autres bases de données car elles dépendent grandement de la caméra utilisée.

TABLE 2.1 – Difficultés sur KITTI.

Difficulté	Conditions
Easy	Hauteur de la boîte 2D > 40 pixels, complètement visible, pourcentage de troncation < 15%
Moderate	Hauteur de la boîte 2D > 25 pixels, partiellement visible, pourcentage de troncation < 30%
Hard	Hauteur de la boîte 2D > 25 pixels, difficilement visible, pourcentage de troncation < 50%

2.4.2 nuScenes

Description de la base

nuScenes (CAESAR et al., 2019) est une base de données accessible publiquement depuis 2019. Le jeu de données a été enregistré dans les villes de Boston et de Singapour et est axé autour des tâches de détection, de suivi et de prédiction de trajectoire des obstacles mobiles.

Le véhicule instrumenté pour réaliser les acquisitions est équipé de :

- 6 caméras RGB ;
- 1 LiDAR mécanique 32 nappes ;
- 5 radars automobiles à 77 GHz ;
- un GPS et une IMU.

Les références exactes des capteurs ne sont pas précisées. Le placement des capteurs est explicité dans la figure 2.9.

Les six caméras sont positionnées et orientées de sorte à capturer la scène sur 360° comme le montre la figure 2.10.

Le site de la base propose deux sous-ensembles, une base dite « Mini » et la base complète. La base « Mini » contient 10 scènes tandis que la base complète contient 1000 scènes, dont 850 destinées à l'entraînement et 150 pour l'évaluation. Chaque scène représente 20 s d'acquisitions, enregistrées dans les deux villes suivant plusieurs conditions météorologiques et lumineuses (pluie, nuit, etc.).

FIGURE 2.7 – Exemples de scènes capturées sur KITTI. Les captures ont été enregistrées dans des environnements variés (route, zone résidentielle, centre-ville …) en plein journée.

Concernant le *benchmark* de détection, les annotations sont réalisées sur 2 Hz soit toutes les 0.5 s. Cependant, les acquisitions intermédiaires aux annotations appelées « *sweeps* » sont également présentes, permettant ainsi l'accumulation de plusieurs nuages de points. Cette méthode est recommandée dans la mesure où les nuages de points capturés sont très épars contrairement à ceux capturés par KITTI. Dans le cas d'approches multimodales, cela pose une problématique supplémentaire, car contrairement aux données annotées, les *sweeps* de différents capteurs ne sont pas synchronisés.

Représentation des obstacles

Les cibles, qu'il s'agisse d'objets potentiellement mobiles ou d'infrastructures fixes, sont annotées suivant 23 classes. Toutefois, certaines classes d'objets sont très rares tandis que d'autres se ressemblent. De ce fait, le challenge de détection d'obstacles ne se focalise que sur 10 classes : *barrier*, *bicycle*, *bus*, *car*, *construction_vehicle*, *motorcycle*, *pedestrian*, *traffic_cone*, *trailer*, *truck*.

La base nuScenes se veut être une base pour de l'analyse en contexte dynamique. Chaque cible de la base est caractérisée par :

- sa classe : elle correspond à la catégorie sémantique de la cible ;
- ses attributs : pour certaines classes, les attributs apportent une information supplémentaire sur l'état de la cible. La classe *cycle* dispose par exemple d'un attribut *with_rider* et d'un autre attribut *without_rider*. Toutes les classes ne possèdent pas forcément d'attributs ;
- les paramètres de sa boîte englobante : une boîte est définie par sa position, ses dimensions et son orientation. nuScenes prend le parti de représenter l'orientation par des quaternions. La rotation est donc totalement libre contrairement à la base KITTI qui la restreint à l'axe vertical. L'orientation des cibles de la classe *cone* est ignorée ;

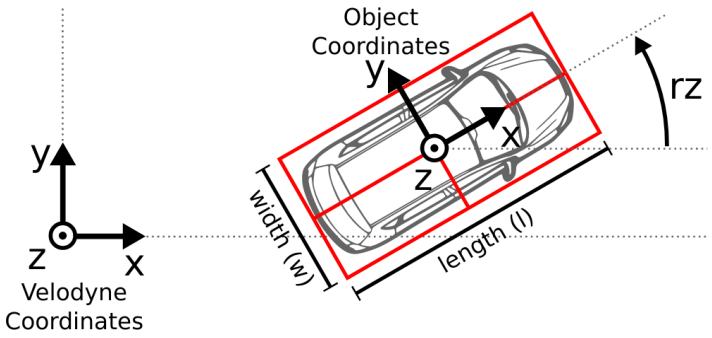


FIGURE 2.8 – Représentation des obstacles dans la base KITTI dans le repère du LiDAR (l'angle θ est nommé *rz*).

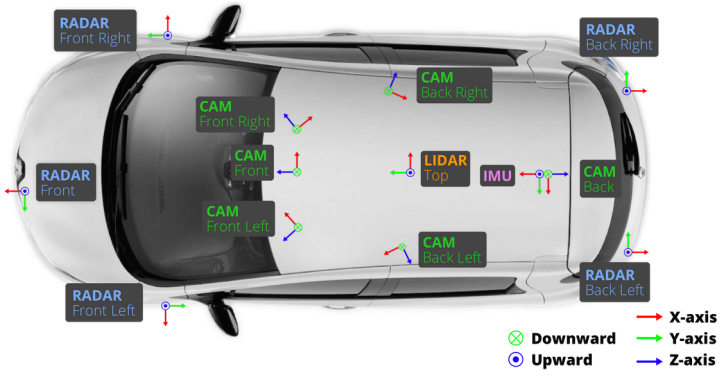


FIGURE 2.9 – Placement des capteurs sur le véhicule utilisé pour la construction de nuScenes (CAESAR et al., 2019).

– la vitesse : la base nuScenes repose sur l'accumulation de plusieurs acquisitions sur le temps. Il est alors possible d'estimer la vitesse de la cible en exploitant les nuages antérieurs. La vitesse des cibles de classes *barrier* et *traffic_cone* est ignorée.

2.4.3 Pandaset

Description de la base

La base Pandaset est une base de donnée accessible depuis 2020. Tout comme les autres bases récentes, la base propose des détections sur 360°. Contrairement aux autres bases qui sont soit uniquement dédiées à la recherche, soit nécessite une licence auprès des concepteurs de la base, la base Pandaset permet une utilisation commerciale sans frais supplémentaires.

Le véhicule est équipé de :

– 1 caméra frontale longue focale à 10 Hz ;
– 5 caméras grand angle 10 Hz ;
– 1 LiDAR mécanique Hesai Pandar64, 360°, 10 Hz ;
– 1 LiDAR Solid-State frontal PandarGT, 10 Hz
– un GPS/IMU ;

Le placement et les champs d'acquisition des capteurs sont indiqués sur la figure 2.12.

La base propose une centaine de scènes capturées dans l'état de Californie, soit à San Francisco, soit sur la route *El Camino Real* entre les villes de San Mateo et Palo Alto. Chaque scène représente une acquisition de 8 s à 10 Hz soit 80 acquisitions par scène. Au total, 8240 scènes d'entraînement

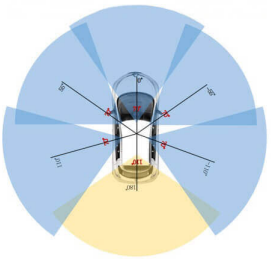


FIGURE 2.10 – Orientation et champs de vision des caméras sur nuScenes. Source : *nuScenes*.

FIGURE 2.11 – Exemples de scènes capturées sur nuScenes. Les environnements capturés sont essentiellement urbains, mais enregistrés à différents moments de la journée.

sont présentes dans la base (cf. Figure 2.12). Aucune séparation entre données d'entraînement et données de test n'est initialement définie, cette séparation étant à la charge de l'utilisateur.

Représentation des obstacles

Au niveau des annotations, celles-ci sont définies tout autour du véhicule et adoptent un format hybride à KITTI et nuScenes. Concernant la boîte englobante, chaque obstacle est défini par une position (x, y, z), des dimensions (dx, dy, dz) correspondant respectivement à la largeur, la longueur et la hauteur, et une orientation θ correspondant à l'angle de lacet, équivalent ici à l'angle de la rotation d'axe $\boldsymbol{z}$. Pour les autres caractéristiques, un obstacle est également défini par sa catégorie sémantique ainsi que certains attributs spécifiques à certaines classes. Par exemple, pour les piétons, des attributs *Sitting*, *Lying*, *Walking* et *Standing* ont été créés pour désigner la pose et l'activité du piéton. La base indique également le capteur LiDAR et/ou la caméra utilisée pour définir chaque annotation.

(a) Caméra frontale (b) Caméras grand angle (c) LiDAR Hesai Pandar64 (d) LiDAR PandarGT

FIGURE 2.12 – Positionnement et champs de vision des capteurs sur le véhicule d'acquisition Pandaset. Source : *Pandaset by Hesai and Scale*

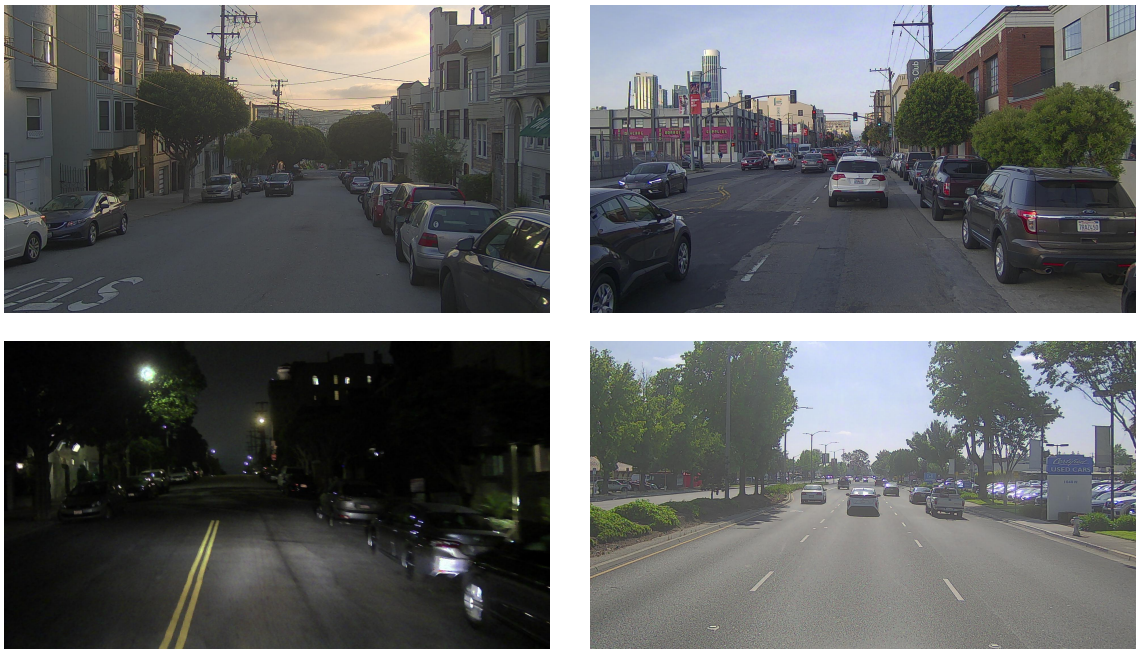


FIGURE 2.13 – Exemples d'acquisitions sur Pandaset. Les environnements sont essentiellement urbains et capturés à diverses heures de la journée.

2.4.4 Ford AV Dataset

Le jeu de données Ford Autonomous Vehicule Dataset (AGARWAL et al., 2020) est une base principalement dédiée à l'estimation de pose et à l'odométrie. Le jeu de données contient notamment des nuages de points accumulés permettant de décrire précisément l'environnement. Néanmoins, aucune annotation relative à la détection d'obstacles n'est présente. Le véhicule est équipé de :

- 6 caméras Pointgrey 1.3MP ;
- 1 caméra Pointgrey 5MP située au niveau du rétroviseur intérieur ;
- 4 LiDARs mécaniques 32 nappes Velodyne HDL-32E 10 Hz ;
- un GPS/IMU ;

Le placement et les champs d'acquisition des capteurs sont indiqués sur la figure 2.14. Les scènes de la base ont été enregistrées à Detroit, USA, de jour à différentes saisons.

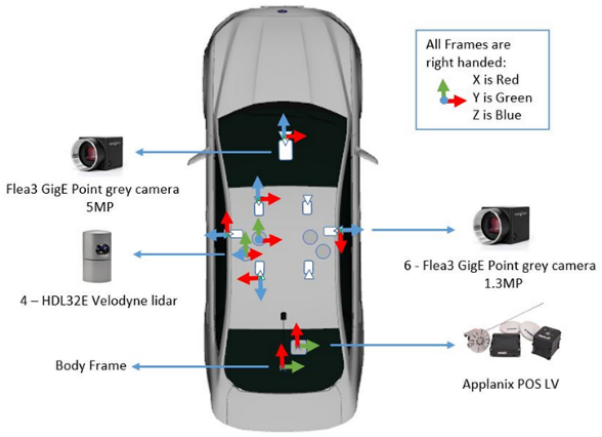


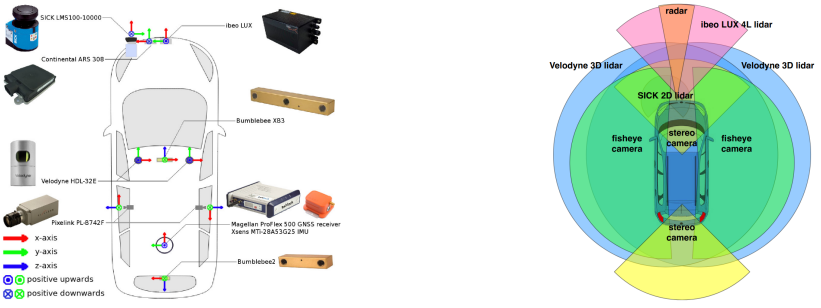
FIGURE 2.14 – Placement des capteurs sur le véhicule Ford (AGARWAL et al., 2020).

2.4.5 UTBM Robocar

Tout comme Ford AV Dataset, le jeu de données UTBM Robocar (YAN et al., 2020) est principalement destiné à l'estimation de pose de l'ego-véhicule et l'odométrie par LiDAR. Ce jeu de données ne dispose pas non plus d'annotations pour la détection 3D. Le véhicule est équipé de :

- 1 caméra stéréo Teledyne FLIR Bumblebee XB3 orientée vers l'avant ;
- 1 caméra stéréo Teledyne FLIR Bumblebee 2 à l'arrière ;
- 2 caméras équipées de lentilles fisheye ;
- 1 LIDAR 4 nappes Ibeo LUX 4L ;
- 1 LiDAR plan SICK LMS100−10000 ;
- 2 LiDARs 32 nappes Velodyne HDL-32E 10 Hz ;
- un GPS/IMU ;

Le placement et le champ de vision des capteurs sont indiqués sur la figure 2.15.



(a) Placement des capteurs

(b) Champ de vision des capteurs

FIGURE 2.15 – Capteurs sur le véhicule UTBM (YAN et al., 2020).

2.4.6 Autres jeux de données pour la conduite autonome

Dans cette section, nous présentons succinctement d'autres bases de données définies pour l'analyse de scènes pour la conduite autonome. Ces jeux de données ne seront cependant pas exploités dans ces travaux.

Waymo

La base Waymo (Sun et al., 2020), accessible depuis 2019, se focalise sur les tâches de détection et de suivi 2D et 3D. Le véhicule est équipé de cinq LiDAR dont un LiDAR 360° situé sur le toit du véhicule, de cinq caméras (Figure 2.16). La base contient 1150 scènes de 8 s chacune. Les nuages de points de chaque LiDAR ont la particularité d'être encodés sous la forme d'images de profondeur. La position de chaque pixel dans l'image correspond à l'inclinaison et à l'azimut du point 3D correspondant dans un repère sphérique dont l'origine se situe au centre du LiDAR. Les canaux encodent différentes informations telles que la distance du point au centre du capteur ou l'intensité du point 3D.

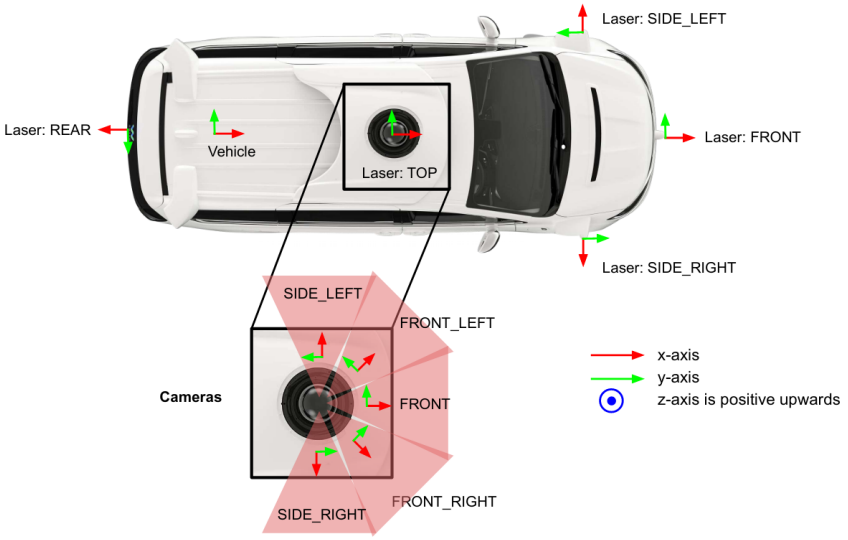


Figure 2.16 – Placement des capteurs sur le véhicule Waymo (Sun et al., 2020).

Cette base dispose de plusieurs *benchmarks* permettant la comparaison de différents algorithmes sur plusieurs tâches. Dans le cas de la détection 3D, les annotations regroupent 3 classes d'objets : voitures, piétons cyclistes. L'assiette des objets est considérée comme nulle. Ainsi, les boîtes englobantes sont représentées par leur position (x, y, z), leurs dimensions (h, w, l) et l'orientation par rapport à l'axe vertical θ.

KITTI-360

La base KITTI-360 (Xie et al., 2016 ; Liao, Xie et Geiger, 2021) est une base construite par les chercheurs ayant développé la base KITTI originale. Elle met davantage l'accent sur les taches de segmentation (sémantique et instance) sur les nuages de points, en plus de proposer des ressources pour la détection 3D. Le véhicule est équipé de :

- 2 caméras fisheye 180° ;
- 1 LiDAR Velodyne HDL-64E, LiDAR mécanique 64 nappes ;
- 1 LiDAR SICK LMS-200 ;
- 2 caméras perspectives en stéréo ;
- une IMU/GPS.

Leur placement sur le véhicule est détaillé dans la figure 2.17 et la figure 2.18 illustre des données capteurs issues de la base. Les classes d'objets considérées pour la détection 3D sont la classe *Car* et la classe *Building*. L'assiette des cibles n'est pas prise en compte, les boîtes englobantes sont ainsi décrites avec les mêmes paramètres que ceux utilisés dans la base KITTI.

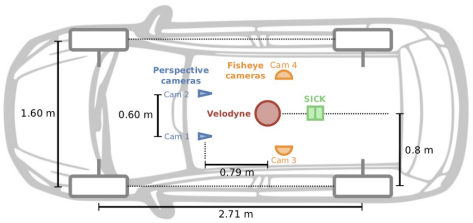
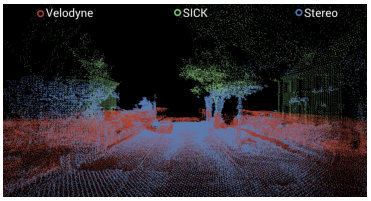


FIGURE 2.17 – Placement des capteurs sur le véhicule KITTI-360 (LIAO, XIE et GEIGER, 2021).

(a) Caméras Frontales



(b) Nuages de points issus des différents capteurs

(c) Caméras fisheye

FIGURE 2.18 – Exemple de données capteur présentes dans la base KITTI-360 (LIAO, XIE et GEIGER, 2021)

Apolloscape

La base Apolloscape (HUANG et al., 2018 ; HUANG et al., 2019) est une base construite par l'entreprise Baidu. Comme KITTI, la base propose des *benchmarks* sur plusieurs tâches telles que l'estimation de trajectoire, la détection et le suivi d'obstacles ou la détection de lignes au sol. Le véhicule utilisé est illustré sur la figure 2.19 et est équipé de :

- 1 système Riegl VMX-1HA constitué de 2 LiDARs VUX-1HA ;
- 1 système Riegl VMX-CS6 constitué de 2 caméras ;
- une IMU/GNSS.

FIGURE 2.19 – Véhicule Apolloscape (HUANG et al., 2019)

Dans les paragraphes suivants, nous présentons les outils utilisés pour la quantification des performances des algorithmes de détection à partir des annotations présentées précédemment.

2.5 Discussion sur les métriques d'évaluation

2.5.1 Constitution d'une vérité de terrain et limites des annotations

La constitution d'une vérité de terrain sur des nuages de points est difficile et chronophage. D'une part, contrairement à la détection 2D sur images pour laquelle les cibles sont en général définies par 4 paramètres (coordonnées du centre + largeur + hauteur ou coordonnées des points extrêmes), les cibles pour la détection 3D comportent plus de paramètres, à savoir les trois coordonnées du centre, les trois dimensions et l'orientation, cette dernière pouvant avoir plusieurs formulations. Sur KITTI, l'orientation est définie comme la rotation autour de l'axe vertical donc un scalaire tandis que sur nuScenes, l'orientation est définie par un quaternion, permettant de représenter les rotations sur les trois axes. Or, l'écran d'ordinateur ne pouvant afficher qu'une vue partielle du nuage de points, l'annotateur devra en permanence changer de points de vue pour vérifier si les annotations placées sont correctes et pertinentes. La figure 2.20 illustre un exemple de logiciel d'annotation utilisé pour créer des vérités de terrain.

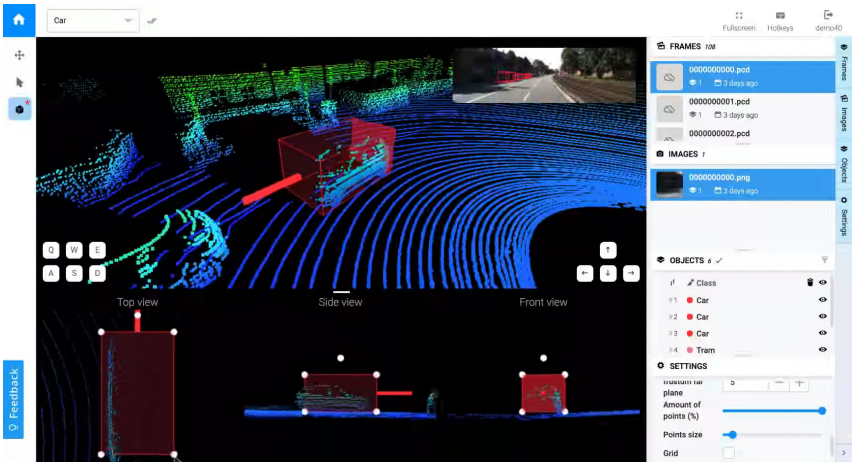
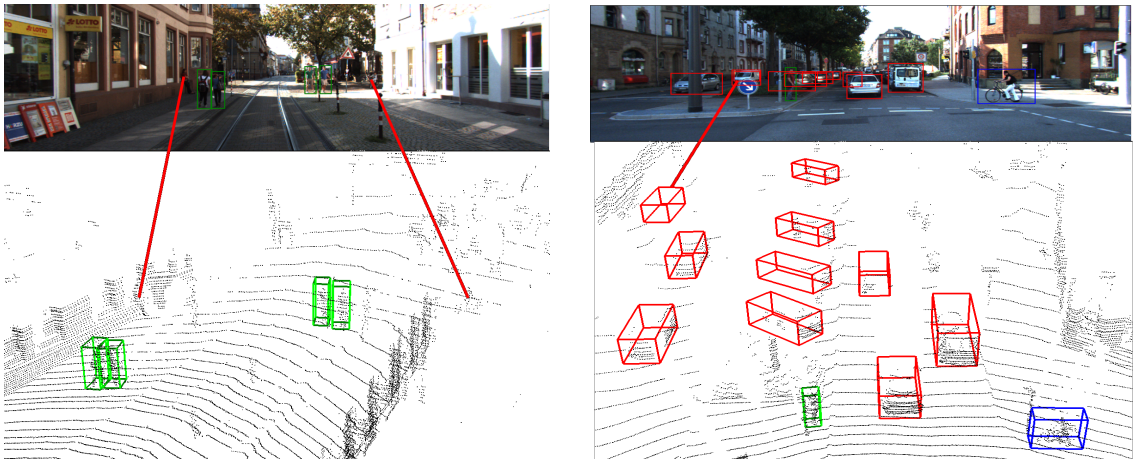


FIGURE 2.20 – Exemple de logiciel d'annotation de nuage de points pour la détection 3D (*Outil d'annotation Supervisely*).

Les enregistrements étant des séquences, il est possible d'exploiter les acquisitions précédentes afin de reporter les annotations d'une image à l'autre, par exemple à l'aide de techniques de *tracking* ou d'interpolations. Cependant, les techniques d'automatisation doivent être vérifiées a posteriori au risque de générer des incohérences comme celles indiquées sur la figure 2.21.

Ces erreurs doivent être le moins présentes possibles. En effet, les scores de performances se servent de ces annotations en tant que références. L'algorithme obtenant le meilleur score est donc celui qui arrive à reproduire au mieux les biais de l'annotation.

Un biais souvent constaté dans les bases exploitant l'image concerne le champ de vision. Dans des bases telles que KITTI, les annotations fournies n'existent que dans le champ de vision de la caméra frontale. De ce fait, lors du traitement d'un nuage de points, toutes les zones qui n'apparaissent pas dans le champ de vision sont considérées comme des zones négatives, même si des cibles existent en zone périphérique. Se pose alors la question de l'exclusion des zones périphériques. Si seuls les points appartenant au champ de vision sont conservés, les détections deviendront plus difficiles aux limites de champ de vision du fait de la troncature. Si tout le nuage est inclus, le réseau risque de sur-apprendre des dépendances spatiales, forçant alors l'utilisation de prétraitement lors de l'inférence.



(a) Cibles existantes, mais non annotées

(b) Cibles invisibles sur le nuage de points (aucun point 3D dans la boîte)

FIGURE 2.21 – Exemples d'erreurs possibles dans les vérités de terrain.

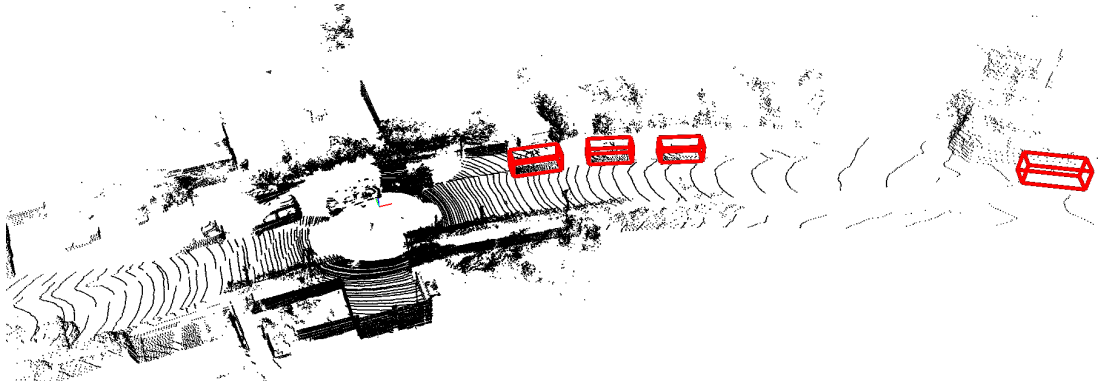


FIGURE 2.22 – Différence entre annotations et données. Ici, les vérités de terrain ne sont définies qu'à l'avant de l'ego-véhicule.

2.5.2 Métrique de référence : Average Precision

Dans le domaine de la détection d'objets (générique ou ciblé pour un domaine précis), l'*Average Précision* est la métrique la plus communément utilisée pour quantifier la performance des algorithmes. À l'origine, cette métrique est principalement employée dans le cadre de systèmes fournissant des résultats classés par un certain ordre. On peut notamment citer le domaine de la recherche d'information : pour une requête donnée, un moteur de recherche attribue une valeur de pertinence aux documents d'une base et produit les résultats dans l'ordre décroissant.

Les paradigmes actuels de système de détection définissent une sortie comme étant un volume fermé de position définie, généralement des boîtes, associée à un score de prédiction.

Nous explicitons dans les paragraphes suivants la manière dont est calculée l'*Average Precision* pour la détection d'objets.

D'abord, il faut définir ce que l'on appelle une fonction d'évaluation. Cette fonction est chargée de définir la pertinence d'un résultat. Pour reprendre l'exemple du moteur de recherche, si l'on dispose d'un document cible, d'une requête associée, cette fonction permet de déterminer si l'un des résultats renvoyés par le moteur de recherche correspond au document cible ou non.

Les systèmes de détection reposant sur des surfaces ou des volumes, cette fonction d'évaluation est souvent définie comme la superposition entre une prédiction et une boîte de référence

que l'on qualifiera de « vérité de terrain » (*Ground Truth* en anglais). Ces vérités de terrain sont définies manuellement pour une base donnée afin de servir de référence pour la comparaison d'algorithmes et/ou de données d'entraînement pour les algorithmes d'apprentissage machine.

La fonction la plus utilisée est l'*Intersection over Union* (IoU), parfois appelé indice de Jaccard. Soient deux ensembles A, B, l'IoU est défini comme étant le ratio de l'intersection des deux ensembles sur leur union :

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.3}$$

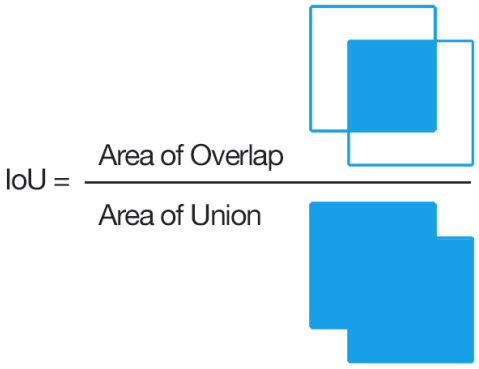


FIGURE 2.23 – Définition de l'IoU (*Intersection over union (IOU) for object detection*).

Une prédiction est considérée comme correcte par rapport à une vérité de terrain si l'IoU calculée dépasse un certain seuil. Le terme « *True Positive* » (TP) est alors utilisé. De même, un « *False Positive* » (FP) est une prédiction qui n'a pas lieu d'être, considéré comme valide par l'algorithme utilisé mais ne correspondant à aucune vérité de terrain et un « *False Negative* » (FN) est une vérité de terrain qui a été manquée, associée à aucune prédiction. La figure 2.24 donne un exemple de vrai positif et de faux positif pour une valeur seuil fixée.

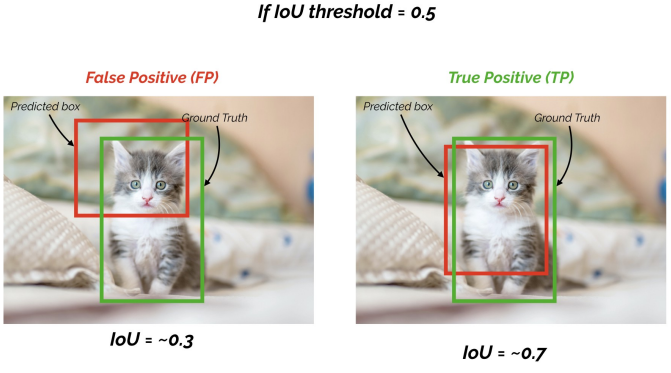


FIGURE 2.24 – Exemple de FP (gauche) et de TP (droite) pour un seuil d'IoU donné (ici 0.5). Source : *mean Average Precision*

Le terme « True Negative » (TN) existe également dans le cas d'ensembles de recherche discrets. En recherche d'information, cela correspond aux documents de la base dont on a correctement prédit leur non-pertinence par rapport à la requête. En détection d'objets cependant, cela correspond à l'ensemble des boîtes n'appartenant pas à la liste des vérités de terrain, donc non quantifiable.

TABLE 2.2 – Classification des prédictions en fonction des cibles.

		Cible	Cible
		Positif	Négatif
Prediction	Positif	True Positive	False Positive
Prediction	Negatif	False Negative	~~True Negative~~

Connaissant le nombre de TP, FP, FN, il est possible de définir la précision et le rappel (*Recall*). La précision est le rapport de propositions correctes sur l'ensemble des propositions :

$$Precision = \frac{TP}{TP+FP} \tag{2.4}$$

Le rappel est le rapport du nombre de propositions correctes par le nombre de vérités de terrain :

$$Rappel = \frac{TP}{TP+FN} \tag{2.5}$$

La courbe précision-rappel illustre le compromis entre les deux mesures pour différents scores attribués aux prédictions. En effet, si l'on choisit de ne conserver que peu de propositions, FP diminue et la précision augmente, mais cela occasionne un risque de manquer certaines vérités de terrain, causant une augmentation de FN et donc une réduction du rappel.

Un bon détecteur doit produire des propositions couvrant un maximum de vérités de terrain disponibles ($FN \to 0$) et n'identifiant que des objets pertinents ($FP \to 0$). Cela se traduit sur la courbe par une précision qui reste importante même si le rappel augmente. Une grande aire sous la courbe (parfois appelée AUC pour *Area Under the Curve*) illustre un bon détecteur. Cette aire correspond également à l'*Average Precision* (AP).

L'*Average Precision* est une valeur entre 0 et 1 qui est la précision moyennée sur l'ensemble des valeurs de rappel.

En pratique, la courbe n'est pas totalement monotone et produit des « zigzags », complexifiant le calcul de l'AUC. Pour atténuer cet effet, l'approche la plus commune est une interpolation sur 11 points. La forme de la courbe précision-rappel est résumée par la moyenne des valeurs de précision sur 11 valeurs de rappel également réparties entre 0 et 1 compris. L'AP s'exprime sous la forme :

$$AP = \frac{1}{11} \sum_{R \in \{0, 0.1, \ldots, 1.0\}} P_{interp} R \tag{2.6}$$

avec

$$P_{interp}(R) = \max_{\tilde{R}, \tilde{R} \geq R} P(\tilde{R}) \tag{2.7}$$

La figure 2.25 illustre la différence entre P et P_{interp}.

L'AP est calculée individuellement pour chaque classe. Le *mean Average Precision* (mAP) est la moyenne des AP pour les classes considérées par le détecteur :

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.8}$$

avec AP_i étant l'AP de la i^{eme} classe étudiée et N le nombre total de classes étudiées.

Le *benchmark* KITTI Object considère 3 classes : les voitures, les piétons et les cyclistes. Concernant la classe « Voiture », l'AP est calculée en utilisant un seuil sur l'IoU de 0.7 tandis que pour les piétons et les cyclistes, ce seuil est fixé à 0.5. Cela est principalement dû à la plus grande difficulté de détection des piétons et cyclistes, ceux-ci étant physiquement plus petits donc un

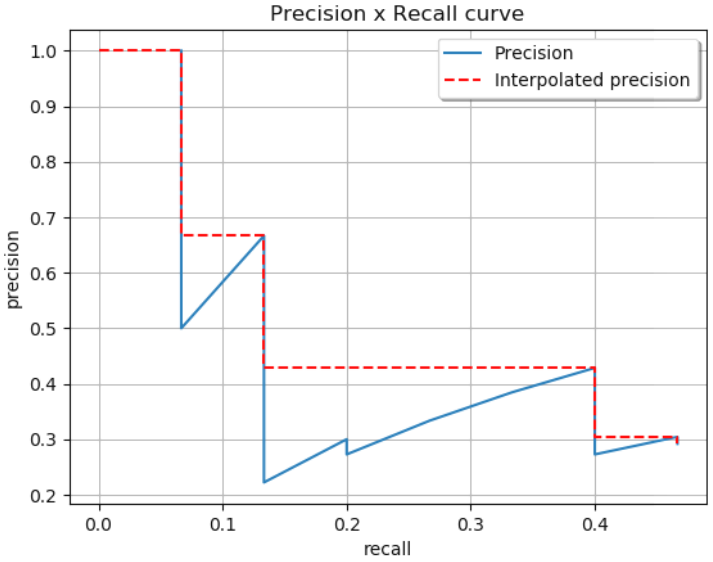


FIGURE 2.25 – Exemple de courbe précision-rappel et de son interpolation. La courbe initiale (bleue pleine) présente des zigzags tandis que son interpolation (rouge discontinue) est monotone décroissante.

faible nombre de points pour les représenter. De plus, en fonction des classes, la portée maximale des labels change. Ainsi, les annotations sur les véhicules peuvent exister jusqu'à 80 m tandis que les annotations sur les piétons dépassent rarement les 40 m. Cette distinction présente plusieurs inconvénients. D'abord, cette distinction a poussé les chercheurs à définir des modèles dédiés aux véhicules et des modèles dédiés aux usagers fragiles. En effet, la majorité des méthodes reposent sur des discrétisations de l'espace. Or, si la portée maximale considérée augmente, la taille des cellules définies et ainsi la précision globale de l'algorithme diminue. Dans ce cas, une augmentation de la résolution de la grille (et donc l'augmentation de cellules dans la grille) peut être réalisée mais au prix d'une augmentation de la consommation mémoire et du temps de calcul.

La méthode d'évaluation de KITTI calcule également des AP séparées en fonction de la difficulté des cibles. Ces difficultés (*Easy*, *Moderate*, *Hard*) sont définies en fonction de la taille apparente de la cible dans l'image et de sa visibilité. Elle peut être dissimulée par d'autres obstacles ou n'être visible qu'en partie car positionnée sur les bords du champ de vision de la caméra.

2.5.3 Variantes de l'Average Precision

Parmi les remarques portées à la métrique d'AP définie plus haut, l'une d'elles est qu'elle ne se soucie que de l'occupation de l'obstacle, et ce, de manière binaire de par le seuil d'IoU utilisé. Les objets détectés sont globalement bien positionnés, mais la marge d'erreur entre la prédiction considérée comme valide par l'IoU et la vérité de terrain n'est pas calculée. Ce type d'incertitude sur les paramètres peut se révéler critique.

Le jeu de données nuScenes définit le *nuScenes Detection Score* (NDS) qui est une moyenne pondérée de plusieurs métriques. Le mAP est calculé pour définir les prédictions positives. Cependant, contrairement au mAP défini dans KITTI qui se base sur l'IoU, le mAP de nuScenes se base uniquement sur la distance entre le centre de la prédiction et celui de la vérité de terrain. Le mAP est la moyenne de 4 seuils de distance : 0.5 m, 1 m, 2 m, 4 m. En plus du mAP, cinq autres scores sont définis sur les prédictions considérées comme positives. Les scores concernent :

– l'erreur en translation ATE (*Average transalation Error*) ;

- l'erreur en échelle *ASE* (*Average Scale Error*) ;
- l'erreur en orientation *AOE* (*Average Orientation Error*) ;
- l'erreur en vitesse *AVE* (*Average Velocity Error*) ;
- l'erreur sur l'attribut *AAE* (*Average Attribute Error*) : certaines classes (par exemple la classe Piéton) disposent d'attributs qui leur sont propres (par exemple « en mouvement », « à l'arrêt debout », « assis »).

Les erreurs sont transformées en scores par la formule $TP\_score = \max(1 - TP\_error, 0.0)$ Le *NDS* est obtenu en assignant un poids de 5 au mAP et de 1 aux autres métriques.

$$NDS = 5mAP + score_{ATE} + score_{ASE} + score_{AOE} + score_{AVE} + score_{AAE} \tag{2.9}$$

2.6 Synthèse

Il existe depuis peu un certain nombre de ressources disponibles pour la détection d'objets 3D dans des environnements dédiés à la conduite. La base KITTI, l'une des plus anciennes bases publiques, mais qui ne date que de 2012, sert en général de point d'entrée et de référence pour vérifier qu'un algorithme dispose de capacités minimales avant de passer à des bases plus grosses telles que `Waymo`. nuScenes introduit le concept d'accumulation permettant la prise en compte de la dynamique de la scène. Cependant, le processus utilisé en devient plus fastidieux, et plus difficilement applicable en conditions réelles. Pandaset dispose, comme KITTI, d'un même ordre de grandeur de scènes. De plus, la pluie ou la brume sur certaines scènes constitue un gain non négligeable. Pandaset, comme nuScenes, fournit à l'utilisateur des annotations à 360°, ce qui le rapproche des conditions d'utilisation en situation réelle. Cette base peut constituer une alternative à la base KITTI pour l'apprentissage mais ne possède pas de *benchmarks* dédiés permettant la comparaison d'algorithmes. À noter que contrairement à KITTI, cette base autorise une utilisation commerciale sans surcoût, ce qui la rend intéressante aux yeux des industriels.

Auparavant réalisées par un faible nombre de personnes, avec leurs biais propres sur des petites quantités, les annotations sont de plus en plus sous-traitées à des sociétés spécialisées afin de gérer les plus grands volumes de données, comme c'est le cas de nuScenes avec la société *Scale*. Si le *crowdsourcing* permet de moyenner les biais et de traiter plus de données, il induit également un risque dans la mesure où les participants ne sont pas forcément experts et peuvent incorporer des erreurs si certaines directives sont absentes. La base Pandaset propose à chaque acquisition des annotations tout autour du véhicule. Cependant, ces annotations sont accumulées sur plusieurs instants $\{t_i, t_{i+1}, t_{i+2}, \ldots\}$. Il en résulte qu'une entité est mentionnée présente n'est pas effectivement visible dans les données capteur à un instant t_i. Ce phénomène est illustré sur la figure 2.26.

L'*Average Précision* est actuellement la métrique la plus utilisée pour évaluer la performance d'un algorithme de détection. Cependant, au niveau industriel, cette métrique reste peu explicite, en partie parce qu'elle représente principalement l'adéquation entre les résultats prédits et une base de référence. Elle n'illustre pas ou très insuffisamment certains aspects des algorithmes quant à leur capacité, en conditions réelles, d'estimer les incertitudes sur les résultats fournis ou d'informer sur les conditions optimales d'utilisation.

Dans le chapitre suivant, nous étudions succinctement les évolutions des méthodes de détection d'objets. Bien que l'objet des travaux se focalise sur la détection 3D, nous faisons un détour sur les méthodes de l'état de l'art en termes de détection générique sur les images RGB. En effet, ces dernières ont grandement influencé les approches 3D, même si l'utilisation de nuages de points impose des traitements différents.

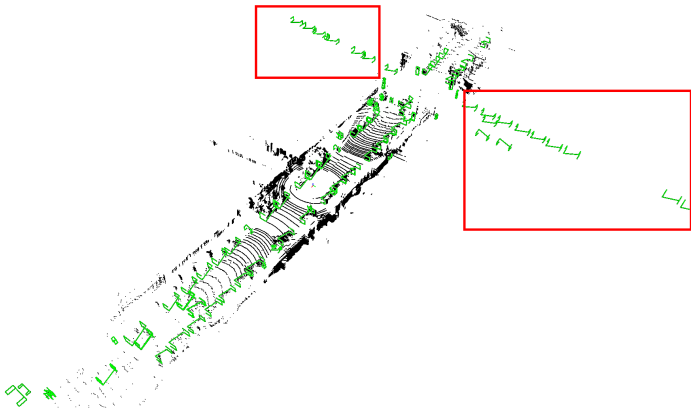


FIGURE 2.26 – Illustration des annotations non traitées de Pandaset. En vert les annotations, les cadres rouges indiquent des régions annotées, mais invisibles pour le LiDAR au moment de l'acquisition.

Chapitre 3

État de l'art en détection d'obstacles depuis des capteurs embarqués à bord de véhicules

Sommaire

3.1 Introduction

Avec le déploiement de bases de données publiques pour le véhicule autonome et l'essor des techniques d'apprentissage artificiel, le domaine de la détection 3D a connu un gain de popularité conséquent. De nombreuses méthodes aux approches variées ont ainsi vu le jour.

Dans ce chapitre, nous présentons dans un premier temps l'évolution des méthodes d'apprentissage profond sur le traitement des images 2D, en particulier pour les tâches de classification et de détection 2D, celles-ci ayant grandement influencé le domaine de la détection 3D. Puis, nous présentons des méthodes de détection 3D qui nous ont semblé pertinentes pour comprendre l'évolution du domaine. Ces méthodes sont séparées en fonction du type de données utilisées en entrée.

3.2 Réseaux de neurones convolutifs pour la détection 2D

Cette section a pour objectif de présenter les réseaux de neurones convolutifs (CNN pour *Convolutional Neural Network*). Nous présentons dans un premier temps les caractéristiques des réseaux de neurones profonds. Puis, nous détaillerons l'évolution des domaines de classification d'image et de détection 2D et l'impact qu'ont eu les réseaux de neurones dans les deux domaines.

3.2.1 Modèles historiques

Perceptron Le perceptron défini dans (Rosenblatt, 1958) est une proposition pour modéliser le fonctionnement d'un neurone biologique. Un neurone biologique est constitué :

- de dendrites, récupérant les signaux d'entrée du neurone ;
- d'un noyau activant un signal de sortie en fonction des signaux d'entrée ;
- d'axones servant de branche de sortie pour les signaux générés ;
- de synapses qui sont les points de connexion entre les axones et les dendrites de neurones voisins.

Le perceptron, par analogie, est modélisé par :

$$f(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x} + b) = \sigma(\sum_i w_i x_i + b) \tag{3.1}$$

avec $\boldsymbol{x} = (x_1 \cdots x_n)^T$ l'entrée du neurone, $\boldsymbol{w} = (w_1 \cdots w_n)^T$ les poids de chaque branche du neurone, b le biais de réseau, σ la fonction d'activation du neurone permettant l'ajout d'une non-linéarité. La figure 3.1 illustre graphiquement un perceptron et la comparaison avec un neurone biologique.

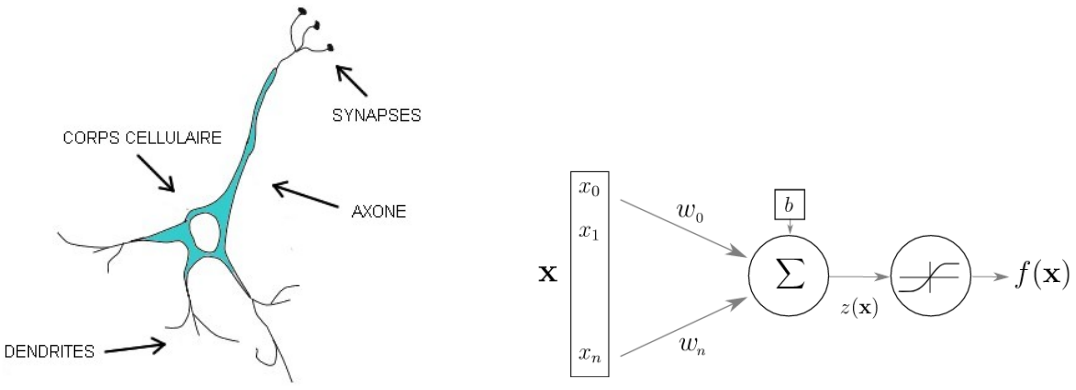


Figure 3.1 – Comparaison entre neurone biologique et neurone formel.

Perceptron multicouche (MLP for Multi-Layer Perceptron) Un perceptron multicouche est un type de réseau de neurones artificiel constitué d'une couche d'entrée, d'une couche de sortie, d'une ou de plusieurs couches cachées intermédiaires entre les deux. Une couche est définie comme un ensemble de perceptrons non reliés entre eux. La figure 3.2 illustre un exemple de MLP avec deux couches cachées.

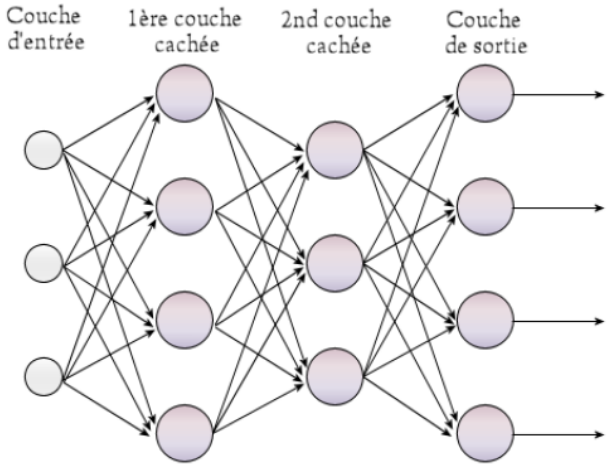


FIGURE 3.2 – Illustration d'un MLP à deux couches cachées. Source : *Perceptron multicouche - Wikipédia*

Entraînement d'un réseau de neurones L'entraînement d'un réseau de neurones consiste à déterminer les valeurs des paramètres de ce réseau afin de minimiser un objectif représenté par une fonction de coût. Nous nous plaçons dans le cadre de l'apprentissage supervisé, c'est-à-dire que pour les exemples de la base de données, les sorties voulues sont connues. Les fonctions d'activation non linéaires au sein du réseau rendent l'optimisation des paramètres du réseau non convexe. De ce fait, l'approche la plus employée consiste en une approche itérative. À chaque itération, l'erreur sur la prédiction est propagée au sein des neurones du réseau afin de mettre à jour les poids du réseau en vue de réduire la valeur de cette fonction de coût. Cette étape de mise à jour de paramètres est en général réalisée par un algorithme fondé sur la descente de gradient.

L'une des méthodes d'optimisation les plus communes pour les réseaux de neurones est la descente de gradient stochastique (SGD) (BOTTOU, 2010). Si nous notons θ_k les paramètres du réseau de neurones à l'étape k, une mise à jour des paramètres à l'aide de la SGD peut être décrite par :

$$\theta_{k+1} \leftarrow \theta_k - \eta_k \nabla_\theta J(\theta_k, x^{(i)}, y^{(i)}) \tag{3.2}$$

avec η_k le taux d'apprentissage (*learning rate*) à l'instant k, $(x^{(i)}, y^{(i)})$ le couple entrée/sortie voulue, J la fonction objectif, $\nabla_\theta J$ le gradient de la fonction objectif J vis-à-vis des paramètres du réseau.

La plupart des méthodes utilisées à l'heure actuelle, telles que Momentum (QIAN, 1999) ou Adam (KINGMA et BA, 2014) sont des modifications de la SGD visant à améliorer la convergence.

3.2.2 Réseau de neurones convolutif

Un premier prototype appelé Cognitron a été défini par Fukushima (FUKUSHIMA, 1975). Les premiers modèles de CNN se rapprochant des réseaux actuels ont été définis par LeCun et al. (LECUN, 1989 ; LECUN et al., 1990 ; LECUN et al., 1998), essentiellement pour de la reconnaissance de caractères manuscrits avec le réseau LeNet, illustré sur la figure 3.3.

Les réseaux de neurones convolutifs (CNN pour *Convolutional Neural Network*) utilisent des couches de convolution. Les couches de convolution fonctionnent sur le principe des fenêtres

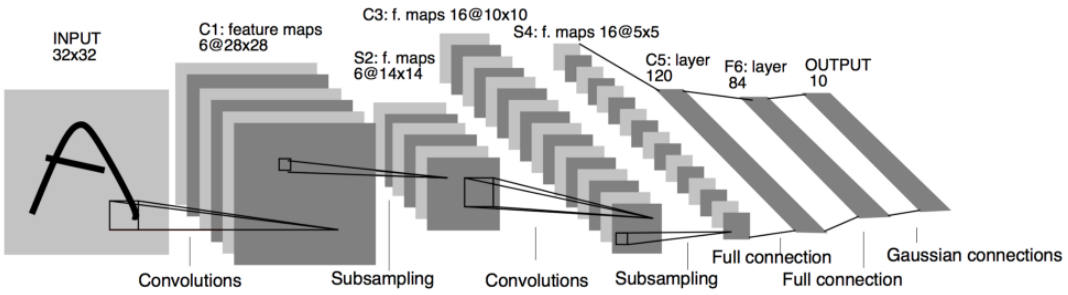


FIGURE 3.3 – Architecture du réseau LeNet (LECUN et al., 1990).

glissantes comme le montre la figure 3.4. L'image est balayée par un ensemble de noyaux de convolutions, chaque noyau extrayant des caractéristiques propres au niveau local. L'opération de convolution permet d'acquérir des informations visuelles pertinentes et de conserver les relations spatiales entre les différents éléments visualisés dans l'image. Comme les MLP, les CNN sont organisés en couches successives. Les couches convolutives les plus proches de l'entrée tendent à capturer des caractéristiques locales, dites de « bas niveau » telles que les contours, les textures ou les couleurs. Les couches plus profondes fusionnent ces caractéristiques bas niveau pour extraire des caractéristiques plus abstraites telles que la présence d'un objet d'un certain type. L'un des inconvénients des MLP provient du fait que chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et précédente, créant ainsi un nombre de poids dépendant de la taille de l'image. Dans le cas des couches de convolutions, chaque neurone n'est relié qu'aux unités dans son champ réceptif. De plus, les poids sont partagés entre les neurones d'une même couche. Comme les noyaux de convolution sont partagés sur les neurones, les motifs obtenus sont indépendants de la position dans l'image.

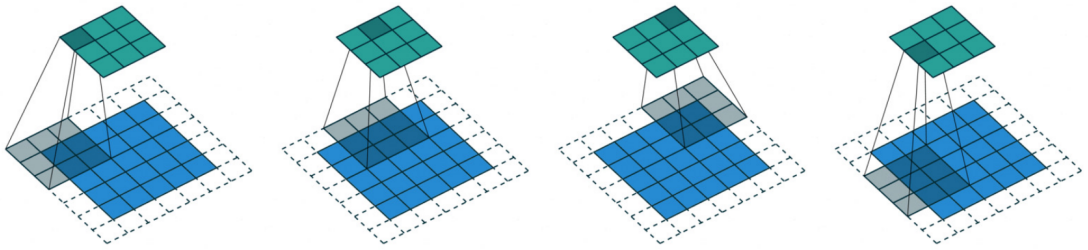


FIGURE 3.4 – Fonctionnement d'une convolution 2D. Taille du noyau : 3, Pas : 2, *padding* (zone périphérique en tirets) : 1. Source : *Convolutions in Deep Learning*

3.2.3 CNN pour la classification d'images

La première véritable application des réseaux de neurones artificiels fût la classification d'images. Nous détaillons dans cette section la distinction principale entre les méthodes dites traditionnelles et les méthodes utilisant des CNN, à savoir la séparation ou la fusion des tâches d'extraction de caractéristiques et la classification comme l'illustre la figure 3.5.

Traitement traditionnel

Le processus traditionnel d'un algorithme de classification automatique d'images repose sur deux phases distinctes : l'extraction manuelle des caractéristiques majeures et la classification des caractéristiques extraites.

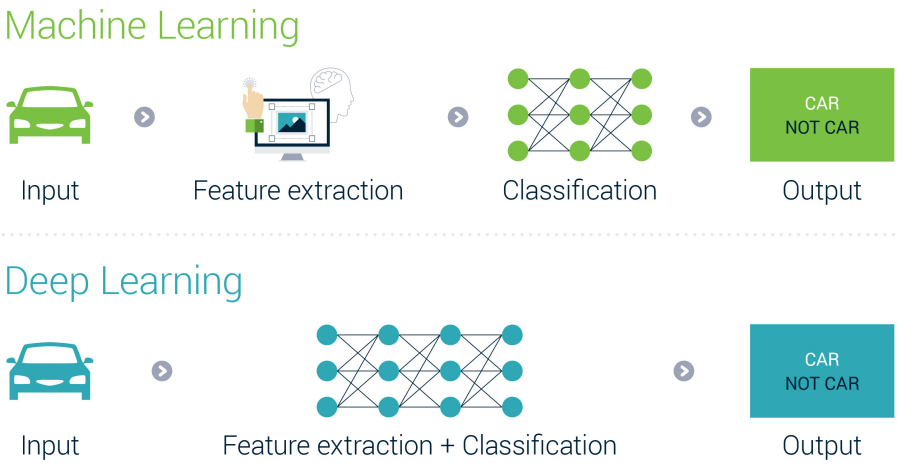


FIGURE 3.5 – Différence de chaînes de traitement entre méthodes traditionnelles et réseaux de neurones pour la classification d'images.

Extraction des caractéristiques Pour une tâche donnée, l'image brute ne fournit pas les éléments critiques nécessaires à la réalisation de l'objectif. Il faut dans un premier temps extraire de l'image des caractéristiques discriminantes. Dans le cas de classification d'images, il faut une représentation compacte décrivant le contenu de l'image. Deux images illustrant des sujets très différents doivent avoir des représentations distantes afin de pouvoir correctement les séparer. De ce fait, la qualité des résultats dépend grandement de la représentation choisie.

Classification Les représentations obtenues sont ensuite classifiées par un algorithme choisi. Les méthodes d'apprentissage machines les plus utilisées pour la classification sont les *Support Vector Machines* (SVM), les *Random Forest* et les *K-nearest neighbours* (KNN).

Critique de l'approche La constitution des représentations des images nécessite un certain nombre d'a priori et implique des choix dépendant du contexte d'utilisation et des connaissances de l'ingénieur. De plus, le choix d'une représentation plutôt qu'une autre peut grandement impacter les capacités de l'algorithme de classification.

Classification après AlexNet

En 2012, AlexNet (KRIZHEVSKY, SUTSKEVER et HINTON, 2012) démontre les capacités des CNN par ses performances jusqu'alors incomparables sur le jeu de données de classification ImageNet. Depuis, les réseaux de neurones artificiels sont devenus la norme pour tout ce qui concerne le traitement automatique de ressources multimédia. L'un des aspects les plus appréciés des réseaux de neurones est l'entraînement bout-à-bout (*end-to-end learning*). Les utilisateurs ne fournissent que les couples entrée/sortie, mais ne doivent pas gérer manuellement la représentation des données, celle-ci est automatiquement apprise par le réseau.

Développement des architectures Comme la figure 3.3 l'indique, le réseau LeNet est constitué de moins de dix couches. Avec l'avènement du calcul sur GPU, les réseaux sont devenus plus imposants. Cet agrandissement des réseaux a également occasionné une hausse des performances sur le jeu de données ImageNet. La figure 3.6 montre une comparaison du nombre d'opérations réalisées et des performances de différentes architectures de CNN sur le benchmark de classification d'images Imagenet. La tendance illustrée sur la figure semble affirmer le fait que les performances augmentent globalement avec le nombre d'opérations effectuées.

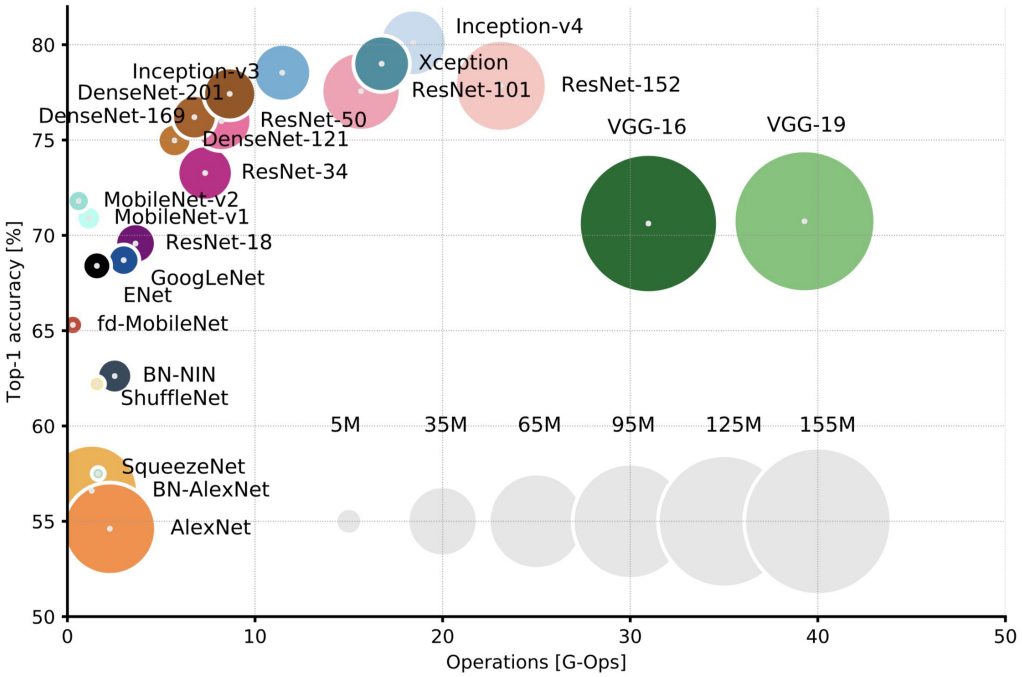


FIGURE 3.6 – Exactitude de la meilleure prédiction en fonction du nombre d'opérations réalisées. La taille de chaque cercle correspond au nombre de paramètres du modèle. Source : (CANZIANI, PASZKE et CULURCIELLO, 2016 ; *Analysis of deep neural networks*).

Cependant l'architecture « couche par couche » simple est en général délaissée au profit d'architectures disposant de connexions internes plus complexes. En effet, des réseaux comme VGG16 (SIMONYAN et ZISSERMAN, 2014) disposent d'un grand nombre de paramètres, mais l'utilisation séquentielle pure des couches ne semble plus apporter d'améliorations suffisantes. Cette stagnation est notamment due à un phénomène d'effacement du gradient au travers des couches. (HE et al., 2016) a introduit les connexions résiduelles mises en œuvre dans les réseaux labellisés ResNet tandis que les réseaux dits DenseNet font appel à des connexions denses présentées dans (HUANG et al., 2017) (Fig. 3.7). Ces deux approches permettent d'intégrer des caractéristiques issues de couches plus proches de l'entrée vers les couches plus profondes, permettant ainsi l'entraînement de réseaux bien plus profonds.

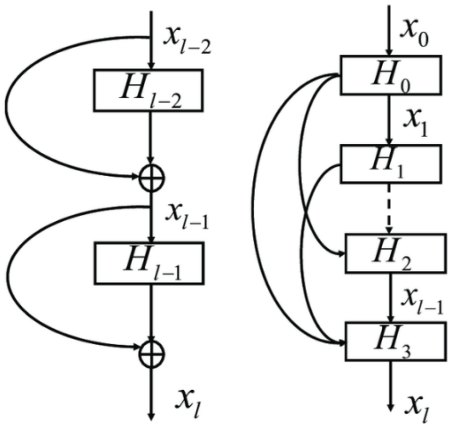


FIGURE 3.7 – Illustration des connexions résiduelles et des connexions denses (MA et al., 2019).

Normalisation Les réseaux de neurones actuels disposent de plusieurs milliers, voire millions de paramètres organisés en couches. Cependant, durant un entraînement, les poids de chaque couche sont mis à jour en considérant les poids des couches précédentes comme fixes, ce qui n'est pas le cas. Ce phénomène peut alors complexifier et allonger les entraînements. La *Batch Normalization* (Ioffe et Szegedy, 2015) a ainsi été développée pour stabiliser et accélérer les entraînements en normalisant les entrées de chaque couche dans chaque *batch*. Plusieurs méthodes concurrentes de normalisation ont été proposées par la suite comme le montre la figure 3.8. La *Group Normalization* (Wu et He, 2018) a par exemple été développée principalement pour mieux gérer les entraînements pour lesquels la taille du batch est faible.

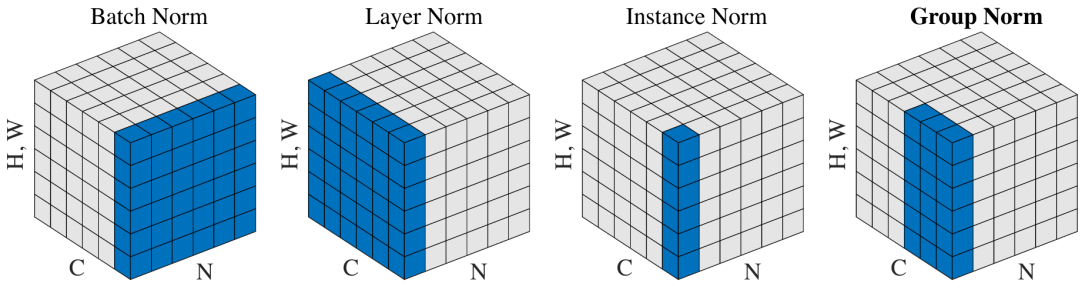


Figure 3.8 – Exemples de normalisation employées dans les réseaux de neurones. N est l'axe des *batches*, C l'axe des canaux et (H, W) l'axe des dimensions spatiales. Image extraite de (Wu et He, 2018).

Fonctions d'activation Un autre aspect actif de la recherche concerne l'évolution des fonctions d'activation. Les fonctions historiques sont la fonction sigmoïde $f(x) = \frac{1}{1+e^{-x}}$ et la fonction tangente hyperbolique. Cependant, les valeurs absolues de sortie étant comprises entre 0 et 1, les valeurs des gradients tendaient à s'annuler au travers des couches du réseau. La fonction `ReLU` utilisée dans AlexNet (Krizhevsky, Sutskever et Hinton, 2012) a été conçue pour atténuer ce phénomène, la mise à zéro des valeurs négatives pouvant toujours annuler la mise à jour de certaines parties du réseau. De plus, la fonction n'est pas dérivable en 0. De ce fait, d'autres alternatives furent développées telles que `Leaky ReLU` (Maas, Hannun et Ng, 2013), `ELU` (Shah et al., 2016) ou encore `GeLU` (Hendrycks et Gimpel, 2016). La figure 3.9 illustre les profils des courbes des fonctions d'activation les plus communes.

3.2.4 CNN pour la détection d'objets dans l'image

Dans cette section, nous présentons succinctement l'évolution du domaine de la détection d'objets dans des images RGB. Ce domaine a par la suite fortement impacté les méthodes axées détection 3D. S'agissant du traitement d'images, la même scission entre méthodes traditionnelles et méthodes axées réseaux de neurones a eu lieu comme le montre la figure 3.10.

Méthodes de détection traditionnelles

Comme énoncé précédemment, le traitement d'images repose sur deux phases : une phase d'extraction des caractéristiques et une phase de classification. L'une des premières contributions majeures fut la méthode de Viola et Jones (Viola et Jones, 2001). Orientée vers la reconnaissance de visages, elle permettait une détection en temps réel et bien plus rapide que des méthodes aux performances équivalentes. La méthode se distingue sur trois aspects : l'utilisation d'images intégrales pour accélérer de manière significative le calcul de caractéristiques de Haar, la sélection des caractéristiques pertinentes par *Adaboost* et l'utilisation d'une cascade de détecteurs pour accorder un maximum de ressources aux propositions recouvrant des visages. Par la suite, Dalal

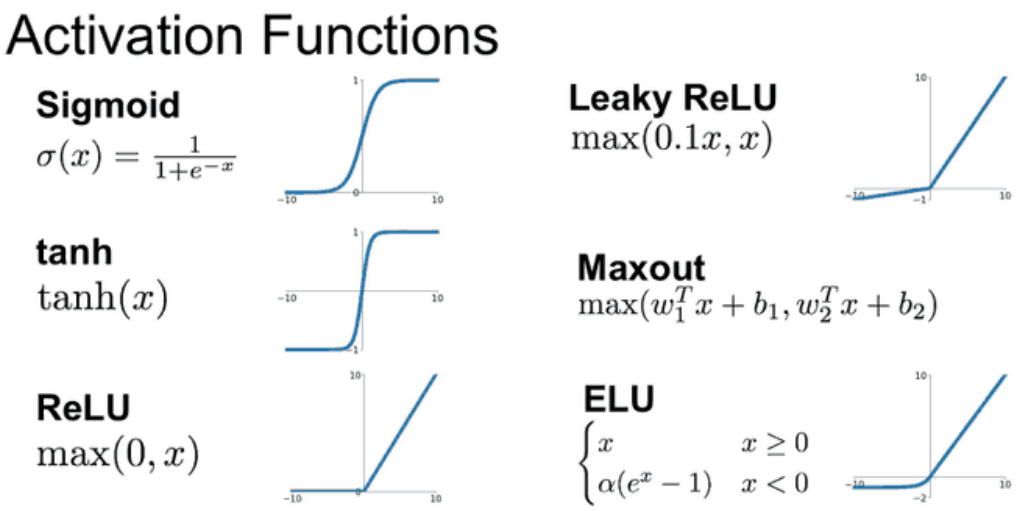


FIGURE 3.9 – Exemples de fonctions d'activation couramment utilisées. Source : *Activation functions - Medium*

et Triggs proposèrent un détecteur de personnes utilisant les histogrammes de gradients orientés (HOG) comme caractéristiques (DALAL et TRIGGS, 2005). Afin de mieux gérer les occlusions, mais aussi l'aspect déformable des personnes, Felzenszwalb et al. suggérèrent de décomposer les cibles en sous parties (FELZENSZWALB, MCALLESTER et RAMANAN, 2008 ; FELZENSZWALB et al., 2009). Dans le cas de la détection de personnes, cela revient d'abord à détecter la tête, les membres et le torse.

Réseaux de neurones pour la détection

Les performances du réseau AlexNet sur le *benchmark* Imagenet ont montré les capacités des réseaux de neurones sur une tâche de classification d'images.

L'une des premières tentatives marquantes de transposition à la détection fut R-CNN (GIRSHICK et al., 2014). Des propositions sont générées dans un premier temps par Selective Search (Van de SANDE et al., 2011). Chaque proposition est redimensionnée puis donnée à un CNN pré-entraîné pour en extraire des caractéristiques. Les caractéristiques obtenues sont ensuite classifiées par SVM pour estimer le contenu de chaque proposition et corrigées par un modèle de régression distinct. Bien que performante, cette solution présente deux problèmes. Le temps d'exécution est très long dans la mesure où le CNN doit être appliqué à chaque proposition, indépendamment des autres. De plus, le réseau nécessite une taille fixe d'image. SPPNet (HE et al., 2015) comble ces deux problèmes, d'une part, en ne calculant les *features* qu'une seule fois sur toute l'image, et d'autre part, par l'intermédiaire d'un opérateur dit *Spatial Pyramid Pooling* permettant d'extraire de la carte globale des caractéristiques de dimensions fixes quelque soit les dimensions de la proposition. Fast-RCNN (GIRSHICK, 2015) améliore les approches proposées par R-CNN et SPPNet en intégrant la classification et la régression au sein du réseau de neurones. Bien que les approches successives accélèrent le temps de traitement et la facilité d'utilisation en intégrant les diverses parties du système de détection dans un réseau de neurones, la proposition de région est toujours menée par *Selective Search*. Faster-RCNN (REN et al., 2015) marque un palier en intégrant la proposition de régions et ainsi toute la chaîne de traitement dans un réseau unique. Faster-RCNN extrait les caractéristiques des propositions en bout de réseau. Les *Feature Pyramid Networks* (FPN) (LIN et al., 2017a) améliorent le concept en extrayant les caractéristiques à différentes étapes du réseau afin d'établir de l'information sémantique à différentes échelles. Le réseau, reposant sur l'architecture du Faster-RCNN, a notamment permis une meilleure détection

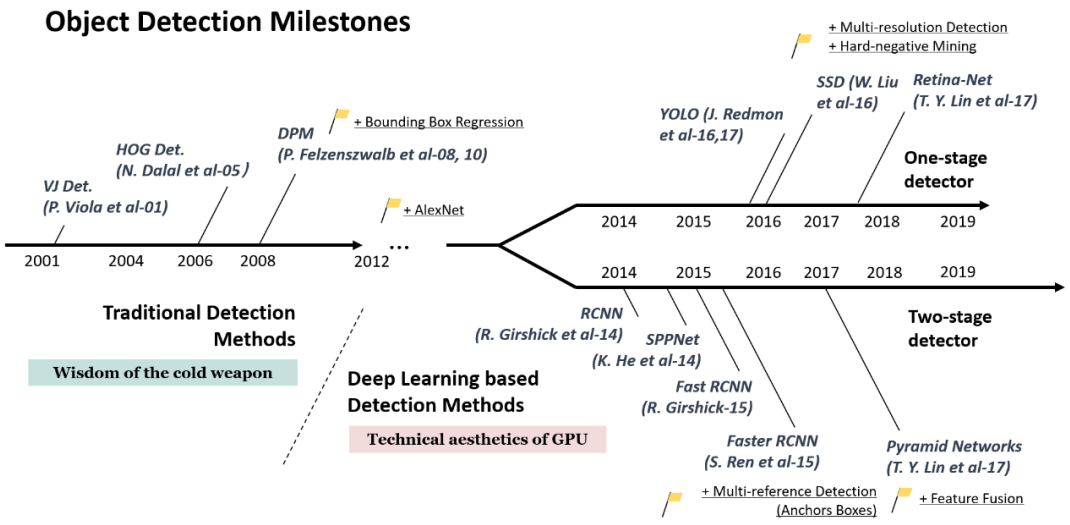
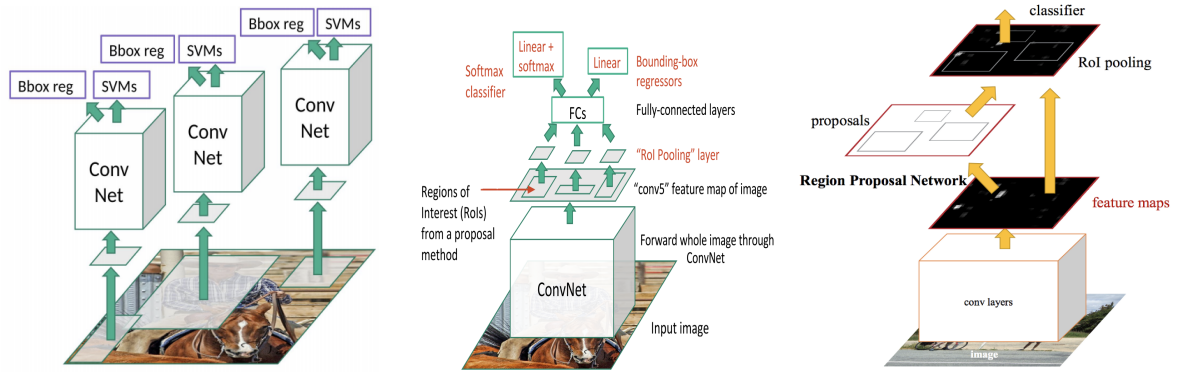


FIGURE 3.10 – Frise chronologique des méthodes de détection 2D exposée dans (ZOU et al., 2019).

sur des cibles de tailles très variées. La figure 3.11 montre sous forme de schémas l'évolution entre RCNN, Fast-RCNN et Faster-RCNN.



(a) RCNN (b) Fast-RCNN (c) Faster-RCNN

FIGURE 3.11 – Comparaison des architectures de RCNN, Fast-RCNN et Faster-RCNN. Source : *Object Detectors*

Les méthodes énoncées ci-dessus sont dites à deux phases (*two-stage detection*) dans la mesure où une première phase est chargée d'émettre des propositions tandis qu'une seconde doit véritablement réaliser une classification et une régression. YOLO (pour « *You Only Look Once* ») (REDMON et al., 2016) est l'un des premiers détecteurs dits « *single-stage* ». L'image est divisée en cellules et pour chaque cellule, le réseau estime une boîte englobante et les probabilités de classe. Le réseau illustre des performances s'approchant de celles de Faster-RCNN mais en étant bien plus rapide. SSD (LIU et al., 2016) utilise un fonctionnement proche de celui des FPN, c'est-à-dire que les détections finales sont l'agglomération de détections effectuées à différentes échelles sur différentes couches du réseau. Les réseaux *single-stage* constituent un compromis en choisissant la vitesse au détriment de la performance de détection. Lin et al. proposèrent avec leur réseau RetinaNet une nouvelle fonction objectif, baptisée *Focal Loss* (LIN et al., 2017b). La supposition principale concerne le nombre de propositions « positives » (celles encadrant les cibles) sont bien moins nombreuses que les boîtes dites d'arrière-plan. Dans le cas d'un détecteur à deux phases, la première permet de filtrer les propositions, ce qui n'est pas le cas d'un réseau *single-stage*. La

Focal Loss est une modification de la fonction d'entropie croisée couramment utilisée dans la classification qui permet de donner plus d'importance aux cas pour lesquels le réseau présente plus de difficultés. RetinaNet a, grâce à cette fonction objectif, pu atteindre les performances des détecteurs *two-stage* tout en conservant sa vitesse d'exécution.

La grande majorité des méthodes présentées se fondent sur des ancres, qui sont des boîtes définies en amont pour fournir une base à la régression. L'un des inconvénients de cette méthode provient des hyperparamètres requis (nombre, position et dimensions des ancres, seuils pour définir qu'une ancre contient une cible, etc.) qui peuvent fortement impacter les entraînements. De plus, chaque jeu de données présente ses propres caractéristiques, nécessitant un ajustement précis. De ce fait, des méthodes dites *anchor-free* ont fait leur apparition, promettant plus de flexibilité en assimilant la détection à un problème de localisation de points clés (*keypoint detection*). Un objet est représenté par un ou un ensemble de points. CornerNet (Law et Deng, 2018) vise la détection des coins supérieur-gauche et inférieur droit, tandis que CenterNet (Zhou, Wang et Krähenbühl, 2019) se focalise sur le centre des boîtes comme illustré sur la figure 3.12.

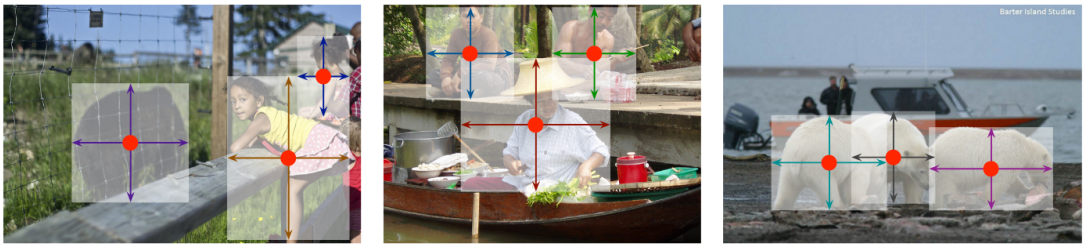


Figure 3.12 – Représentation des sorties de CenterNet (Zhou, Wang et Krähenbühl, 2019). Chaque cible est modélisée comme le centre de sa boîte englobante. Les dimensions de la boîte englobante sont estimées au niveau du centre de la boîte.

3.3 Étude bibliographique en détection 3D : une synthèse

3.3.1 Introduction

Après avoir introduit les principes de la détection d'objet par des approches mettant en œuvre des réseaux de neurones, nous réalisons dans cette section un état de l'art des méthodes présentes dans la littérature. Le domaine ayant connu un grand engouement, nous avons décidé de sélectionner un certain nombre de méthodes qui nous ont semblé les plus pertinentes. De plus, nous avons choisi de partitionner ces méthodes suivant les capteurs exploités par la méthode : LIDAR seul, caméra(s) seule(s) ou utilisation des deux types de capteurs simultanément. D'une part, cela permet de cloisonner les techniques en fonction des capteurs présents sur le véhicule robotisé, d'autre part, certaines méthodes emploient des traitements communs du fait de la présence d'un certain type de données.

Ces trois partitions sont synthétisées au sein des tableaux 3.1, 3.2 et 3.3. Ces tableaux sont constituées de 37 entrées dont 13 se réfèrent à des méthodes n'exploitant que les nuages de points LiDAR, 11 méthodes n'utilisant qu'une ou plusieurs caméras et 13 méthodes utilisant une combinaison des deux types de capteurs. Les architectures de réseaux de neurones se voient en général attribuer un nom court afin de les distinguer des architectures concurrentes. Chaque tableau décrit succinctement les méthodes présentes en leur sein notamment par :

- leur nom ;
- la présence ou l'absence d'un prétraitement sur les données ;
- une rapide description de la méthode ;
- les classes d'objets sur lesquels le réseau a été entraîné ;

– les jeux de données utilisés pour l'entraînement des réseaux.

Bien que l'essor des réseaux de neurones ait réellement commencé en 2012 avec le réseau AlexNet (Krizhevsky, Sutskever et Hinton, 2012) sur des tâches de classification sur ImageNet (Deng et al., 2009), les premiers résultats conséquents pour la détection 3D sur des scènes de conduite n'eurent lieu qu'à partir de 2017 avec MV3D (Chen et al., 2017b). L'apparition de VoxelNet (Zhou et Tuzel, 2018) fut également un grand pas dans la mesure où la méthode fut la première grande preuve que le capteur LiDAR puisse être exploité seul, sans les contraintes liées à la coordination avec la caméra telles que les calibrations extrinsèques ni les données images supplémentaires à traiter. De ce fait, il existe dans la littérature un très grand nombre de méthodes de détection à partir d'un unique capteur LiDAR.

En termes de performances, du fait qu'elles exploitent un capteur de distance, les méthodes « LiDAR » et « LiDAR + Caméra » sont au coude à coude. Les méthodes « Caméra » ne rivalisent que très peu avec elles. Bien qu'elles soient comparées aux deux catégories précédentes sur les mêmes *benchmarks*, les comparaisons dans les articles sont souvent menées entre méthodes « Caméra ». Il est à noter que le domaine de la détection 3D sur images caméras est un domaine plus ancien du fait de la disponibilité antérieure des capteurs caméras et du grand nombre d'études déjà réalisées sur la détection dans les images.

Avec le tableau, nous constatons que la quasi-totalité des méthodes présentes ont été principalement entraînées et testées sur la base KITTI. Son succès repose sur plusieurs points :

– elle fut l'une des premières bases à donner un accès public à des nuages de points issus d'un capteur onéreux (Velodyne HDL-64E) ;

– elle contient un nombre conséquent de données annotées, permettant l'entraînement de techniques d'apprentissage machine ;

– elle dispose d'un système de *benchmark* offrant aux utilisateurs un contexte unifié pour comparer leurs méthodes, mais également de partager leurs codes sources s'ils le souhaitaient ;

– elle fut rendue publique relativement tôt (2012−2013).

Depuis, d'autres bases bien plus conséquentes telles que nuScenes sont apparues. Toutefois, la base KITTI reste la référence principale.

On constate également sur le tableau que la totalité des méthodes présentées ont été appliquées à la détection de voitures, un nombre plus faible a inclus la détection de piétons et de cyclistes. À noter que pour la grande majorité des cas, si les piétons/cyclistes sont traités, l'entraînement diffère de celui réalisé sur les véhicules. Les annotations des piétons/cyclistes dans la base KITTI sont limitées à 50 m tandis que celles des voitures peuvent atteindre 80 m.

Dans les prochains paragraphes, nous décrirons plus en détail le fonctionnement des méthodes présentes dans les tableaux ainsi que les liens entre certaines d'entre elles.

TABLE 3.1 – Tableau récapitulatif de réseaux de détection 3D à l'aide de nuages de points 3D LiDAR.

Référence	Donnée d'entrée	Traitement initial	Brève synthèse de l'approche	Types d'objets	Jeux de données
BirdNet (BELTRAN et al., 2018)	Points 3D	Discrétisation en une BEV de 3 canaux (hauteur, densité, réflectance)	Application d'une architecture similaire à Faster RCNN	Voiture, Piéton, Cycliste	KITTI
Complex-YOLO (SIMON et al., 2019)	Points 3D	Discrétisation en une BEV de 3 canaux (hauteur, densité, réflectance)	Application d'une architecture similaire à YOLO, représentation complexe des angles pour la régression	Voiture, Piéton, Cycliste	KITTI
VoxelNet (ZHOU et TUZEL, 2018)	Points 3D	Voxelisation 3D	Apprentissage des caractéristiques locales à chaque voxel par un *Voxel Feature Encoder* (VFE), convolutions 3D qui aplatissent la grille 3D sur l'axe vertical puis RPN	Voiture, Piéton, Cycliste	KITTI
SECOND (YAN, MAO et LI, 2018)	Points 3D	Voxelisation 3D	Similaire à VoxelNet, remplace les convolutions 3D pleines par des convolutions 3D creuses pour réduire le coût computationnel	Voiture, Piéton, Cycliste	KITTI, nuScenes
PointPillars (LANG et al., 2019)	Points 3D	Voxelisation 2.5D (Voxels colonnes, pas de discrétisation verticale)	Similaire à VoxelNet, pas de convolutions 3D grâce aux voxels colonnes	Voiture, Piéton, Cycliste	KITTI, nuScenes
OHS (CHEN et al., 2019a)	Points 3D	Voxelisation 3D, chaque voxel n'est représenté que par son point moyen	Architecture inspirée par SECOND/PointPillars, les cibles sont représentées par un ensemble de *hotspots* (voxels non vides appartenant à la cible), détection de *keypoints*	Voiture, Piéton, Cycliste et autres classes	KITTI, nuScenes
PointRCNN (SHI, WANG et LI, 2019)	Points 3D	–	Détection en 2 phases : (1) Génération de propositions 3D par segmentation de nuage de points, chaque point sélectionné génère sa propre proposition, (2) Affinage de boîte 3D	Voiture, Piéton, Cycliste	KITTI
Fast Point R-CNN (CHEN et al., 2019b)	Points 3D	Voxelisation 3D	Détection en 2 phases : (1) Proposition de régions 3D à l'aide d'une architecture inspirée de VoxelNet/SECOND (2) Extraction de *features* du réseau de la phase 1, fusion avec les points 3D correspondants à la région estimée, réseau dédié à l'estimation fine des boîtes 3D	Voiture, Piéton, Cycliste	KITTI
CenterPoint (YIN, ZHOU et KRÄHENBÜHL, 2020)	Points 3D	Voxelisation	Architecture inspirée de CenterNet (détection 2D) et SECOND/PointPillars, les objets ne sont représentés que par leur centre sur la carte de classification, possibilité d'intégration de tracking.	nuScenes classes	nuScenes
PV-RCNN (SHI et al., 2020)	Points 3D	Voxelisation 3D + *Furthest Point Sampling* pour extraire des points clés du nuage	Détection en 2 phases : (1) Proposition de région par une architecture inspirée de SECOND, les *feature maps* 3D sont échantillonnées sur plusieurs échelles à l'aide des points clés estimés en prétraitement (2) Extraction de caractéristiques sur les points clés par *RoI Grid Pooling* puis réseau d'affinage des propositions.	Voiture, Cycliste	KITTI
Part-A^2 Net (SHI et al., 2019)	Points 3D	–	Détection en 2 phases : (1) Segmentation sémantique du nuage de points, estimation des parties des cibles et proposition de régions pour les points sélectionnés (2) *Pooling* des points dans chaque proposition puis réseau dédié pour rassembler les parties estimées et affiner la détection 3D.	Voiture, Piéton, Cycliste	KITTI
RangeRCNN (LIANG et al., 2020)	Points 3D	Voxelisation + carte de profondeur en projection sphérique	Une architecture encodeur-décodeur extrait des features 2D de la carte de profondeur, un nuage de points avec les nouvelles *features* est recréé puis voxelisé, un RPN issu de VoxelNet extrait les propositions.	Voiture	KITTI
LaserNet (LIANG et al., 2020)	Points 3D	Carte de profondeur en projection sphérique	Un réseau entièrement convolutif traite la carte de profondeur générée, à chaque point est associé une probabilité de classe dt une distribution de probabilité sur la boîte englobante.	Voiture, Piéton, Cycliste	KITTI

TABLE 3.2 – Tableau récapitulatif de réseaux de détection 3D à l'aide de caméras RGB (monoculaire).

Référence	Donnée d'entrée	Traitement initial	Brève synthèse de l'approche	Types d'objets	Jeux de données
OFT (RODDICK, KENDALL et CIPOLLA, 2018)	RGB Image	–	ResNet Backbone, projection des bordures des voxels 3D dans les *feature maps* de l'image RGB, chaque voxel 3D se voit attribuer les vecteurs caractéristiques situés à l'intérieur de la projection, puis traitement similaire à Voxelnet.	Voiture	KITTI
M3D-RPN (BRAZIL et LIU, 2019)	RGB Image	–	DenseNet Backbone puis 2 branches : (1) extraction de caractéristiques globales à l'image entière, (2) extraction locale de *features* dites *"Depth-Aware"* : l'image est découpée en tranches horizontale, chaque tranche étant traitée par sa propre convolution.	Voiture, Piéton, Cycliste	KITTI
MonoDIS (SIMONELLI et al., 2019)	RGB Image	–	ResNet Backbone, 2D et 3D *heads*, les paramètres des boîtes sont divisés en plusieurs sous-groupes pour améliorer la convergence	Voiture, Piéton, Cycliste	KITTI
CubifAE-3D (SHRIVASTAVA et CHAKRAVARTY, 2020)	RGB Image	–	Plusieurs réseaux entraînés séparément : (1) estimation de profondeur par une architecture encodeur-décodeur, (2) exploitation de l'espace latent défini par l'estimateur de profondeur pour la détection 3D	Voiture, Piéton, Cycliste et autres classes	KITTI, nuScenes, KITTI Virtual
Mono3D (CHEN et al., 2016)	RGB Image	–	Évaluation de projections 2D de boîtes 3D sur divers critères liés à l'image (sémantique, forme, etc.) puis estimation de corrections fines	Voiture, Piéton, Cycliste	KITTI
Deep3DBox (MOUSAVIAN et al., 2017)	RGB Image	–	Détection (MS-CNN (CAI et al., 2016)) puis estimation de la boîte par un réseau dédié, orientation gérée par classification de sections	Voiture, Piéton, Cycliste, classes Pascal3D+	KITTI, Pascal3D+ (XIANG, MOTTAGHI et SAVARESE, 2014)
Deep MANTA (CHABOT et al., 2017)	RGB Image, modèles CAD véhicules	–	2 phases. 1/ Détection 2D itérative fine (estimation des parties du véhicule). 2/ Optimisation de pose d'un modèle CAD choisi sur l'image	Voiture	KITTI
RTM3D (LI et al., 2020)	RGB Image	–	2 étapes. 1/ Estimation de points clés dans l'image (centre + vertices = 9 points) 2/ Optimisation sur la différence entre les projections de vertices et les points clés détectés.	Voiture	KITTI
ROI-10D (MANHARDT, KEHL et GAIDON, 2019)	RGB Image	–	Couplage d'un réseau d'estimation de profondeur et d'un réseau de détection 2D puis utilisation des détections 2D comme ROI pour estimation de boîtes 3D et reconstruction de forme 3D optionnelle	Voiture	KITTI
Pseudo-LiDAR (WANG et al., 2019a)	RGB Image (mono ou stéréo)	–	Estimation monoculaire de profondeur, reconstruction du nuage de points correspondant, application d'un détecteur LiDAR sur le nuage reconstruit	–	–
Pseudo-LiDAR++ (YOU et al., 2019)	RGB Image (mono ou stéréo), nuage de points très basse résolution	–	Similaire à Pseudo-LiDAR, utilisation du nuage de points 3D basse résolution comme points de contrôle pour recaler la reconstruction sur des données réelles.	Voiture	KITTI
SMOKE (LIU, WU et TÓTH, 2020)	RGB Image	–	Estimation de *keypoints* dans l'image avec les attributs 3D, régression basée vertices	Voiture	KITTI
Objects as Points (ZHOU, WANG et KRÄHENBÜHL, 2019)	RGB Image	–	Estimation de *keypoints* dans l'image avec les attributs 3D, régression directe des paramètres	Voiture	KITTI

TABLE 3.3 – Tableau récapitulatif de réseaux de détection 3D fusionnant images RGB et nuages de points 3D

Référence	Donnée d'entrée	Traitement initial	Brève synthèse de l'approche	Types d'objets	Jeux de données
MV3D (CHEN et al., 2017b)	Points 3D, image RGB	Discrétisation du nuage 3D en une BEV de 3 canaux (hauteur, densité, réflectance) et projection cylindrique des points 3D	Détecteur en 2 phases : 1/ la BEV génère des propositions 3D, 2/ Les propositions 3D sont projetées dans la BEV, dans l'image RGB et dans la vue cylindrique, les *features* issues des 3 vues sont extraites puis fusionnées pour affiner les propositions.	Voiture	KITTI
Frustum PointNet (QI et al., 2018)	Points 3D, image RGB	–	Un détecteur 2D est appliqué dans l'image, chaque boîte 2D dans l'image a son frustum correspondant dans l'espace 3D et ne représente qu'une seule cible. Pour chaque frustum, les points à l'intérieur sont utilisés pour estimer la boîte 3D de la cible à l'aide d'un réseau similaire à PointNet++.	Voiture, Piéton, Cycliste	KITTI
AVOD (KU et al., 2018)	Points 3D, image RGB	Discrétisation du nuage 3D en une BEV de 3 canaux (hauteur, densité, réflectance)	Détecteur en 2 passes : 1/ Les *feature maps* de la BEV et de l'image RGB sont calculées, les meilleures propositions sont estimées à partir de l'utilisation jointe des 2 *feature maps*, 2/ les propositions 3D sont projetées dans 2 *feature maps*, les caractéristiques correspondantes sont extraites puis fusionnées pour affiner les boîtes 3D.	Voiture, Piéton, Cycliste	KITTI
PointFusion (XU, ANGUELOV et JAIN, 2018)	Points 3D, image RGB	–	Un détecteur 2D est utilisé pour extraire une région de l'image RGB et un frustum 3D, ces derniers sont traités respectivement par un ResNet et un PointNet. Les caractéristiques extraites des 2 réseaux sont fusionnées puis exploitées pour estimer les paramètres de la boîte 3D.	Voiture, Piéton, Cycliste	KITTI, SUN-RGBD
ContFusion (LIANG et al., 2018)	Points 3D, image RGB	–	Calcul de *feature maps* à partir de l'image RGB, échantillonnage des caractéristiques et reprojection dans la BEV à l'aide de « convolutions continues » exploitant le nuage de points, fusion de la BEV issue de l'image RGB avec les *feature maps* BEV issues du LiDAR	Voiture	KITTI, TOR4D (base privée)
IPOD (YANG et al., 2018)	Points 3D, image RGB	Segmentation sémantique sur image RGB	3 parties : 1/ en exploitant le résultat de la segmentation sémantique, des propositions sont générées autour de chaque point associé aux cibles ; 2/ un réseau spécialisés extrait des caractéristiques globale à l'ensemble du nuage de points et locales à chaque proposition puis les fusionnent ; 3/ les features calculées pour chaque proposition sont utilisées dans un réseau d'estimation 3D.	Voiture, Piéton, Cycliste	KITTI
Frustum ConvNet (WANG et JIA, 2019)	Points 3D, image RGB	–	Un détecteur 2D est appliqué dans l'image, chaque boîte 2D dans l'image a son frustum correspondant dans l'espace 3D et ne représente qu'une seule cible. Pour chaque frustum, le nuage de point correspondant est voxélisé sur une seule dimension le long de l'axe du frustum à la manière d'un VoxelNet. Des convolutions 1D sont utilisés sur cette voxelisation afin d'estimer la boîte 3D.	Voiture, Piéton, Cycliste	KITTI
PointPainting (VORA et al., 2020)	Points 3D, image RGB	Segmentation sémantique sur image RGB	Les caractéristiques des points 3D (coordonnées 3D, réflectance) sont concaténées avec les scores délivrés par la segmentation RGB. Un détecteur 3D LiDAR (PointRCNN, PointPillars, etc.) est ensuite utilisé avec ces points 3D « augmentés » en entrée.	Voiture, Piéton, Cycliste	KITTI, nuScenes
MVX-Net (SINDAGI, ZHOU et TUZEL, 2019)	Points 3D, image RGB	Voxelisation 3D	Le *backbone* d'un Faster RCNN pré-entraîné est utilisé pour extraire des *feature maps* RGB. La détection 3D repose sur un VoxelNet. Deux approches pour intégrer l'information RGB dans le VoxelNet sont testées : 1/ les caractéristiques 2D sont échantillonnées à l'échelle du point 3D et fusionnées avant le VFE, 2/ les caractéristiques 2D sont échantillonnées à l'échelle du voxel et intégrées après le VFE	Voiture	KITTI
MMF (LIANG et al., 2019)	Points 3D, image RGB	BEV et Image éparse de profondeur	Détecteur en 2 phases : 1/ La carte éparse de profondeur et l'image RGB sont utilisées pour génère des *feature maps* intermédiaires et une carte dense de profondeur, cette dernière générant un nuage de points plus dense. La BEV est ensuite fusionnée avec les *feature maps* RGB et le nuage dense obtenu pour définir des propositions. 2/ Les *feature maps* précédemment définies sont échantillonnées à l'aide des propositions de la première phase. Un réseau de raffinage est alors chargé d'exploiter des extractions pour affiner les boîtes 3D.	Voiture (main)	KITTI, TOR4D (privée)
CLOCs (PANG, MORRIS et RADHA, 2020)	Points 3D, image RGB	Détections 2D image et détections 3D LiDAR	Fusion tardive : les boîtes 3D sont projetées dans l'image, une matrice creuse illustrant les superpositions entre détections RGB et 3D est créée, un réseau de neurones est utilisé pour attribuer de nouveaux scores aux détections 3D	Voiture (main)	KITTI

3.3.2 Détection 3D à partir de nuages de points

Nous avons vu que le traitement de l'image a connu des progrès spectaculaires au cours des dernières années, notamment grâce à l'utilisation des réseaux de neurones. Néanmoins, les architectures standard, adaptées aux traitements de données structurées, semblent incompatibles de prime abord avec les nuages de points. Aussi, on croise principalement dans la littérature trois traitements (non exclusifs) appliqués aux nuages de points :

– la projection des points. Cette transformation permet d'obtenir, à partir d'un nuage contenant des coordonnées spatiales une représentation assez proche d'une image obtenue par une caméra. Plusieurs types de projections ont été utilisées dans la littérature. Une projection perspective permet de simuler la modélisation des caméras classiques, quitte à engendrer les déformations induites par la perspective. De plus, en fonction de la densité des données utilisées, cette projection peut donner des images dans lesquelles les pixels que l'on qualifierait de valides représentent une faible partie de l'image. Les projections cylindriques et sphériques essaient quant à elles de se rapprocher le plus possible du fonctionnement du capteur d'acquisition afin de pallier ce problème de données valides, mais engendrent des déformations qui sont plus étrangères à la projection perspective. Ce formatage a permis l'exploitation directe d'algorithmes ou d'opérateurs à l'origine développés pour les images caméras (couleur ou nuances de gris). La figure 3.13 illustre un exemple de projection perspective des points 3D dans le plan image ;

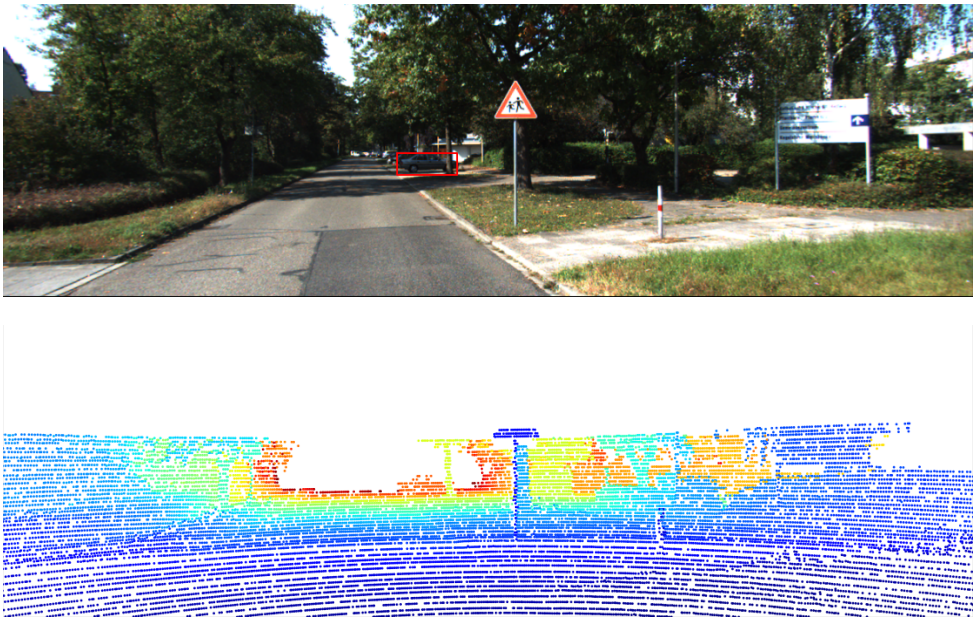


FIGURE 3.13 – Projection perspective des points 3D. La couleur indique la distance sur l'axe longitudinal.

– la transformation en grille régulière. Cette transformation consiste à discrétiser l'espace du nuage de points. En général, seules les dimensions spatiales sont discrétisées. Cela revient à transformer le nuage de points continu en une grille, généralement régulière. Cette transformation repose sur plusieurs paramètres. Dans le cas de grilles régulières, nous pouvons citer la résolution des cellules, la taille de la grille, mais aussi le contenu de la grille. La représentation des cellules vides est également un point important comme mentionné plus haut ;

– l'utilisation directe des points : cette approche a été popularisée dans le domaine de l'apprentissage profond par la publication du réseau PointNet (QI et al., 2017a). À l'origine conçu pour la classification et la sémantisation d'un nuage de points, le réseau a

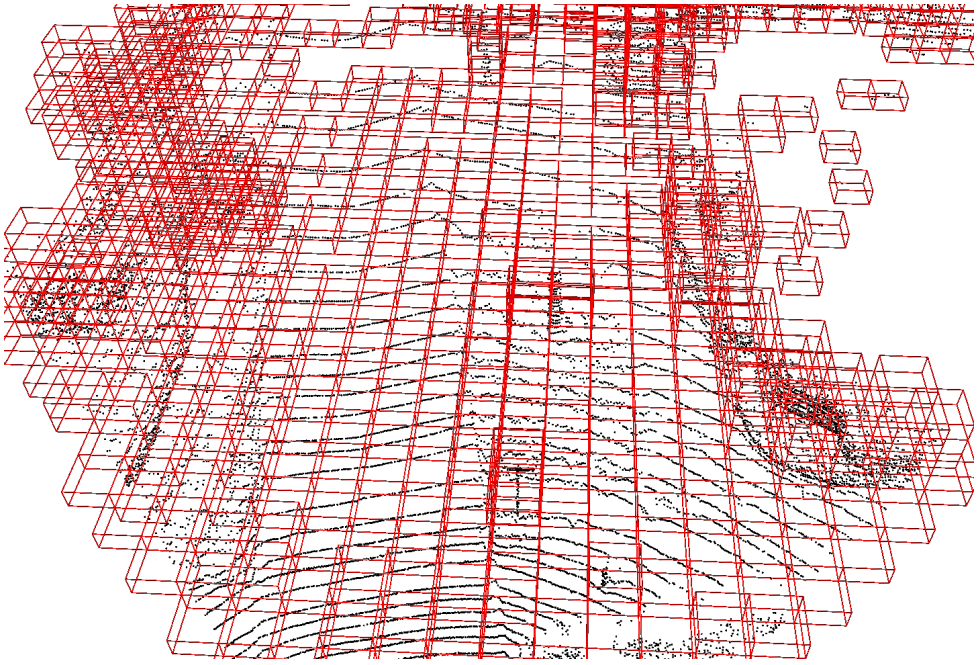


FIGURE 3.14 – Exemple de voxelisation, chaque cuboïde représente un voxel.

trouvé des utilisations dans d'autres domaines. L'emploi d'opérateurs symétriques a permis l'invariance aux permutations. De nombreuses variantes ont ensuite vu le jour, par exemple pour prendre en compte les composantes locales de manière hiérarchique (QI et al., 2017b). On peut également citer des méthodes telles que DGCNN (WANG et al., 2019b), SpiderCNN (XU et al., 2018) ou KPConv (THOMAS et al., 2019) qui redéfinissent et/ou adaptent des opérateurs comme les convolutions aux nuages de points.

Ces représentations ne sont pas exclusives et la plupart des méthodes actuelles combinent différentes formes afin de profiter des avantages de chacune.

Méthodes appliquées aux grilles

Le capteur LiDAR est utilisé dans une grande variété de domaines tels que la reconstruction d'édifices ou la topographie. Cependant, son usage le plus connu concerne l'automobile.

La détection 3D a énormément profité des avancées réalisées en détection sur l'image RGB. BirdNet (BELTRAN et al., 2018) est notamment une adaptation de Faster-RCNN (REN et al., 2017) tandis que Complex YOLO (SIMON et al., 2019) et YOLO3D (ALI et al., 2018) se sont fortement inspirées du réseau YOLO (REDMON et al., 2016). L'entrée n'est toutefois plus une image RGB mais une pseudo-image en vue de dessus créée manuellement par voxelisation, uniquement sur le plan horizontal. Les canaux peuvent ainsi représenter l'altitude maximale des points dans chaque voxel, la réflectance maximale ou la densité de points dans chaque cellule.

L'une des premières grandes approches permettant la détection d'objets à partir du LiDAR seul fut VoxelNet (ZHOU et TUZEL, 2018) dont l'architecture globale est illustrée figure 3.15.

La première phase de la méthode consiste en une voxelisation (une transformation de l'espace en grille régulière) 3D. Le principal point critique se trouve dans l'encodage des cellules. En effet, contrairement aux voxelisations classiques utilisant un encodage fondé sur une ou plusieurs grandeurs sélectionnées manuellement, ici chaque voxel est représenté par un vecteur obtenu par un *Voxel Feature Encoder* (VFE) appliqué aux points 3D en son sein. Ce VFE est un sous-réseau inspiré de PointNet permettant de traiter les points sans traitement supplémentaire. Des couches de convolutions 3D sont ensuite chargées d'aplatir sur la dimension verticale la grille de voxels encodés afin d'obtenir une *feature map* correspondant à une vue orthographique (Bird's Eye View

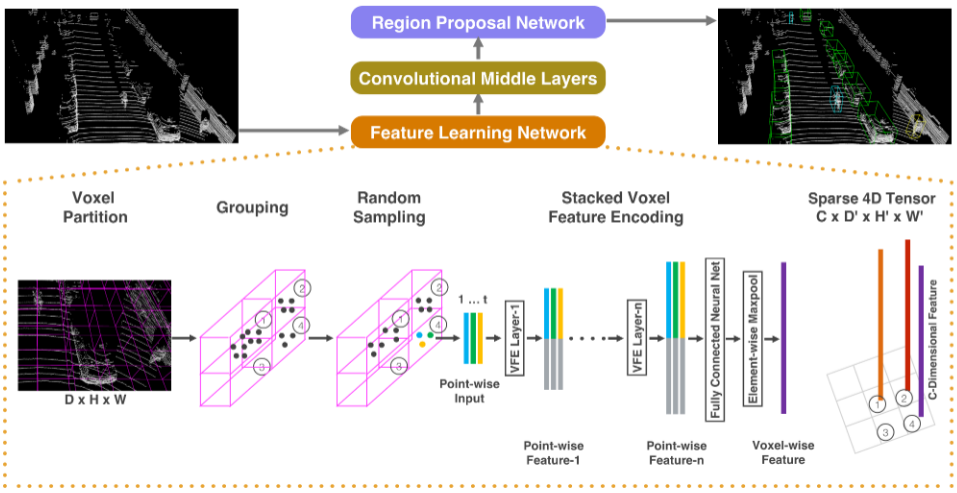


FIGURE 3.15 – Architecture globale du VoxelNet (ZHOU et TUZEL, 2018). Les points sont groupés par voxel qui est encodé par un réseau. Les voxels de la grille 3D sont ensuite assemblé par un ensemble de couches de convolutions 3D puis traitées par un *Region Proposal Network*.

ou BEV). Cette *feature map* est enfin utilisée dans un réseau de propositions (RPN) chargé de délivrer les résultats finaux.

L'un des inconvénients majeurs de cette méthode provient de son coût en mémoire et en temps de calcul. En effet, les voxels vides sont déclarés explicitement dans la mémoire de l'ordinateur comme les voxels occupés (par au minimum 1 point), bien qu'ils n'apportent que peu d'informations. Les convolutions 3D, qui sont déjà coûteuses doivent passer sur tous les voxels, y compris les voxels vides. VoxelNet fut cependant pionnière et la plupart des méthodes s'en sont inspirées et ont amélioré ou modifié certains aspects. L'un des exemples les plus connus est SECOND (YAN, MAO et LI, 2018) (*Sparsely Embedded CONvolutional Detection*). L'un des apports majeurs de la méthode est l'utilisation de structures creuses et de convolutions creuses (GRAHAM, 2014 ; GRAHAM, 2015 ; GRAHAM et van der MAATEN, 2017). D'une part, seuls les voxels occupés sont déclarés (référencés par leur vecteur de *features* et leur position dans la grille 3D), d'autre part, les convolutions 3D ne sont réalisées que sur les voxels valides. Ces deux ajouts ont permis un gain de vitesse et de performances, car les voxels vides superflus n'impactaient pas les résultats finaux.

Le réseau PointPillars (LANG et al., 2019) s'affranchit de l'utilisation de convolutions 3D par l'emploi de voxels « colonnes », la discrétisation n'étant pas effectuée sur l'axe vertical. La grille obtenue est alors directement traitée comme une image, permettant un gain de vitesse, au détriment cependant de la performance de détection. En effet, la verticalité des points n'est plus intrinsèquement codée dans la voxelisation, le VFE doit apprendre les relations entre plus de points ce qui est une potentielle source d'erreur.

D'autres modifications ont été apportées au paradigme créé par VoxelNet. Voxel-FPN (KUANG et al., 2020) s'inspire des *Feature Pyramid Networks* (LIN et al., 2017a), architectures utilisées pour extraire des informations à différentes résolutions spatiales et à différents niveaux sémantiques. Les auteurs définissent plusieurs voxelisations du nuage à différentes résolutions. Chaque voxelisation est encodée dans un premier temps indépendamment des autres. Dans un second temps, les *feature maps* obtenues sont agrégées puis les prédictions sont estimées sur différentes résolutions spatiales.

Méthodes appliquées aux points 3D

Les méthodes présentées jusqu'ici se fondent principalement, voire exclusivement sur une phase de voxelisation, dont la résolution peut impacter la qualité du résultat final, mais aussi le temps d'exécution. Certaines méthodes se fondent sur du traitement de nuages de points bruts. La majorité d'entre elles sont des détecteurs en deux phases, à savoir une partie proposition de région et une partie amélioration/filtrage des propositions. PointRCNN (Shi, Wang et Li, 2019) réalise en premier lieu une segmentation du nuage de points afin d'extraire les points pouvant appartenir aux objets cherchés et génère une proposition pour chaque point sélectionné. La seconde phase utilise les propositions obtenues lors de la première phase pour mieux exploiter le nuage de points et ainsi affiner les résultats. L'architecture globale est indiquée dans la figure 3.16.

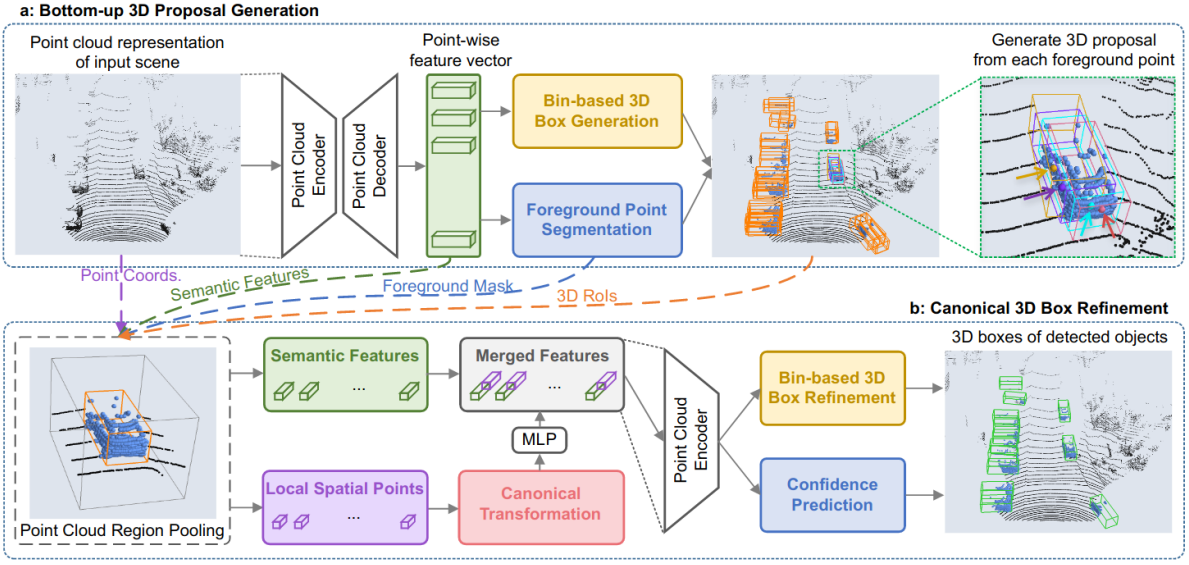


Figure 3.16 – Architecture globale du PointRCNN (Shi, Wang et Li, 2019). La première phase (en haut) réalise une segmentation et génère une proposition 3D pour chaque point segmenté. La seconde phase extrait, à partir des propositions générées, des éléments de la première phase pour affiner les résultats finaux.

Fast PointRCNN (Chen et al., 2019b) et PV-RCNN (Shi et al., 2020) s'inspirent de VoxelNet et de SECOND pour leur première phase. Les *features* pour la seconde phase ne sont plus extraites des points mais des *feature maps* calculées en premiers lieux.

Part-A^2 (Shi et al., 2019) se focalise sur les véhicules. La première phase réalise deux tâches, la segmentation des points 3D et pour chaque point positif, l'assignation de la partie du véhicule qu'il représente (par exemple avant gauche au sol, arrière droit sur le toit, etc.). Cette estimation est ensuite extraite puis utilisée pour la seconde phase.

Les premières méthodes de détection d'objets (RGB 2D ou 3D) ont utilisé des formatages à base de boîtes a priori : les boîtes de dimensions et d'orientation données sont positionnées dans l'espace de recherche (par exemple suivant une grille ou pour chaque point 3D). L'objectif est alors de déterminer lesquelles de ces boîtes peuvent potentiellement contenir une cible d'intérêt et d'ajuster cette boîte à l'obstacle si elle est digne d'intérêt. Les méthodes dites sans ancres (*anchor-free*) telles que CornerNet (Law et Deng, 2018) ou CenterNet (Duan et al., 2019) ont fait leur apparition par la suite. Ces méthodes assimilent la détection d'objets à de la détection de points clés à l'aide de représentations simplifiées, libérant ainsi la contrainte de la définir manuellement des a priori. OHS (Chen et al., 2019a) ou CenterPoint (Yin, Zhou et Krähenbühl, 2020) s'inspirent fortement de ce paradigme.

3.3.3 Détection 3D par fusion caméra-LiDAR

Cette section se focalise sur l'utilisation de capteurs LiDAR et de caméras RGB pour la détection d'obstacles. L'un des arguments en faveur de cette fusion est l'exploitation des avantages des deux capteurs : les nuages de points sont très précis, mais leur potentiel d'identification diminue grandement avec la distance tandis que les caméras sont reconnues pour leurs capacités de classification, mais le processus d'obtention de l'image rend difficile la localisation précise dans l'espace. Cependant, leur mise en place est plus délicate du fait de la gestion des deux capteurs (calibration pour projection, synchronisation, etc.). De plus, l'emploi de deux données augmente en conséquence le temps de calcul. Enfin, les données provenant de ces deux capteurs sont très différentes. De ce fait, les architectures développées comportent souvent plusieurs branches. Deux principaux groupes de méthodes se démarquent dans la littérature : les méthodes à flux parallèles et les méthodes séquentielles.

Méthodes à flux parallèles

Ces méthodes utilisent généralement les images et les nuages de points comme deux entrées qui sont traitées simultanément par le système. Ainsi, MV3D (Chen et al., 2017b) et AVOD (Ku et al., 2018) traitent simultanément les images et les nuages de points pour extraire de chaque modalité des propositions intermédiaires qui sont par la suite traitées et fusionnées afin d'obtenir des prédictions finales.

Pour MV3D (Chen et al., 2017b), l'une des premières méthodes d'envergure sur le *benchmark* KITTI 3D Object Detection, des propositions 3D sont générées dans un premier temps à partir d'une image LiDAR *Bird Eye's View*. Ces propositions sont projetées sur trois vues : une vue représentant le nuage de points en vue de dessus (BEV), une représentation frontale du nuage de points (projection sphérique), et l'image RGB synchronisée avec le nuage de points. Pour chaque vue, les *features* localisées à l'intérieur des propositions sont extraites. Ces *features* extraites sont fusionnées avec celles des autres vues. Les données issues de la fusion sont par la suite utilisées pour estimer les prédictions finales. AVOD (Ku et al., 2018) étend l'idée en projetant toutes les boîtes a priori sur les *feature maps* en haute-définition provenant de la vue LiDAR BEV et de l'image RGB.

Reprenant l'architecture globale de VoxelNet, les auteurs de MVX-Net (Sindagi, Zhou et Tuzel, 2019) ont proposé deux méthodes pour ajouter de l'information issue de la caméra aux voxels 3D. La méthode *PointFusion* consiste en l'ajout d'informations issues de l'image au niveau des points 3D, le *Voxel Feature Encoder* prend donc également en entrée des *features* issues de la caméra. La méthode *VoxelFusion* intègre les informations RGB au niveau du voxel, le VFE n'ayant que les points 3D comme entrée pour des caractéristiques mises en commun à partir de cartes de caractéristiques.

La méthode décrite dans (Liang et al., 2018) exploite les convolutions continues (Wang et Deng, 2018) pour fusionner les *features* de chaque capteur. Les *features* provenant de l'image sont transformées et fusionnées avec les *features* issues de BEV LiDAR à différentes échelles pour obtenir les prédictions. Les auteurs de (Liang et al., 2019) améliorent cette architecture en intégrant des tâches connexes telles que l'estimation du terrain et l'estimation de la profondeur pour dynamiser le sous-réseau de raffinement 3D.

Méthodes séquentielles

Les techniques de traitement d'images RGB ont grandement évolué ces dernières années avec l'application des réseaux de neurones profonds. De ce fait, une fois paramétrées, ces méthodes peuvent servir d'a priori forts pour d'autres tâches moins adaptées aux caméras. Les méthodes présentées dans ce paragraphe sont qualifiées de séquentielles, car elles exploitent les résultats d'un réseau préexistant et pré-entraîné en vue d'améliorer des résultats d'algorithmes ne fonctionnant à l'origine que sur des nuages de points.

Frustum PointNet (Qi et al., 2018), Frustum Convnets (Wang et Jia, 2019) et PointFusion (Xu, Anguelov et Jain, 2018) exploitent les résultats d'un détecteur 2D pour extraire les points dont les projections se situent à l'intérieur de chaque détection et ensuite réduire l'espace de recherche pour chaque objet détecté par le détecteur d'images. Pour chaque détection 2D dans l'image, les points 3D dont les projections sont localisées dans la boîte 2D sont conservés. Frustum PointNet (Qi et al., 2018) exploite un PointNet (Qi et al., 2017a) pour estimer la boîte 3D représentant l'objet visé. La méthode PointFusion (Xu, Anguelov et Jain, 2018) ajoute des informations issues de l'image RGB à l'aide d'un ResNet (He et al., 2016) afin de mieux re-contextualiser le nuage de points obtenu. La méthode Frustum Convnets (Wang et Jia, 2019) n'utilise que le nuage de points. Cependant, contrairement à Frustum PointNet qui traite l'intégralité du frustum en une fois, Frustum ConvNet découpe le nuage de points obtenu selon l'axe du frustum comme le montre la figure 3.17. Chaque tranche est traitée indépendamment des autres par un PointNet. Les sorties Les sorties des cellules sont ensuite rassemblées. Le « vecteur » est enfin traité par un réseau convolutif.

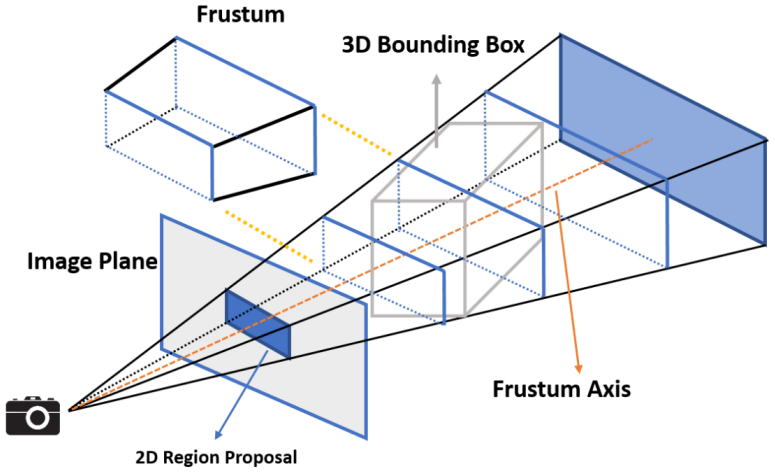


Figure 3.17 – Fonctionnement de Frustum Convnets (Wang et Jia, 2019). Une proposition de région dans le plan image obtenu par un détecteur 2D permet d'extraire le frustum 3D correspondant. Le frustum est découpé en blocs le long de son axe. Chaque bloc est traité dans un premier temps indépendamment des autres puis, dans un second temps, mis en contexte avec les autres blocs du frustum pour localiser l'objet.

IPOD (Yang et al., 2018) exploite d'abord une segmentation sémantique issue d'une image RGB afin de générer des propositions aux niveaux des points appartenant à des obstacles potentiels. Ces propositions sont ensuite utilisées pour extraire les points adéquats et créer les boîtes englobantes correspondantes. Avec PointPainting (Vora et al., 2020), les auteurs illustrent l'amélioration de plusieurs détecteurs LiDAR par l'intégration d'informations sémantiques provenant d'une carte de segmentation sur le nuage de points d'entrée.

3.3.4 Détection 3D par images 2D

Les caméras RGB sont omniprésentes et peu onéreuses. Les images capturées affichant une représentation plane d'une scène 3D, de nombreuses recherches ont eu lieu sur la récupération d'information spatiale et d'identification de cibles. La détection d'objets 3D par l'intermédiaire d'images est un problème mal posé en raison de la perte d'informations de profondeur dans le processus de création d'images. Ainsi, les méthodes appartenant à cette catégorie sont généralement moins précises que celles fondées sur le LiDAR. Nous nous focalisons dans ce paragraphe sur les méthodes monoculaires, c'est-à-dire n'exploitant qu'une seule caméra. Les méthodes présentées font usage d'un certain nombre d'hypothèses sur les cibles à détecter. Ces a priori peuvent

revêtir plusieurs formes, des hypothèses sur la forme globale de la cible, voire l'utilisation de modèles 3D.

Mono3D (Chen et al., 2016) fut l'une des premières méthodes d'envergure. Les auteurs définissent dans un premier temps un ensemble de boîtes 3D, de dimensions, de positions et d'orientations variables dans l'espace de recherche qui sont ensuite projetées dans l'image afin d'obtenir les régions 2D correspondantes. En parallèle, plusieurs résultats intermédiaires sont estimés à partir de l'image d'origine (segmentation sémantique et segmentation d'instances). Pour chacune des propositions, un score est déterminé. Ce score résulte de la somme de fonctions représentant notamment des adéquations de forme, de contexte ou de sémantique avec les cibles recherchées. Les propositions sont ensuite triées en fonction de leur score et filtrées (Figure 3.18). Les propositions restantes sont utilisées dans un réseau de neurones en tant que régions d'intérêt afin d'estimer les caractéristiques des boîtes 3D correspondantes. La méthode requiert au préalable un certain nombre de prérequis (les réseaux de segmentation) et est relativement lente.

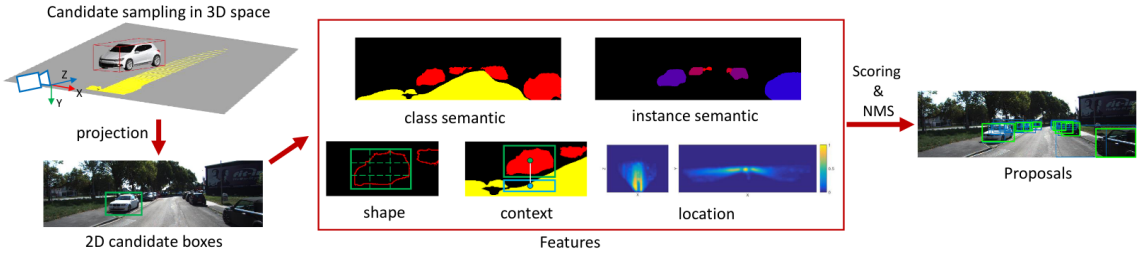


Figure 3.18 – Illustration de Mono3D (Chen et al., 2016). Des boîtes 3D sont générées puis projetées dans le plan image afin d'obtenir une région 2D. Chaque boîte/région est évaluée suivant différents critères (sémantique, forme ...) et se voit attribuer un score.

Deep3DBox (Mousavian et al., 2017) étend les capacités d'un détecteur 2D pour estimer des paramètres de boîtes 3D. L'une des principales contributions de la méthode concerne l'estimation de l'orientation. Les auteurs ne procèdent pas directement à une régression des valeurs angulaires. Le cercle trigonométrique est partitionné en sections. L'estimation de la rotation est ainsi transformée en un problème de classification, auquel est rajoutée l'estimation du décalage entre le « centre » de la section et l'angle visé.

Deep MANTA (Chabot et al., 2017) vise la détection fine des véhicules, à l'aide de modèles 3D issus de la CAO (Conception Assistée par ordinateur). Pour ce faire, une première phase est chargée de d'effectuer une détection 2D des véhicules dans l'image. Cette détection ne se résume pas à la boîte et à la classification. En effet, le réseau est également chargé de réaliser d'autres tâches plus fines telles que l'estimation de la position et la visibilité des parties du véhicule (roues, portières, etc.). Une seconde phase est chargée de procéder à l'estimation 3D. D'abord, le meilleur modèle 3D de la base CAO est sélectionné. Ce modèle est ensuite mis en correspondance avec l'image RGB et les paramètres estimés au cours de la première phase pour estimer les prédictions finales.

Cette approche par recherche de points clés (*keypoint detection*) est notamment reprise dans la méthode RTM3D (Li et al., 2020) dont l'architecture est complètement convolutive. Selon les dires des auteurs, la méthode est temps réel, contrairement à la majorité des algorithmes présentés. ROI-10D (Manhardt, Kehl et Gaidon, 2019) exploite simultanément un détecteur 2D et un réseau d'estimation de profondeur. Connaissant les paramètres de calibrage intrinsèque, la fonction est chargée de minimiser la distance entre les sommets de la boîte estimée par le réseau et ceux de la vérité. En complément de cet objectif, un système est développé d'estimation de maillage 3D dense à partir d'une base CAD de véhicules.

Les auteurs de (Wang et al., 2019a) réalisent une estimation de profondeur à partir de l'image d'origine. Connaissant les paramètres intrinsèques des caméras, les cartes de profondeur obtenues sont transformées en nuages de points 3D. Les algorithmes applicables uniquement sur

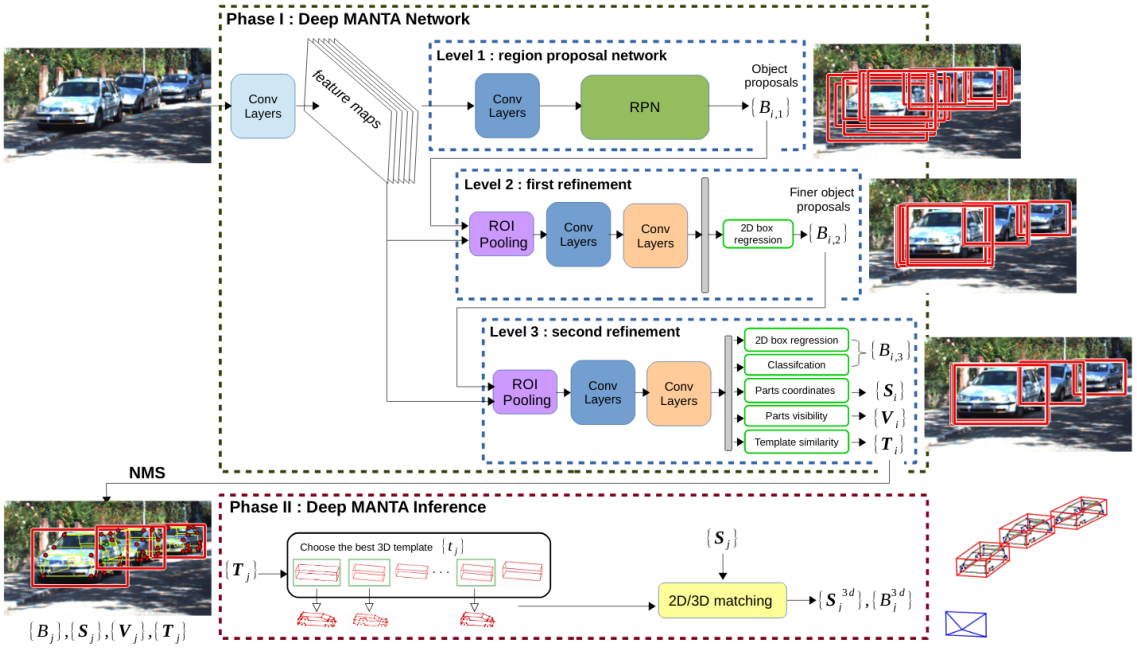


FIGURE 3.19 – Fonctionnement de Deep MANTA (CHABOT et al., 2017). La première phase doit estimer des régions dans l'image et les parties des voitures au sein de chaque région. Pour chaque proposition, la seconde phase sélectionne un modèle 3D CAD puis estime la pose du modèle 3D en fonction des parties estimées sur l'image.

ce type de données sont alors exploitables. Pour un même algorithme de détection, les résultats finaux dépendent alors fortement de la qualité du nuage de points créé. Les auteurs ont ainsi analysé les résultats de AVOD (KU et al., 2018) et de Frustum-PointNet (QI et al., 2018) sur des cartes de profondeur obtenues à l'aide de l'algorithme DORN (FU et al., 2018) (cas monoculaire) ou PSMNet (CHANG et CHEN, 2018). Pour améliorer la qualité du nuage de points estimé, (YOU et al., 2019) exploite les données d'un LiDAR bas coût et à faible résolution verticale. Les points 3D LiDAR servent alors de points de contrôle pour déformer le nuage de points estimé par vision monoculaire et obtenir une représentation plus proche de la réalité.

Dans (RODDICK, KENDALL et CIPOLLA, 2018), une grille 3D est créée au préalable. Chaque cellule est projetée dans l'image. Le vecteur de chaque cellule est ainsi calculé à partir des caractéristiques issues de l'image RGB. Cette grille 3D est « aplatie » sur l'axe vertical pour réduire les coûts en calcul, puis traitée comme une BEV.

Le papier sur MonoDIS (SIMONELLI et al., 2019) décrit une architecture présente dans d'autres travaux. Un premier réseau de détection 2D, constitué d'un *Feature Pyramid Network* (LIN et al., 2017a) délivre des prédictions dans l'image qui sont ensuite exploitées dans un sous-réseau chargé de l'estimation des paramètres de la boîte 3D. La particularité de la méthode provient de la gestion de la fonction de coût. À partir des paramètres estimés par le réseau, contenant notamment les paramètres de position, de dimensions et d'orientation, une représentation intermédiaire est calculée. Les auteurs ont choisi d'utiliser comme représentation les sommets de la boîte produite. Cependant, la fonction de coût n'est pas estimée en une seule fois, l'argument étant que le réseau se voit octroyer la possibilité de diriger l'optimisation sur tous les degrés de liberté. Les auteurs ont donc fait le choix de décomposer la fonction en plusieurs sous-fonctions, chaque sous-fonction n'exploite qu'une partie des paramètres estimés. Ainsi, une première peut n'utiliser que les estimations sur les positions (et assigner la vérité terrain aux autres paramètres), tandis qu'une autre se focalise uniquement sur les paramètres d'orientation. L'optimisation finale est une somme d'optimisations disposant de moins de degrés de liberté. Les auteurs ont démontré une bien meilleure convergence menant également à de meilleurs résultats.

Les auteurs de (Shrivastava et Chakravarty, 2020) entraînent une architecture de type encodeur-décodeur pour l'estimation de la profondeur. Les espaces latents estimés entre l'encodeur et le décodeur sont ensuite utilisés pour estimer les régions 2D dans l'image et les poses des boîtes 3D dans l'espace.

Dans (Brazil et Liu, 2019), un DenseNet (Huang et al., 2017) génère une carte de *features* à partir d'une image d'entrée. Une architecture à deux branches est alors définie. La première branche utilise des convolutions régulières pour estimer les caractéristiques dans l'ensemble de l'image. La seconde branche repose sur l'hypothèse selon laquelle les objets proches sont en général localisés dans la zone basse de l'image. Partant de ce constat, la *feature map* est divisée en tranches régulières horizontales. Un ensemble de convolutions est défini, chaque opérateur n'étant appliqué qu'à une seule tranche, correspondant selon les auteurs à une profondeur définie, d'où le nom de « *Depth Aware Convolutions* ». Les deux branches sont ensuite pondérées et fusionnées en vue d'une optimisation pour estimer les boîtes 3D.

Enfin, les méthodes dites « *anchor-free* » (Law et Deng, 2018 ; Duan et al., 2019 ; Zhou, Wang et Krähenbühl, 2019 ; Zhou, Zhuo et Krahenbuhl, 2019) ont aussi leurs usages dans la vision 3D monoculaire. Ainsi, en plus d'introduire la détection des boîtes 2D par la détection de leur centre, les auteurs de (Zhou, Wang et Krähenbühl, 2019) ont également tenté d'estimer les paramètres de la boîte 3D correspondante. SMOKE (Liu, Wu et Tóth, 2020) se focalise exclusivement sur la détection 3D. Le *keypoint* recherché n'est plus le centre de la boîte 2D, mais la projection du centre de la boîte 3D dans l'image.

3.4 Adaptation de domaine LiDAR

L'adaptation de domaine est le champ de recherche traitant de l'utilisation de méthodes d'apprentissage pour une tâche donnée dans des situations différentes de celles sur lesquelles elles ont été entraînées. La définition même de « domaine » est floue et peut varier en fonction des cas étudiés. Des domaines peuvent diverger par les zones géographiques capturées, les conditions météorologiques ou les capteurs utilisés par exemple. L'adaptation de domaine est un sujet de recherche très actif sur le traitement d'images 2D. Cependant, cette problématique est encore très peu traitée en ce qui concerne les nuages de points et les traitements 3D.

On nomme dans cette section « source » les données issues du domaine d'entraînement (dont on dispose des annotations) et « cible » les données de test. De plus, nous nommons « réseau de perception » le réseau chargé de réaliser la tâche d'intérêt. Il s'agit majoritairement de réseaux de segmentation sémantique. Les méthodes d'adaptation de domaine peuvent être groupées selon 3 catégories principales : la représentation de données invariante au domaine, l'association de domaines et l'apprentissage de caractéristiques invariantes au domaine.

Représentation de données invariante au domaine Les méthodes appartenant à ce groupe effectuent un prétraitement sur les données sources pour générer une représentation choisie manuellement qui ne varie pas avec le domaine d'utilisation avant le traitement par le réseau de neurones. À l'inférence, les données cibles subissent ce même prétraitement avant le réseau. Des méthodes comme (Triess et al., 2019) ou (Shan et al., 2020) reposent sur la projection de nuages de points sur une vue frontale. Un sur-échantillonnage sur cette vue à l'aide de réseau plus orienté traitement d'image permet alors d'obtenir un nuage plus dense avant traitement par un réseau de perception. Les auteurs de (Piewak, Pinggera et Zöllner, 2019) analysent les performances de leur architecture de segmentation sémantique sur des nuages de 32 et 128 nappes afin de montrer l'invariance de l'encodage utilisé. Complete & Label (Yi, Gong et Funkhouser, 2021) utilise un réseau de complétion pour estimer à partir d'un nuage de point la surface sous-jacente. Cette reconstruction sert alors de domaine invariant pour un réseau de segmentation comme le montre la figure 3.20.

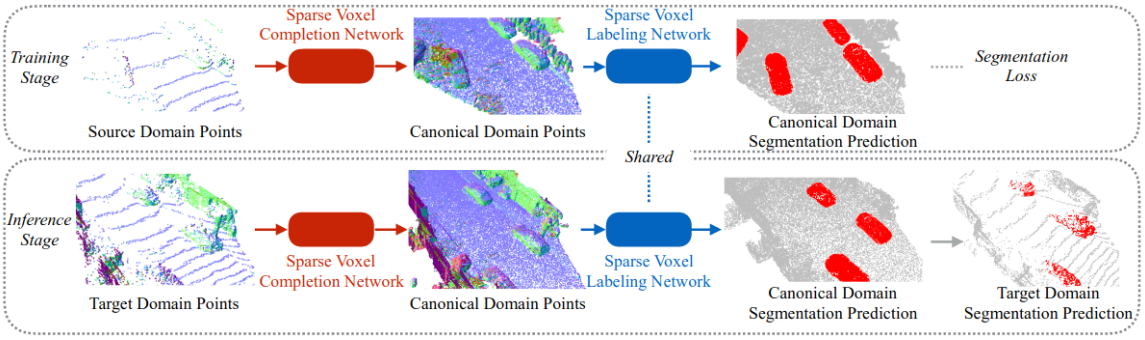


FIGURE 3.20 – Fonctionnement de Complete & Label (YI, GONG et FUNKHOUSER, 2021)

Association de domaines L'association de domaine consiste à transformer avant le réseau de neurones les données sources de sorte à ressembler à des données cibles. Lors de la phase de test, le réseau ayant appris sur une simulation de la distribution cible, les données cibles sont transmises sans prétraitement au réseau. Ce groupe inclut notamment les méthodes visant le passage de données simulées vers un domaine réel. Des méthodes comme (SALEH et al., 2019) ou (SALLAB et al., 2019) reposent sur l'utilisation de GAN (*Generative Adversarial Networks*), en particulier l'architecture CycleGAN (ZHU et al., 2017), pour générer à partir de vues synthétiques des images BEV plus réalistes qui sont ensuite utilisées par un réseau de perception. (LANGER et al., 2020) vise une adaptation capteur à capteur. Pour cela, les auteurs réalisent dans un premier temps un nuage dense par accumulation des nuages de points 3D d'une séquence. Les spécifications du capteur cible sont alors utilisées pour générer un jeu semi-synthétique à partir du nuage accumulé. Un réseau de segmentation entraîné sur le domaine source est alors ré-entraîné pour s'ajuster aux statistiques du domaine cible.

Apprentissage de caractéristiques invariantes au domaine Les méthodes de ce groupe ressemblent aux méthodes de représentation invariante. Lors de l'entraînement, les données sources et cibles sont traitées par un même réseau d'extraction de caractéristiques. Afin de conditionner l'apprentissage de ces caractéristiques, on introduit une composante d'alignement chargée de faire correspondre les espace intermédiaires sources et cibles. Il s'agit de la principale différence avec des méthodes à représentation invariante : les données cibles sont intégrées à l'entraînement par cette composante d'alignement. Cependant, seules les caractéristiques sources sont utilisées par le réseau de perception. Lors de la phase de test, les données cibles sont d'abord traitées par le réseau d'extraction puis par le réseau de perception. Par exemple, pour SqueezeSegV2 (WU et al., 2019), l'entraînement du réseau est réalisé sur deux fronts. D'un côté la segmentation est apprise à l'aide des données étiquetées du jeu de données synthétiques GTA-LIDAR et une fonction de coût pour la segmentation *Focal loss*. De l'autre côté, ce même réseau est appliqué sur des données réelles non annotées, les sorties obtenues à parties des données réelles et des données synthétiques sont alors alignées à l'aide d'une fonction de coût sur l'alignement géodésique. LiDARNet (JIANG et SARIPALLI, 2020) s'inspire du fonctionnement des GAN. L'extracteur de caractéristiques prend le rôle d'un générateur. L'objectif est alors de générer des caractéristiques suffisamment proches de celles générées par le domaine cible pour qu'un réseau devant discriminer les deux domaines ne puisse plus faire la différence.

3.5 Conclusions et impacts sur les travaux menés

3.5.1 Conclusions

Contrairement à l'image qui dispose toujours d'une structure régulière, la structure à adopter pour le traitement de nuages de points 3D avec des réseaux de neurones ne fait pas consensus.

L'utilisation d'opérateurs que l'on pourrait qualifier de standards pour le traitement d'images (opérateurs déjà implémentés dans la majorité des *frameworks* d'apprentissage) comme les convolutions impliquent le choix d'un formatage régulier tandis que l'exploitation directe des nuages requiert des opérateurs et traitements particuliers et moins communs.

En termes de détection 3D, les méthodes n'utilisant que les nuages de points et les méthodes axées fusion de capteurs sont en concurrence. Les premiers travaux tels que MV3D sont axés sur de la fusion de données de capteurs différents. Toutefois, VoxelNet et SECOND ont montré qu'il était possible d'obtenir des résultats équivalents, voire meilleurs avec le capteur LiDAR seul, pour la portée étudiée (moins de 80 m), sans les contraintes liées à la fusion de capteurs telles que la nécessité de calibration extrinsèque. Les méthodes axées LiDAR seul sont donc bien plus prédominantes dans la littérature. Les principales avancées ont grandement porté sur les architectures employées pour exploiter au mieux l'information 3D. Les techniques axées fusion exploitent de plus en plus les résultats des réseaux axés LiDAR. Les pistes de recherche se focalisent de plus en plus sur les façons d'implanter l'information issue de l'image dans des architectures 3D préexistantes.

Les méthodes fondées uniquement sur la caméra ne rivalisent pas en termes de performances avec les techniques exploitant un capteur de distance, au point que les méthodes monoculaires ne sont comparées qu'entre elles. Toutefois, leur intérêt n'est pas limité. En effet, les caméras ont de grandes chances d'être intégrées aux véhicules autonomes, notamment pour tout ce qui concerne la détection de signalisation. De ce fait, disposer d'un détecteur courte distance « d'appoint » reste toujours un plus.

Toutes les méthodes présentées jusqu'à présent sont généralement entraînées et testées sur différents partitionnements d'un même jeu de données. Les données de validation restent donc proches du domaine d'entraînement. Néanmoins, cela ne permet pas d'identifier si ces réseaux sont exploitables dans des contextes différents, surtout avec différents capteurs et donc facilement réutilisables. L'adaptation de domaine appliquée aux nuages de points 3D est un sujet peu traité dans la littérature. Cet aspect fut une des principales directions suivies au cours de cette thèse.

3.5.2 Positionnement du chapitre 4 « Pseudo-cartes d'occupation »

Le chapitre 4 « Pseudo-cartes d'occupation » présente un détecteur n'utilisant que le LiDAR pour de la détection en BEV de véhicules. L'idée est d'expérimenter des approches de détection plus graphiques. D'une part, la première volonté est d'obtenir « littéralement » une BEV de la scène, c'est-à-dire une vue exposant quelles parties de l'espace sont exposées par une catégorie d'objets particuliers. En effet, dans les approches de détection courantes, les résultats du réseau sont formatés de sorte que chaque pixel de la *feature map* de sortie corresponde à une proposition de position et de taille. Dans notre cas, nous visons plus une sorte de carte d'occupation, dont chaque pixel caractérise uniquement sa zone correspondante de l'espace. Une proposition consiste alors à un ensemble de pixels. Une telle représentation est plus graphique et plus facilement interprétable, mais moins optimale dans la mesure où l'absence d'un élément peut perturber l'estimation de la forme de la cible. La seconde volonté est d'obtenir une représentation simple et paramétrique des obstacles permettant une estimation rapide des paramètres de la boîte.

3.5.3 Positionnement du chapitre 5 « Détection d'obstacles résistant aux changements de résolution »

Les méthodes présentées dans ce chapitre, qu'elles utilisent un seul ou les deux capteurs parviennent à obtenir d'excellentes performances sur les benchmarks associées à leur jeu d'entraînement. Néanmoins, les données d'entraînement restent assez représentatives des données utilisées pour l'évaluation. Cela équivaut à entraîner un réseau pour une configuration et des cas d'utilisation précis et connus. Cependant, peu d'études se focalisent sur l'usage de réseaux pré-

entraînés sur des données très différentes. Cela peut concerner un changement sur les capteurs utilisés ou sur les zones géographiques et les environnements capturés.

Les travaux menés durant cette thèse se sont orientés suivant deux directions opposées à celle susmentionnée, à savoir comment obtenir des détections pertinentes avec un minimum de données 3D et comment rendre un détecteur exploitable sur plusieurs configurations de capteurs. Ce point, traité dans le chapitre 5 « Détection d'obstacles résistant aux changements de résolution », est un aspect crucial pour le monde industriel dans la mesure où un nouvel équipement sur un véhicule n'induira pas forcément de repartir à zéro pour rendre un véhicule utilisable (acquisition des données → annotation des données → entraînement du/des modèle(s)).

Cet aspect « variations du capteur LIDAR » implique également l'utilisation de capteurs de résolution plus faible par rapport au capteur utilisé en apprentissage. Lorsque le nuage de points est dense, il est possible d'estimer de manière directe les paramètres des obstacles. Ce n'est plus forcément le cas lorsque le nuage d'entrée devient plus épars. Le chapitre 5 propose une étude de cas d'usages au travers de trois réseaux et introduit des techniques pour permettre d'améliorer les performances sur des situations inconnues par le réseau. En outre, la méthode présentée dans le chapitre 4 repose sur un réseau de segmentation et l'utilisation d'ellipses. Cette approche est plus difficilement transposable vers des réseaux de détections plus standards éprouvés par la communauté. De ce fait, le chapitre 5 présente également une nouvelle représentation des boîtes englobantes utilisable avec des architectures plus traditionnelles. Bien que l'objectif s'apparente à celui des méthodes d'adaptation de domaine, nous visons principalement une meilleure résilience face aux variations de domaine plutôt qu'un ajustement à un domaine précis.

Chapitre 4

Pseudo-cartes d'occupation pour la détection de véhicules

Sommaire

4.1 Introduction

La majorité des méthodes de détection 3D et BEV sur des nuages de points reposent sur l'utilisation d'ancres, c'est-à-dire des boîtes englobantes de dimensions, positions et orientations définies au préalable. Chaque ancre est une candidate et le rôle du réseau est, d'une part de sélectionner les propositions les plus pertinentes, et d'autre part d'ajuster les ancres sélectionnées aux cibles. Cette approche est héritée des réseaux de détection 2D dédiés à l'image. Dans l'image cependant, tous les objets, quelles que soient leurs catégories peuvent apparaître de tailles variables. Ainsi du fait de la perspective, un piéton au premier plan peut apparaître bien plus grand sur la capture qu'un camion éloigné. De ce fait, les propositions générées manuellement peuvent théoriquement correspondre à toutes les cibles. Les propositions sélectionnées sont celles qui possèdent déjà une grande proximité avec les cibles de la base de données. En détection BEV et 3D, les distorsions dues à la perspective n'existent pas. De ce fait, la pratique commune consiste à définir pour chaque classe ciblée une boîte de dimensions définies à l'avance. Le choix de ces dimensions est arbitraire, mais peut notamment être estimé par des moyens statistiques, par exemple en utilisant la moyenne des dimensions des cibles de ladite classe dans le jeu de données. Cette approche, bien que performante, réduit la flexibilité de la méthode, particulièrement si l'on tente d'exploiter des catégories différentes issues d'un autre jeu de données, les ancres devant être estimées à l'avance. De plus, le choix des seuils IoU utilisés pour définir si une ancre est positive ou non augmente le nombre d'hyperparamètres pouvant grandement influencer le déroulement des apprentissages.

Ce chapitre présente une première proposition de méthode pour pallier le problème de représentation des obstacles. Le nuage de points 3D d'entrée est dans un premier temps voxelisé puis est utilisé dans un réseau de segmentation. L'objectif du réseau de segmentation est de sortir une carte en BEV dont chaque pixel représente la probabilité d'occupation de son voxel respectif par une cible. La carte en elle-même représente uniquement l'état global de la scène, mais ne représente en rien l'état des cibles prises une par une. Nous représentons dans cette méthode les cibles par des ellipses. Ce choix occasionne une représentation des obstacles sur la carte de sortie permettant l'extraction des cibles et de leurs caractéristiques par l'intermédiaire de techniques de traitement d'image et d'optimisation. Ce processus est succinctement illustré sur la figure 4.1.

Nous travaillons dans ce chapitre uniquement sur des nuages de points et nous intéressons aux voitures. Les voitures constituent la classe prépondérante dans la majorité des jeux de données. Vues de dessus, leur forme convexe se rapproche de rectangles arrondis, voire d'ellipses. Ainsi, nous optons tout d'abord pour un encodage minimal qui permet déjà d'identifier des voitures mais qui ne serait pas suffisamment informatif pour décrire des classes plus complexes. Aussi, pour entamer notre étude, nous nous limitons à la détection des voitures. Il s'agira de détection BEV dans la mesure où les résultats extraits ne présentent pas d'informations sur la dimension verticale (altitude du centre et hauteur). Nous nous focalisons sur les données provenant de la base KITTI. En effet, à l'heure où ont été réalisés les travaux, l'objectif était d'obtenir une première approche alternative au problème de détection. L'aspect « portabilité » entre jeux de données n'est donc pas encore abordé.

4.2 Description de la méthode

La méthode décrite dans ce chapitre peut être décomposée en deux parties : l'estimation de la carte de probabilités à l'aide d'un réseau de segmentation et l'extraction des résultats.

Nous décrivons, dans un premier temps, le traitement du nuage de points d'entrée pour obtenir un formatage compatible avec le réseau de segmentation. Puis, nous détaillons le modèle utilisé pour représenter les obstacles sur les données de vérités de terrain utilisées par la partie segmentation. Ensuite, le choix des réseaux de segmentation étudiés sera détaillé. Enfin, nous expliquerons le processus utilisé pour extraire chaque obstacle indépendamment des autres à

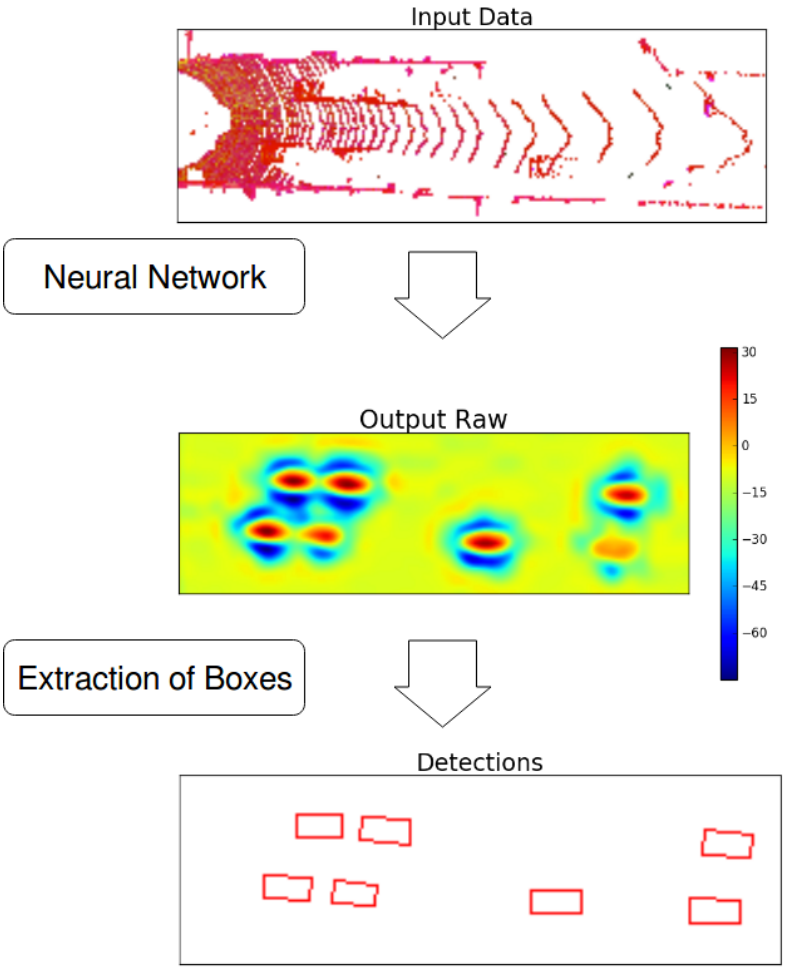


FIGURE 4.1 – Illustration du processus utilisé. Un réseau de neurones prend en entrée un nuage de points 3D formatés et retourne une carte. Cette carte est par la suite traitée pour extraire les boîtes englobantes des cibles.

partir de la carte de segmentation obtenue.

4.2.1 Formatage des données 3D

Le nuage de points est converti en une grille de voxels 3D. Cependant, les convolutions 3D constituent un coût calculatoire non négligeable, augmentant ainsi le temps d'exécution et l'espace mémoire nécessaire. La totalité des véhicules est positionnée sur un plan du sol à faible pente (pas forcément nulle toutefois). De ce fait, il est possible de simplifier la représentation du nuage en réduisant la représentation verticale. L'objectif revient à transformer le nuage de points en une pseudo-image en vue orthographique décrivant la scène. Une telle représentation permet d'utiliser des opérateurs et réseaux préexistants et optimisés, car plutôt spécialisés pour le traitement 2D.

Le système de coordonnées utilisé est le suivant : l'origine du repère est le centre du capteur LiDAR, l'axe x est orienté vers l'avant du véhicule, l'axe y vers la gauche, l'axe z vers le haut.

Chaque point 3D est décrit à travers ses coordonnées $(x,y,z)^T \in \mathbb{R}^3$ et sa réflectance r. Le nuage est d'abord transformé en une grille régulière de $n_x \times n_y \times n_z$ cellules, chaque cellule représente un pavé de $s_x \times s_y \times s_z$ m^3. Chaque voxel est représenté sous la forme d'un vecteur $[z_{min}, z_{max}, \bar{z}, r_{max}, d]^T \in \mathbb{R}^5$ avec $(z_{min}, z_{max}, \bar{z})$ les altitudes minimales, maximales et moyennes des

points contenus dans le voxel. r_{max} est la réflectance maximale et d le nombre de points. Une vue 3D des valeurs $(z_{min}, z_{max}, \bar{z})$ est illustrée figure 4.2 tandis que la figure 4.3 expose les vues de dessus générées pour les canaux (z_{max}, r_{max}, d).

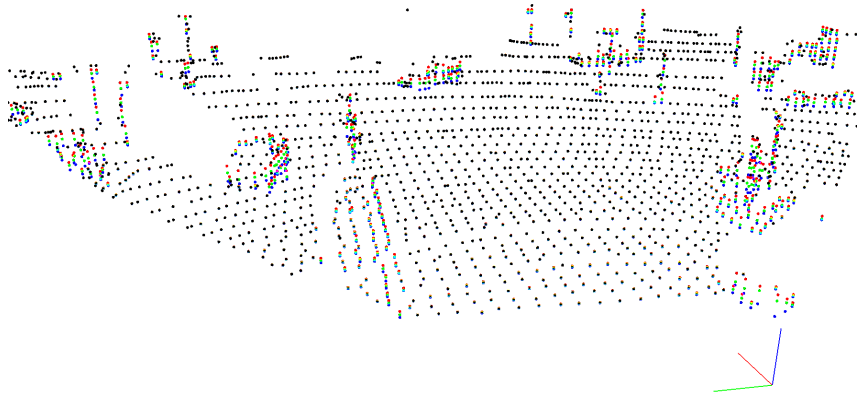


FIGURE 4.2 – Exemple de voxelisation. Les points rouges ont la hauteur maximale, les bleus ont la hauteur minimale, les verts la hauteur moyenne. Les cellules sont réparties régulièrement sur le plan XY.

La grille de voxels est, à ce stade, représentée par un tenseur de taille $n_x \times n_y \times n_z$ cellules, chaque cellule étant représentée par un vecteur de dimension 5. Cependant, comme dit précédemment, cette grille 3D est transformée en une pseudo-image 2D de (n_x, n_y) « pixels » constituée de $5 \times n_z$ canaux.

4.2.2 Représentation paramétrique des véhicules et vérité de terrain

Les obstacles sont très souvent représentés par des boîtes orientées. Si elles présentent certains avantages, particulièrement en ce qui concerne le calcul d'intersections entre deux boîtes orientées, nous nous focalisons sur l'utilisation des ellipses inscrites dans ces boîtes. Ce changement de représentation permet l'utilisation de certains algorithmes d'optimisation. De plus, le passage entre les deux représentations n'occasionne aucune perte d'information.

Un obstacle est défini par la position de son centre (x_c, y_c), sa largeur w, sa longueur l et son orientation θ. Pour les véhicules, nous supposons que l'annotation fut réalisée de sorte que $l > w$. On définit $a = \frac{l}{2}$ la demi-longueur qui est également le demi-grand axe de l'ellipse et $b = \frac{w}{2}$ la demi-largeur et le demi-petit axe. Ces variables sont illustrées sur la figure 4.4. De plus, concernant l'orientation, nous nous focalisons sur la direction de la cible, le sens pouvant être obtenu soit par analyse plus fine, soit par prise en compte du mouvement des cibles détectées sur les acquisitions précédentes.

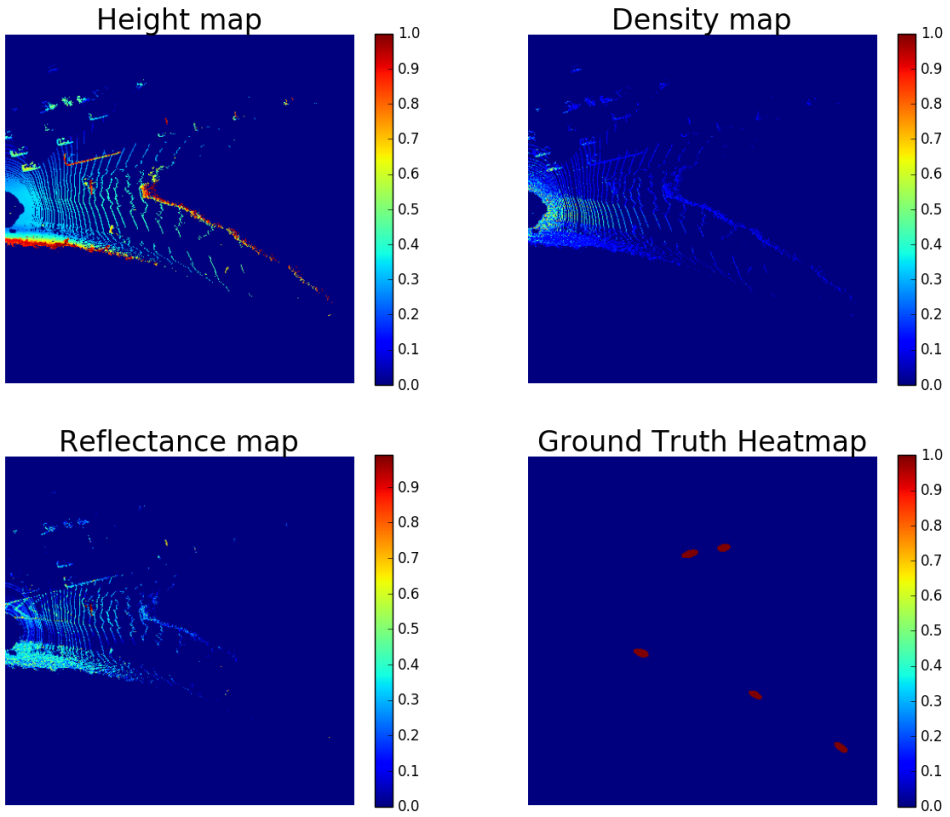


FIGURE 4.3 – Illustration des canaux d'entrée (hauteur maximale, densité et réflectance) ainsi que la vérité de terrain générée. Pour les données d'entrées, l'image correspond à l'application d'un opérateur `max` sur la dimension z.

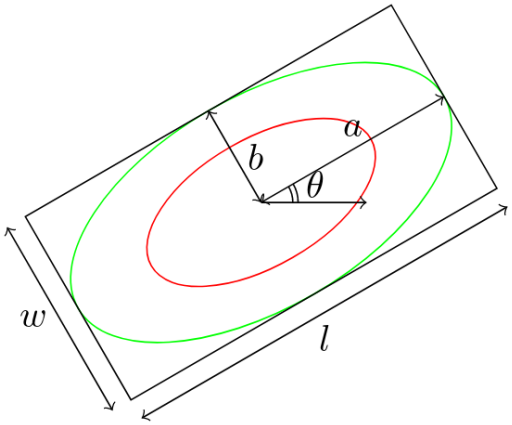


FIGURE 4.4 – Transformation de la boîte englobante orientée en ellipse. L'ellipse rouge est une version réduite de l'ellipse inscrite verte.

La vérité de terrain utilisée pour le réseau de neurones est une image constituée de taches elliptiques, chaque ellipse correspond à un obstacle comme le montre la figure 4.5.

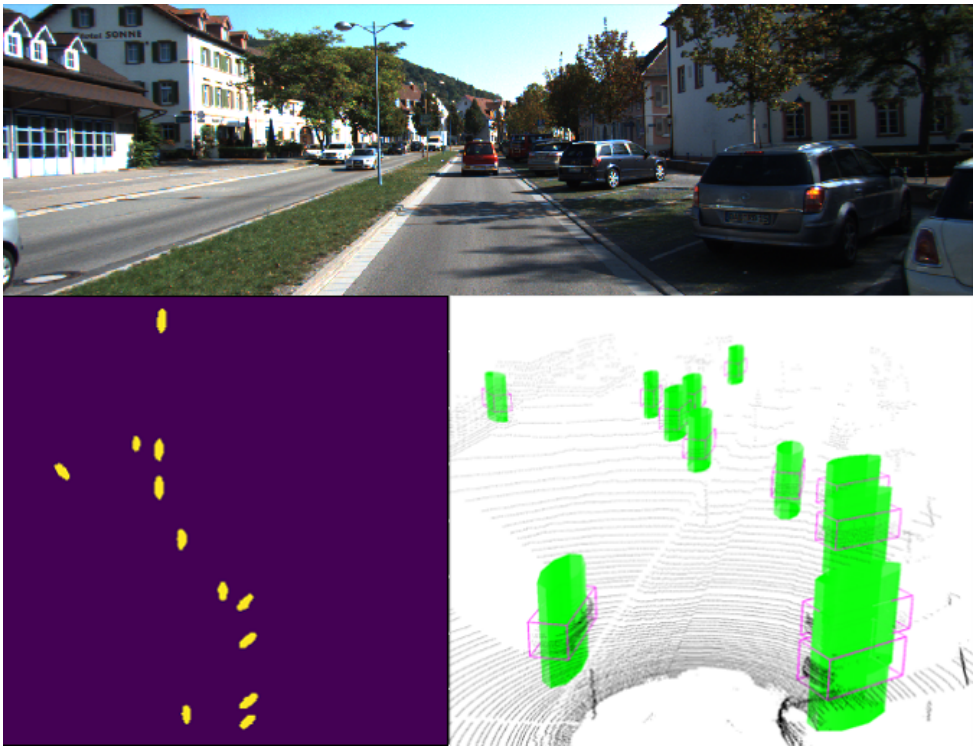


FIGURE 4.5 – Exemple de vérité de terrain générée à base d'ellipses. En haut, l'image RGB uniquement pour la visualisation de la scène. En bas à gauche, la vérité de terrain créée. En bas à droite, les annotations en 3D, les formes vertes représentent les ellipses calculées à partir des boîtes.

4.2.3 Architecture de segmentation

Le rôle du réseau de segmentation est de produire une carte de segmentation en BEV dont le score de chaque pixel décrit la probabilité du voxel correspondant d'être occupé par un véhicule. La carte de sortie donc est comparable à l'estimation d'une grille d'occupation, mais pour une catégorie d'objet spécifique. Si une cellule présente un score élevé, cela signifie que le réseau estime que cette cellule est occupée par une voiture. Dans le cas inverse, la cellule est soit occupée par un objet d'un autre type (une glissière de sécurité, un mur, etc.), soit vide. Une fonction sigmoïde est appliquée à la sortie du réseau afin de restreindre les valeurs des scores entre 0 et 1.

Il existe actuellement de nombreuses architectures dédiées à la segmentation sémantique, principalement sur images RGB. Nous pouvons notamment citer SegNet (BADRINARAYANAN, KENDALL et CIPOLLA, 2017), DeepLab et ses versions alternatives (CHEN et al., 2017a ; CHEN et al., 2018), PSPNet (ZHAO et al., 2017) ou bien HRNet (WANG et al., 2020). La puissance d'un algorithme de segmentation sur image RGB étant relative aux gradients de l'image originale, les principales avancées sont axées sur l'attention aux détails sur les pixels et à l'exploitation du contexte. Dans le cadre de nos expériences, contrairement aux images RGB denses, seule une faible partie des pixels de la BEV est occupée par des véhicules. De plus, l'objectif n'est pas de s'ajuster aux données, mais plutôt d'extrapoler l'occupation des cibles, seule une partie des véhicules étant visible (par exemple l'arrière d'une voiture, l'avant n'est pas visible du fait du fonctionnement du LiDAR). Nous souhaitons exploiter un réseau de segmentation rapide à mettre en place, mais également rapide en termes de temps d'exécution. Deux réseaux ont été étudiés : le U-Net (RONNEBERGER, FISCHER et BROX, 2015) (Figure 4.6) et le PSPNet (ZHAO et al., 2017) (Figure 4.7).

L'architecture U-Net (RONNEBERGER, FISCHER et BROX, 2015) est une architecture encodeur-

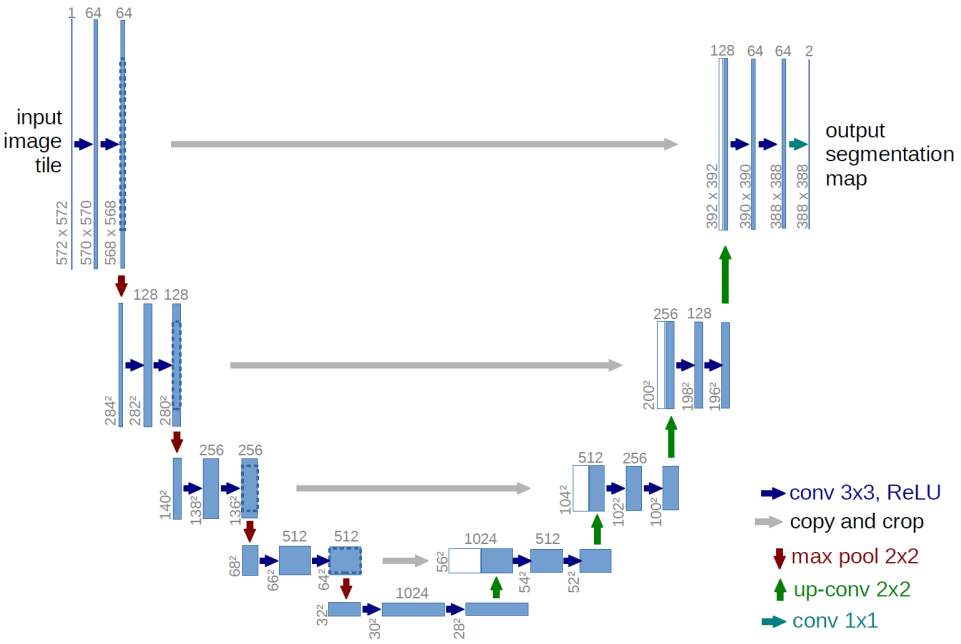


FIGURE 4.6 – Architecture du U-Net.

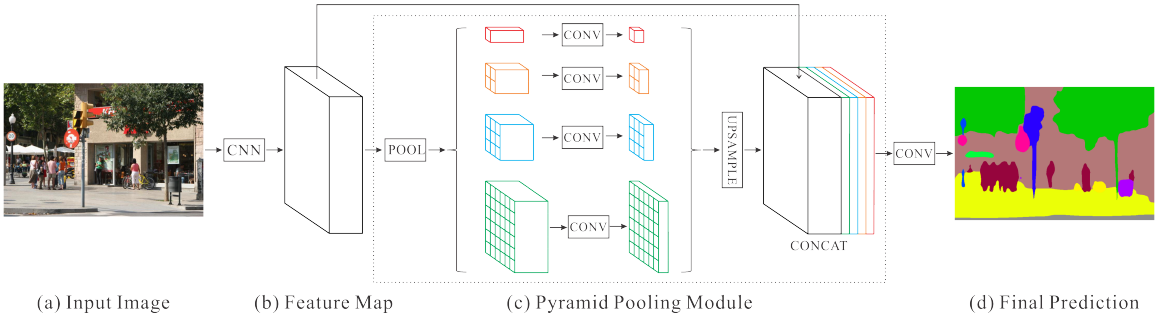


FIGURE 4.7 – Architecture du PSPNet.

décodeur exploitant des caractéristiques bas-niveau par l'intermédiaire de connexions entre les deux parties du réseau pour reconstruire une sortie plus fidèle aux détails présents dans l'image d'entrée. Simple à mettre en place et originellement employée pour la segmentation d'images médicales, cette architecture a montré des performances acceptables sur des tâches plus génériques de segmentation sémantique. L'architecture PSPNet (ZHAO et al., 2017) (*Pyramid Scene Parsing Network*) a été définie en partant du constat que la plupart des architectures antérieures se focalisaient principalement sur les aspects locaux sans prendre en compte les caractéristiques de la globalité de la scène. L'un des exemples donnés dans l'article concerne la confusion d'un FCN (LONG, SHELHAMER et DARRELL, 2015) entre un bateau et une voiture sur une image représentant un ponton sur un cours d'eau. La scène donne alors un a priori fort sur les classes pouvant exister dans la scène. Partant de ce constat, les auteurs ont développé des opérateurs de *pooling* multiéchelles pour extraire des caractéristiques selon plusieurs degrés de détails. Dans le cas de la conduite autonome, le fait que la scène soit une école, un parking ou bien une autoroute peut donner des informations sur la présence et la disposition de véhicules. Par conséquent, nous sommes partis de l'hypothèse selon laquelle certaines structures inhérentes à certains types d'environnements pourraient potentiellement améliorer la segmentation. Néanmoins, PSPNet étant plus complexe et plus lent, le réseau est principalement utilisé dans cette étude pour vérifier

si le réseau U-Net, plus léger, parvient à obtenir des performances similaires dans les conditions les plus basiques.

4.2.4 Extraction des détections

Le réseau de segmentation fournit une carte décrivant une estimation de l'état des voxels de la scène, mais ne fournit pas les obstacles individuellement. Il s'agit de la principale différence entre segmentation et détection. De ce fait, cette section détaille la façon dont les obstacles sont représentés dans la BEV, les techniques employées pour extraire les obstacles individuellement.

Comme énoncé précédemment, le réseau de segmentation délivre une carte contenant des cibles elliptiques représentant les véhicules. Les ellipses sont des coniques, donc sont exprimables sous la forme d'un modèle de 6 paramètres A,B,C,D,E,F avec $(A,B,C) \neq (0,0,0)$ et $4AC - B^2 > 0$. Un point (x,y) appartient à une conique Q si :

$$Q(x,y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{4.1}$$

Les ellipses étant des courbes fermées, il est possible de définir un intérieur $Q(x,y) < 0$ et un extérieur $Q(x,y) > 0$.

Les 6 paramètres de la conique sont estimés par les équations suivantes :

$$\begin{aligned}
A &= a^2\sin(\theta)^2 + b^2\cos(\theta)^2 \\
B &= 2(b^2 - a^2)\sin(\theta)\cos(\theta) \\
C &= a^2\cos(\theta)^2 + b^2\sin(\theta)^2 \\
D &= -2Ax_c - By_c \\
E &= -Bx_c - 2Cy_c \\
F &= Ax_c^2 + Bx_cy_c + Cy_c^2 - a^2b^2
\end{aligned} \tag{4.2}$$

Disposant d'un modèle paramétrique, il est donc possible de réaliser des optimisations afin de récupérer les caractéristiques sous-jacentes des ellipses de l'image.

Un exemple de processus simple d'extraction d'ellipses dans une image est décrit dans l'algorithme 1 et illustré figure 4.8. On définit T_{pics} un seuil définissant quels pixels peuvent être des « pics » et T_{valid} un seuil pour binariser la carte sortie par le réseau. Pour chaque pic, on extrait le *patch* correspondant de la carte binarisée. Les pixels du *patch* représentant la cible courante sont ensuite extraits par un algorithme de remplissage par diffusion (*Flood Filling*). Ici, pour chaque pixel parcouru, ses 8 voisins sont analysés pour la propagation. Cette approche est utilisée pour ne récupérer que les pixels connexes au pic courant. La taille du *patch* étant fixe, cela permet d'éviter l'inclusion de cibles voisines. Une fois les pixels récupérés sous la forme d'un masque binaire, le gradient du patch est calculé, les contours de la région sont définis comme étant les pixels dont la norme du gradient n'est pas nulle. Connaissant les correspondances entre les pixels de la carte et les voxels dans l'espace 3D, on peut donc sélectionner les voxels appartenant au contour de la cible Une ellipse est alors ajustée à partir des pixels du contour afin d'obtenir les caractéristiques de la cible. La méthode d'optimisation choisie pour la régression est l'algorithme présenté dans (Fitzgibbon, Pilu et Fisher, 1999). Cette technique est une méthode d'optimisation sous contrainte. Elle contraint la conique obtenue à être une ellipse (pas de parabole ou d'hyperbole).

Le processus décrit est appliqué sur les pixels de la carte, la précision des résultats dépendra fortement de la résolution choisie pour la carte de sortie.

Le rôle d'un réseau de segmentation étant de classifier chacun des pixels d'une image, la vérité de terrain doit définir quels pixels sont occupés par des voitures. En détection, les obstacles sont généralement représentés par des boîtes englobantes. Ici, les obstacles sont définis par une ellipse inscrite dans la boîte englobante (Figure 4.4). Cela permet de décrire l'objet par une forme

Data : Carte de probabilités MP, taille du *patch* s, seuil des valeurs sources T_{pics}, seuil de validation T_{valid}
Result : Liste des détections $dets$

$pics \leftarrow$ trouverPics (MP, T_{pics});
$H_{binaire} \leftarrow$ binariser (MP, T_{valid});
$dets \leftarrow$ [];

foreach p *in* $pics$ **do**
    $patch \leftarrow$ extrairePatchAutourPic ($H_{binaire}$, p);
    $patchC \leftarrow$ remplissage ($patch$, p);
    $patchCGradNorm \leftarrow$ norm (grad ($patchC$));
    $ellipse \leftarrow$ ajusterEllipse ($patchCGradNorm$);
    $paramtresBoite \leftarrow$ extraireInfoEllipse ($ellipse$);
    $dets$.insérer ($paramtresBoite$);
end

Algorithme 1 : Procédure utilisée pour extraire les ellipses d'une carte de probabilités.

paramétrique simple. Une hypothèse est posée quant à ces ellipses : il n'y a aucune intersection entre celles-ci. Le facteur de réduction appliqué à l'ellipse inscrite permet de renforcer les écarts entre les obstacles (exemple : zones de stationnement). La scène est alors illustrée comme un ensemble de cibles elliptiques avec différentes tailles et orientations.

4.3 Évaluation expérimentale

Cette section détaille les différents paramètres expérimentaux sélectionnés et les résultats correspondants.

4.3.1 Données d'entrée

Pour les données d'entrée, l'espace des recherches est restreint à $[-40.0, 40.0] \times [-1.0, 3.0] \times [0.0, 80.0]$ (les distances sont en m). Les dimensions de la grille sur le plan horizontal sont $(n_x, n_z) = (400, 400)$; chaque voxel correspond à un pavé de taille $(s_x \times s_z) = (0.2 \times 0.2)$ m^2.

4.3.2 Extraction d'ellipses

Le *patch* extrait autour de chaque pic fait 40 pixels de côté. Le seuil pour les pics T_{pics} est fixé à 0.95 tandis qu'un pixel est considéré comme valide (occupé par un véhicule) si sa valeur est supérieure à $T_{valid} = 0.1$. Pour accélérer le processus, une carte de visites a été créée pour éviter les traitements similaires : si un pic est détecté dans une région déjà visitée, il est ignoré.

4.3.3 Fonction de coût

La fonction de coût utilisée pour l'entraînement est une *focal loss* (Lin et al., 2017b). Il existe de nombreuses fonctions de coût pour la segmentation dans la littérature. La *focal loss* est une fonction de coût principalement conçue pour mieux s'adapter aux déséquilibres de classe, c'est-à-dire la prédominance d'une ou de plusieurs classes cibles sur les autres. Dans le cas présent, la grande majorité des pixels de la carte de segmentation appartient au *background* (espaces vides, infrastructures, autres usagers, etc.). La *Focal Loss* est définie pour chaque pixel comme :

$$FL(p_t) = -\alpha_t (1 - p_t)^{\beta} \log(p_t) \tag{4.3}$$

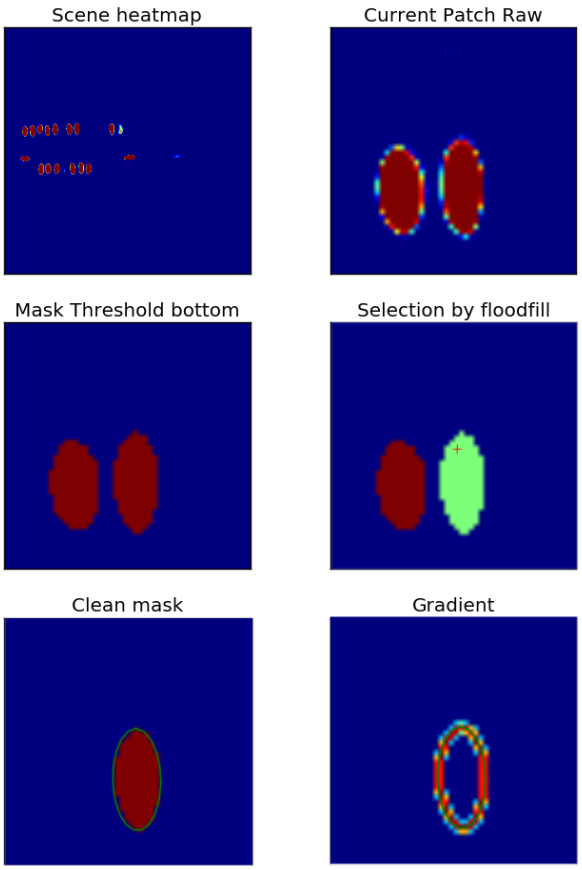


FIGURE 4.8 – Illustration du processus d'extraction des prédictions. La plus grande valeur sert de point de départ. Une diffusion est opérée afin de ne sélectionner que les pixels relatifs à la cible courante. Le gradient du *patch* est ensuite calculé. Les pixels du gradient dépassant un certain seuil sont utilisés comme entrées de l'algorithme d'optimisation.

avec p_t la probabilité estimée par le modèle, α_t est un terme de modulation lui-même dépendant d'un terme α, β et α sont les paramètres de la fonction. (α, β) sont fixés à $(0.25, 2)$. La valeur pour une image consiste en la moyenne de cette fonction appliquée à tous les pixels. Le principal a priori de cette fonction est de considérer que plus une catégorie de cibles a d'exemples, plus le système à entraîner sera performant sur cette classe. Cette fonction modifie la dynamique de la fonction d'entropie croisée afin de donner plus de poids aux pixels dont l'erreur est grand. À l'inverse, si l'écart entre la vérité de terrain et la prédiction est faible, la sortie de la fonction sera plus négligeable.

4.3.4 Paramètres de l'entraînement

Comme énoncé précédemment, nous nous sommes focalisés sur le U-Net et le PSPNet. L'entraînement a été réalisé sur un ensemble de 3741 échantillons parmi les 7481 nuages de la base KITTI, le reste étant destiné à la validation.

Le réseau complet est entraîné à l'aide d'un optimiseur *Adam* (KINGMA et BA, 2014) sur 150 époques, avec un taux d'apprentissage de base de 0.01 et un facteur de décroissance de 0.8 appliqué toutes les 15 époques. La décroissance du taux d'apprentissage est une pratique commune. Au fil de l'apprentissage, le réseau s'approche de plus en plus du minimum global de la fonction de coût. Un taux d'apprentissage élevé au départ permet au réseau de bien se positionner dans l'espace de ses paramètres, mais peut causer une convergence vers une solution

sous optimale, voire créer un comportement divergent. Les paramètres ont été sélectionnés suite à une dizaine d'entraînements.

4.3.5 Résultats et discussion

Le tableau 4.1 montre les résultats obtenus par notre méthode et des approches concurrentes sur des jeux de données de *validation*.

TABLE 4.1 – Performances pour la détection en BEV des véhicules sur les données de validation. Valeurs en *Average Precision* (AP) (en %).

Méthode	*Easy*	*Moderate*	*Hard*
MV3D(CHEN et al., 2017b)	86.55	78.10	76.67
Frustum Pointnet (QI et al., 2018)	88.16	84.02	76.44
VoxelNet (ZHOU et TUZEL, 2018)	89.60	84.81	78.57
SECOND (YAN, MAO et LI, 2018)	**89.96**	**87.07**	79.66
U-Net $n_z = 1$	79.49	76.69	73.94
PSPNet ResNet34 $n_z = 1$	79.67	76.67	73.92
U-Net $n_z = 5$	84.45	83.30	**82.01**

Le premier constat concerne la comparaison entre U-Net et PSPNet. Pour les deux réseaux, nous avons sélectionné $n_z = 1$. Il s'agit des conditions extrêmes où chaque voxel est encodé sur toute la dimension verticale. Les détails dans la forme des objets ne sont donc pas exprimés par la structure de la grille de voxels. On constate que, malgré le fait que PSPNet soit plus complexe dans son fonctionnement, les deux réseaux délivrent des performances assez similaires. Les deux réseaux sont entraînés suivant les mêmes conditions expérimentales. À partir des mêmes données, les deux réseaux ont réussi à extraire la même quantité d'information sur les véhicules.

Concernant le UNet avec $n_z = 5$, plusieurs points sont notables. D'abord, les scores obtenus sont plus importants que le U-Net $n_z = 1$. Les données étant plus « expressives » sur l'axe vertical, les cibles sont bien plus facilement identifiables. On constate également que la décroissance des performances avec la difficulté est très faible.

La segmentation demandée au réseau est non conventionnelle. Dans une segmentation classique, les objets sont bien délimités par un contour. Dans le cas présenté ici, seule une partie du véhicule est représentée, le réseau doit donc extrapoler l'espace occupé par le véhicule. En observant la carte de sortie, on constate une sorte de « zone négative » autour de chaque cible dans laquelle les valeurs sont bien plus basses que le reste des zones associées au *background* comme le montre la figure 4.9.

Le fait est que cette zone n'est pas définie explicitement dans la vérité de terrain, seuls les pixels associés aux véhicules sont indiqués. Cela signifie que, lors de l'apprentissage, le réseau a mis l'accent sur l'apprentissage des frontières. Cela peut s'expliquer par l'utilisation de la *focal loss*. En effet, les images comptent un nombre bien plus important de pixels associés au *background* que de pixels « positifs ». Lors de l'entraînement, le réseau tend à annuler rapidement la grande majorité de la carte afin de faire baisser rapidement les valeurs de la fonction de coût. Mais au bout d'un certain nombre d'époques, lorsque les pixels *background* commencent à tous avoir des valeurs suffisamment faibles, la pondération de la *focal loss* sur les grandes erreurs va forcer l'optimisation à se focaliser sur les cibles et réduire l'importance des autres pixels *background*. Les pixels périphériques aux cibles sont donc minimisés plus brutalement afin d'obtenir une distinction nette entre les pixels cibles et les pixels *background*. Ce phénomène est important

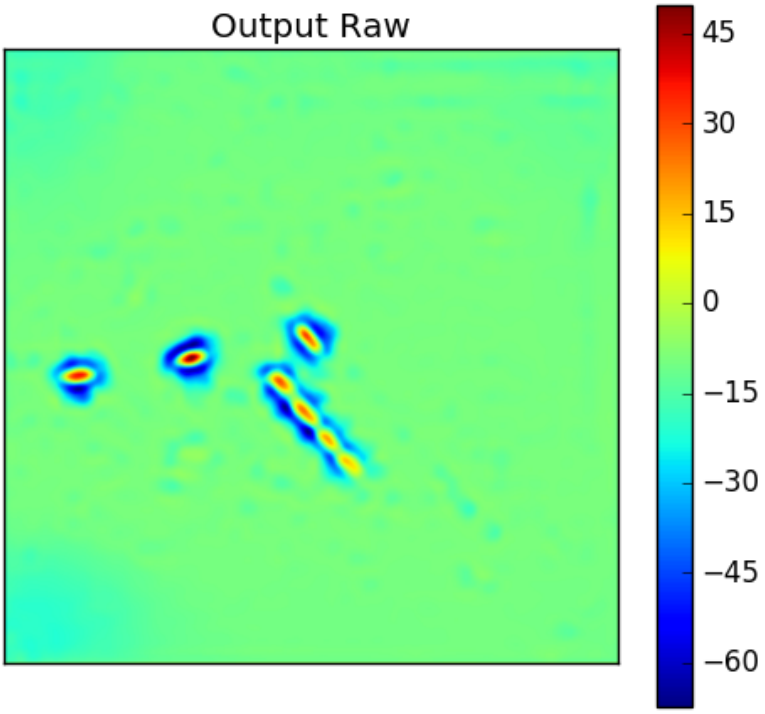


FIGURE 4.9 – Illustration de la « zone négative » (en bleu) sur la sortie du réseau, avant le sigmoïde.

notamment dans le cas des véhicules en stationnement. En effet, les cibles plus distantes sont camouflées par celles du premier plan. Comme il n'y a pas d'intersection entre les obstacles, ces zones négatives permettent de mieux positionner les objets entre eux.

La figure 4.10 illustre une comparaison entre prédictions (carte et boîtes prédites) et les vérités de terrain. Il y a des dépendances importantes des boîtes extraites, d'une part, à la résolution de la grille de sortie, et d'autre part, à la qualité de la segmentation. La résolution de la grille impacte en effet le nombre de voxels associé à chaque véhicule, mais également le nombre de pixels de contours utilisés lors de l'optimisation de Fitzgibbon. Une légère erreur sur la carte de sortie introduit un bruit sur les entrées de l'optimisation et donc sur l'ellipse et la boîte de sortie.

On peut observer sur la figure deux fausses détections, une qui correspond à l'ellipse rouge et une qui correspond à l'ellipse bleue. L'ellipse rouge est un faux négatif (une cible qui a été manquée) représentatif de l'une des particularités de la base KITTI, à savoir les annotations qui sont définies dans le frustum de la caméra avant. Cependant, la définition de la frontière du frustum et les conditions d'existence d'annotations à proximité de ces frontières n'est pas explicitées, de ce fait, il existe des cas pour lesquels les annotations existent alors que l'obstacle est très difficilement perceptible sur l'image RGB, et inversement, des cas pour lesquels l'annotation n'est pas définie alors qu'une grande partie de la cible est visible. Dans le cas présent, le réseau a estimé que la voiture était en dehors du frustum et donc ne l'a pas représentée sur la carte de segmentation. La seconde ellipse (en bleu) est un faux positif, une détection qui n'existe pas dans les annotations de la base. Bien qu'une voiture soit physiquement présente à cette position, le fait qu'elle n'apparaisse pas dans les annotations cause une erreur.

En termes de temps de traitement, le réseau est exécuté en 30 ms tandis que l'extraction d'ellipse sur un cas avec une dizaine de cibles prend 3 ms sur un CPU Intel Core i7−7700K 4.2 GHz, ce qui permet de rester sous la barre des 10 Hz du LiDAR.

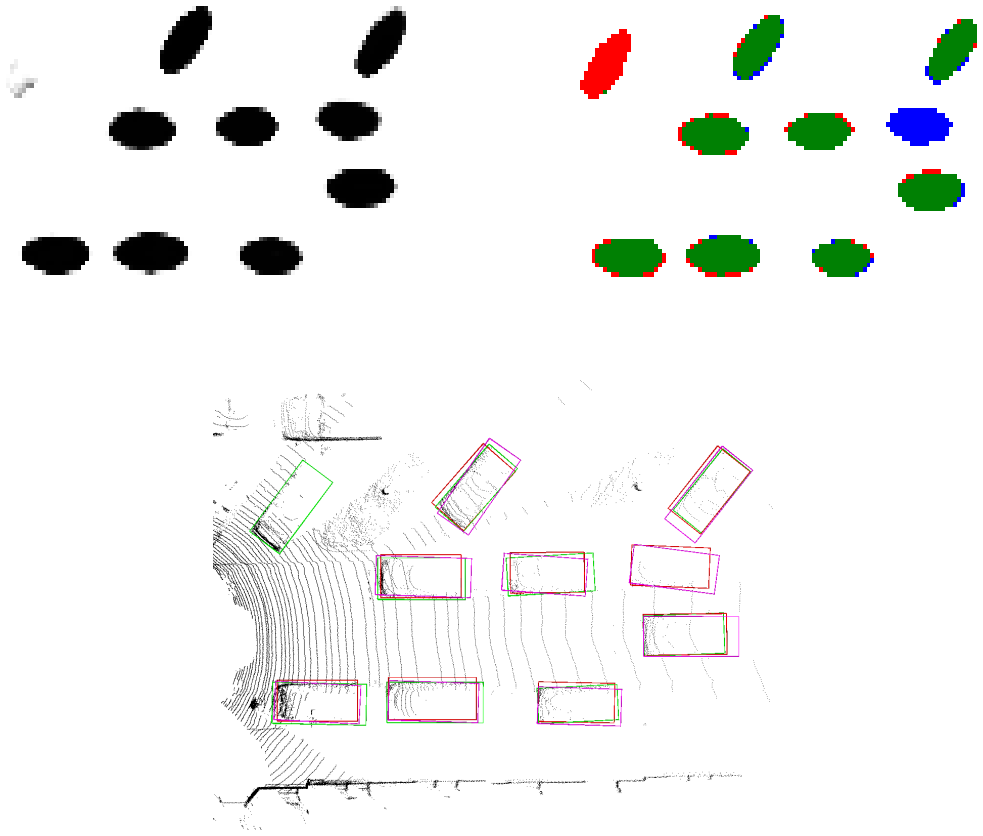


FIGURE 4.10 – Résultats quantitatifs sur les données de validation. (a) La carte de probabilités produite par le réseau. (b) Différence entre la vérité de terrain et la carte estimée binarisée (seuil à 0.5). Le vert signifie vrai positif, le rouge faux négatif et le bleu faux positif. (c) Détections sous formes de boîtes. Les vertes représentent la vérité de terrain, les rouges les boîtes issues de l'extraction d'ellipse, les magenta les boîtes issues d'un VoxelNet.

4.4 Conclusion et analyse critique

La méthode présentée dans cette section repose sur deux phases, une partie réseau de neurones et une partie post-traitement. Le réseau délivre une pseudo-carte d'occupation dont chaque pixel indique la probabilité estimée que le voxel correspondant soit occupé par une cible. La représentation choisie permet d'identifier les cibles à des ellipses pleines. La carte ne délivrant que l'état estimé de la scène, un post-traitement d'extraction des cibles est appliqué à la sortie du réseau afin d'extraire les boîtes englobantes. La fréquence du processus de traitement est supérieure à 10 Hz, évitant ainsi la perte de données capteurs à cause de retards. De plus, les hyperparamètres nécessaires concernent principalement la discrétisation de la grille d'études. Concernant la sortie du réseau, un système fondé sur les ancres en nécessite au minimum cinq (longueur/largeur, orientations des ancres, seuil de superposition pour considérer une ancre comme positive/négative) tandis que notre approche repose sur deux seuils, l'un lié aux scores des pixels de la carte de sortie et l'autre à la taille du patch. En outre, ces seuils n'impactent pas l'apprentissage, mais uniquement les prédictions.

Cependant, la méthode présentée fonctionne bien sur les véhicules, car ils sont facilement identifiables sur des scènes en vue de dessus. En fonction de leur placement par rapport à l'ego-véhicule, les formes en « L » sont visibles à même la carte d'entrée générée. Toutefois, du fait de la représentation simple des données et de la discrétisation choisie, le réseau n'est pas

adapté pour la détection des piétons et cyclistes, ces derniers n'occupent pas autant d'espace sur le plan (Oxy) que des véhicules. D'une part, les piétons ne seraient identifiables que par un faible nombre de pixels sur la carte, d'autre part, ils seraient trop facilement assimilables à d'autres objets verticaux tels que des réverbères ou des troncs d'arbre. En outre, sur les objets lointains, les pixels ne forment pas toujours des ellipses aux contours bien définis du fait du manque d'information. L'extraction des paramètres des cibles reposant sur un seuillage, cela peut conduire à des erreurs importantes dans l'estimation des dimensions.

Outre ces défauts, le système constitue déjà une première approche performante et sert de preuve de concept sur l'utilisation de l'occupation dans un système de détection. Cette direction est explorée dans les chapitres suivants.

Chapitre 5

Détection d'obstacles résistant aux changements de résolution dans les nuages de points

Sommaire

5.1 Introduction

L'un des aspects cruciaux liés aux réseaux de neurones concerne la généricité vis-à-vis des données. Un système de reconnaissance doit pouvoir s'adapter et délivrer des prédictions fiables sur des données qui n'ont jamais été vues par le réseau lors de l'apprentissage. Les domaines du transfert d'apprentissage et de l'adaptation de domaine sont devenus des sujets souvent traités en ce qui concerne les données issues de caméras RGB. En effet, les caméras sont omniprésentes, rapidement déployables et facilement accessibles, rendant les données provenant de sources multiples très accessibles. Toutefois, cela suppose également de connaître les caractéristiques du nouvel ensemble de données, réduisant ainsi les possibilités d'usage dans des situations totalement inconnues. Les capteurs 3D LiDAR sont quant à eux encore très coûteux et bien moins accessibles. Qui plus est, l'annotation de données 3D reste complexe. Ainsi, les bases de données publiques contenant des données LiDAR sur route annotées sont plus rares et souvent centrées sur des environnements ou des villes précis.

Deux types de variations existent principalement sur les données automobiles :

- la variation de capteurs. Dans ce cas, les données varient par leur mode d'acquisition et par leur format. Concernant les capteurs LiDAR, cela peut concerner la manière dont sont acquis les points 3D (le type de balayage) ou plus fréquemment la résolution du capteur et donc le nombre de points acquis. Sur des données caméra, ces variations peuvent provenir du bruit d'acquisition, de la colorimétrie ou bien des déformations géométriques induites par les optiques du capteur ;
- la diversité des scènes capturées. Certaines bases se focalisent sur un nombre limité de lieux pour les acquisitions (par exemple, quelques villes principales). Les acquisitions peuvent également être contraintes par les conditions météorologiques et lumineuses (pluie, brouillard, nuit, etc.).

Si les acquisitions en elles-mêmes ne sont pas les plus coûteuses, l'annotation et les ressources nécessaires limitent grandement le nombre de données utilisables pour les entraînements.

Dans une optique de performances et pour assurer une meilleure sécurité, il est envisageable de ne se focaliser que sur un seul capteur. Cette approche peut néanmoins engendrer un sur-apprentissage sur une configuration de capteurs. D'autres configurations risquent ainsi de ne pas être exploitables par le réseau entraîné. Or, il peut être utile de ré-exploiter sur de nouveaux projets des résultats obtenus dans un cadre antérieur. Cela est d'autant plus vrai dans le cadre des capteurs LiDAR. Ces derniers restent encore onéreux donc peu accessibles. De plus, même si les performances sont moindres, un détecteur versatile peut tout de même réduire les efforts nécessaires pour annoter de nouvelles données

Dans ce chapitre, nous proposons des techniques d'entraînement permettant d'accroître les capacités de détection de réseaux de neurones sur plusieurs types de distributions de nuages de points 3D à partir d'une seule base (et d'un capteur LiDAR), comme le montre la figure 5.1.

Plusieurs contributions ont été réalisées :

- nous introduisons une réduction du nombre de couches dans le nuage d'entrée lors de l'entraînement. Cette méthode présente deux avantages. Le premier est relatif à l'augmentation de données, dans la mesure où le réseau observe une plus grande variété de données durant l'entraînement. Le second aspect concerne la difficulté de détection. Dans un nuage de points très épars, l'identification d'objets avec un nombre limité de points est délicate. De ce fait, l'image, dont la résolution ne varie pas, est mise davantage à contribution aussi bien pour la classification que pour l'estimation des paramètres des boîtes ;
- nous introduisons également une nouvelle représentation des objets dans la scène. Dans la grande majorité des cas, un obstacle est défini comme une boîte englobante orientée. Dans le chapitre 4, chaque objet était défini comme une ellipse. Ici, nous choisissons une représentation probabiliste, ici des lois normales, permettant ainsi l'usage de distances probabilistes en tant que fonctions de coût.

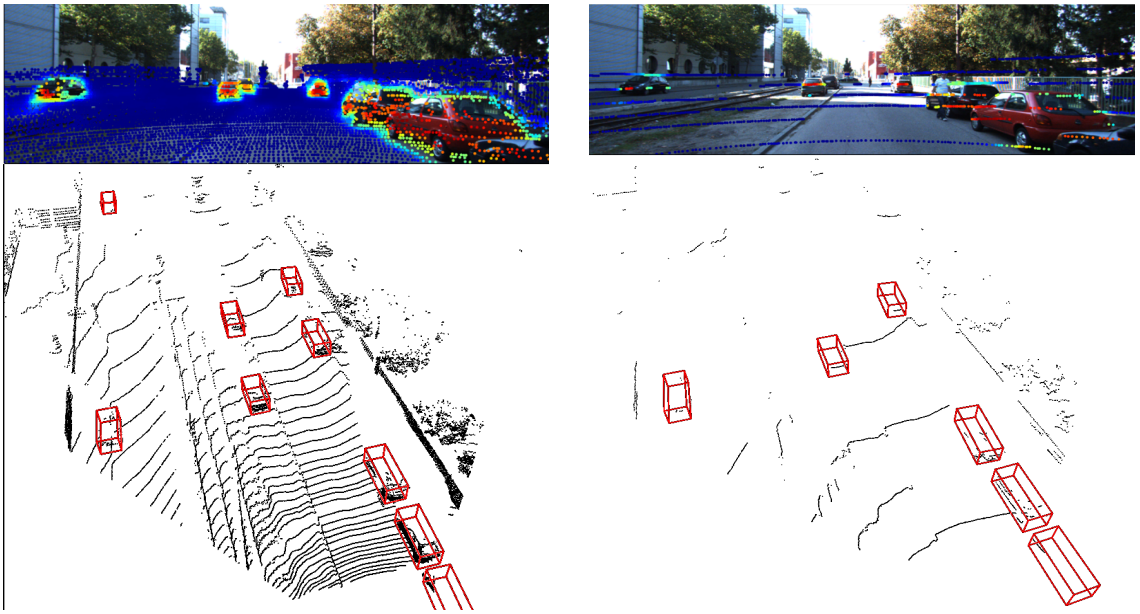


FIGURE 5.1 – Capacité du réseau entraîné à traiter des nuages de points de résolutions différentes. En haut : surimpression de la segmentation des voxels sur l'image RGB (bleu : 0.0, rouge : 1.0). En bas : nuage de points utilisé et détections résultantes. Les détections manquantes sur l'image de droite proviennent de l'absence de points 3D.

Ces techniques sont appliquées sur trois réseaux, le premier n'utilisant que les points 3D, les deux autres exploitant à la fois le nuage de points 3D et la caméra RGB. Ces deux architectures ne sont constituées que d'opérateurs standard, permettant ainsi l'export vers d'autres *frameworks* d'apprentissage profond grâce à des outils tels que ONNX (*Open Neural Network Exchange*).

Nous nous intéressons également dans ce chapitre à la détection BEV des voitures. Néanmoins, la méthode est extensible à d'autres classes d'objets. Le cas mono-classe est traité dans ce chapitre mais les réseaux peuvent également être entraînés pour gérer plusieurs classes simultanément en modifiant le nombre de canaux sur la sortie (un canal par classe sur la classification). En fin de chapitre, nous expérimentons les approches proposées sur des données non annotées. Ces données au format *rosbag* proviennent des bases UTBM Robocar et Ford AV Datasets. Chacune de ces bases utilise un LiDAR 32 nappes.

5.2 État de l'art sur l'estimation des orientations en détection BEV et 3D

Les paramètres métriques (position et dimensions) sont globalement estimés à l'aide des techniques similaires dans les différentes méthodes de la littérature. Néanmoins, dans le cas de l'orientation, il existe plusieurs propositions distinctes. Dans l'intégralité des méthodes étudiées, les variables métriques sont exprimées en mètres, tandis que les angles sont exprimés en radians.

Comme détaillé dans le chapitre 2, l'orientation est, dans la plupart des cas, représentée par un scalaire θ pour le cap de la cible. D'autres bases représentent les trois axes de rotations, soit par l'intermédiaire d'angles d'Euler, soit par l'intermédiaire de quaternions. Dans notre cas, nous nous focalisons sur la représentation scalaire. En effet, la majorité des routes traitées disposent d'une faible pente, l'assiette est le plus souvent négligeable.

Notons $\theta_{gt}, \theta_{pred} \in \mathbb{R}$ respectivement les angles visés et prédits.

Pour la fonction de coût, l'approche la plus directe consiste à effectuer une régression directe

de l'angle θ à l'aide d'une fonction de coût reposant sur une distance Lp, $p \in \mathbb{N}$ ou p l'infini. Les fonctions de coût les plus courantes étant :

- la fonction de coût $L1$, reposant sur une distance L1 (distance de Manhattan) : $L(\theta_{gt}, \theta_{pred}) = |\theta_{gt} - \theta_{pred}|$;
- la fonction de coût MSE (*Mean Square Error*), reposant sur une distance $L2$ (distance euclidienne) au carré : $L(\theta_{gt}, \theta_{pred}) = (\theta_{gt} - \theta_{pred})^2$;
- la fonction dite *Smooth L1*, un mélange des deux précédentes en fonction de la valeur de la différence : $L(\theta_{gt}, \theta_{pred}) = SmoothL1(\theta_{gt} - \theta_{pred})$

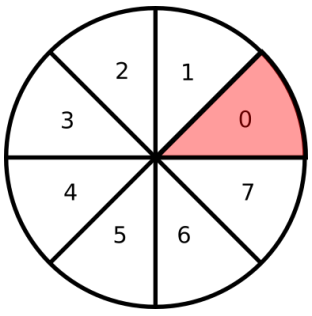
avec

$$SmoothL1(x) = \begin{cases} 0.5x^2 & \text{si } |x| < 1 \\ |x| - 0.5 & \text{sinon.} \end{cases} \tag{5.1}$$

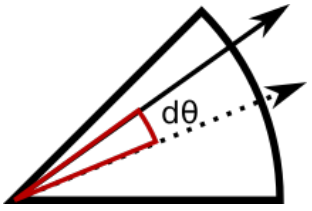
L'inconvénient majeur de cette méthode concerne les discontinuités. Ainsi, si nous représentons $\theta \in [-\pi, \pi]$, $\theta_{gt}, \theta_{pred}$ proches respectivement de $-\pi$ et π sont des angles très similaires, mais dont la valeur de la fonction de coût sera très importante du fait de la discontinuité. Une alternative consiste à appliquer la fonction de coût non pas à l'angle θ mais à $(\cos\theta, \sin\theta)$. Les discontinuités sont ainsi supprimées.

Les auteurs de SECOND (Yan, Mao et Li, 2018) proposent de décomposer la régression angulaire. Deux fonctions objectif sont introduites. La première est une fonction de coût fondée sur l'erreur du sinus : $L(\theta_{gt}, \theta_{pred}) = SmoothL1(\sin(\theta_{gt} - \theta_{pred}))$. Cette fonction permet d'obtenir la direction (la droite représentant l'axe de la cible) mais pas son sens. Une seconde fonction de classification est alors chargée de déterminer si la cible est orientée vers l'avant ou vers l'arrière. Une *Focal loss* est utilisée pour la classification.

D'autres approches telles que celle présentée dans (Mousavian et al., 2017) tentent de ramener le problème d'estimation de l'orientation à un problème de classification. La première étape consiste à partitionner le cercle trigonométrique en sections puis à déterminer dans quelle section se situe l'orientation de la cible comme illustré dans la figure 5.2a. Cependant, la classification ne permettrait de renvoyer qu'un nombre fini d'orientations. Pour combler ce manque, une seconde étape consistant en une estimation entre l'angle réel et l'angle correspondant au centre de la section est ajoutée (Figure 5.2b). L'un des points critiques de cette méthode concerne la résolution de la discrétisation qui devient un hyperparamètre supplémentaire pouvant influencer l'entraînement.



(a) Le cercle trigonométrique découpé en n sections, ici $n = 8$



(b) Régression de la différence entre le demi angle de la section (en pointillés) et l'angle visé

FIGURE 5.2 – Illustration du *MultiBin Orientation Regression*.

Dans les propositions précédentes, chaque paramètre de boîte englobante est traité indépendamment par une fonction de coût telle que la fonction *Smooth L1*, la fonction de coût finale étant la somme des fonctions de coût associées à chaque paramétré. Une approche employée

notamment dans la méthode MonoDIS (Simonelli et al., 2019) consiste non pas à traiter indépendamment chaque paramètre, mais à générer à l'aide d'une fonction une représentation alternative de la boîte, ici les sommets de la boîte définie par les paramètres. Si nous notons $\boldsymbol{\phi}$ les paramètres de la boîte prédite par le réseau, la fonction $\mathscr{F}$ a pour objectif de générer les sommets 3D B de la boîte correspondante : $B = \mathscr{F}(\boldsymbol{\phi})$. La fonction de coût consiste alors en une minimisation des distances entre sommets prédits B et sommets cibles $\hat{B}$ comme l'illustre la figure 5.3.

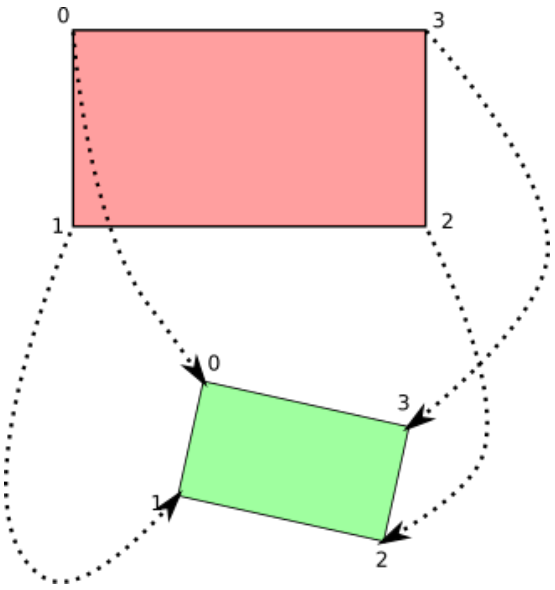


Figure 5.3 – Régression de sommets. La boîte verte représente $\hat{B}$ tandis que la rouge indique B.

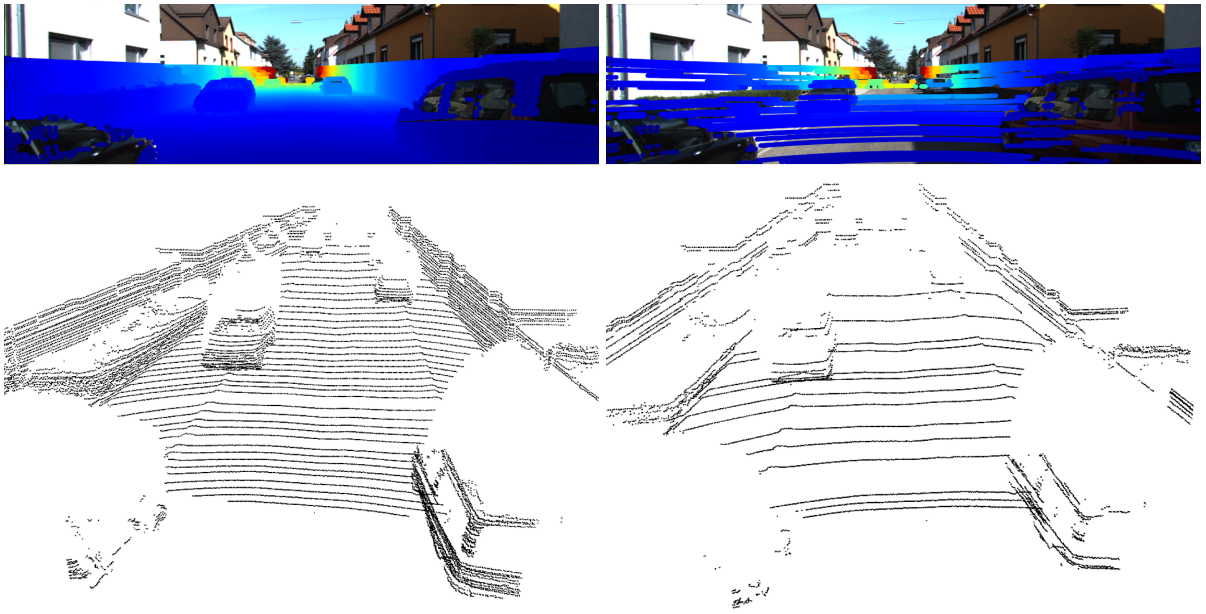
Cette représentation unifiée des paramètres a notamment inspiré une des contributions réalisées en matière d'entraînement des réseaux.

5.3 Techniques proposées

5.3.1 Réduction du nombre de nappes

La grande majorité des LiDAR utilisés sont des LiDAR mécaniques. Il est ainsi possible de modéliser un repère sphérique dont l'origine est situé proche du point d'émission du LIDAR. On peut ainsi définir une nappe/couche comme un ensemble de points 3D partageant la même latitude dans ce repère sphérique.
Une même scène peut revêtir plusieurs apparences suivant les caractéristiques du LiDAR utilisé (nombre et répartitions des nappes, résolution horizontale, champ de vision...) mais aussi de la pose du capteur sur le véhicule.
L'augmentation de données est un processus couramment appliqué permettant d'augmenter artificiellement la variabilité des données vues par le réseau au cours de l'entraînement. L'une des techniques couramment appliquées consiste à introduire un bruit sur la position (par exemple sur l'axe z) et/ou l'orientation du nuage de points. L'apparence globale du nuage n'est cependant pas modifiée. En plus d'appliquer les techniques communes, nous proposons une méthode pour générer des distributions de nuages de points. Pour chaque échantillon de la base de données, certaines couches du nuage de points sont supprimées comme le montre la figure 5.4. Pendant l'entraînement, nous conservons entre 25% et 60% de nappes dans chaque nuage.

Du côté de l'image, afin de conserver une uniformité dans les dimensions, nous extrayons de chaque image un *patch* de dimensions 1024×256 pixels. Afin que la cohérence des données soit



(a) Nuage de points initial (b) Nuage de points après retrait de nappes

FIGURE 5.4 – Illustration du retrait aléatoire de nappes. Les projections des points du nuage sont affichées sur les images RGB. Les couleurs représentent la distance au capteur LiDAR.

conservée, tous les points dont les projections image sont situés à l'extérieur de la région extraite sont retirés.

Le réseau RGB étant complètement convolutif, il est toutefois possible d'utiliser en inférence n'importe quelle taille d'image.

5.3.2 Représentation gaussienne des obstacles

Dans le cadre de la conduite autonome, les obstacles 3D sont représentés par une position $[x,y,z]^T$ (exprimés en m), des dimensions $[h,w,l]^T$ (exprimés en m) et une orientation θ suivant l'axe vertical (exprimés en rad). À partir des paramètres (x,y,w,l,θ), qui sont les paramètres recherchés dans le cadre de détection BEV, nous définissons pour chaque obstacle une loi normale $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ telle que :

$$\boldsymbol{\mu} = [x,y]^T \in \mathbb{R}^2,\ \boldsymbol{\Sigma} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \in \mathbb{R}^{2\times 2} \tag{5.2}$$

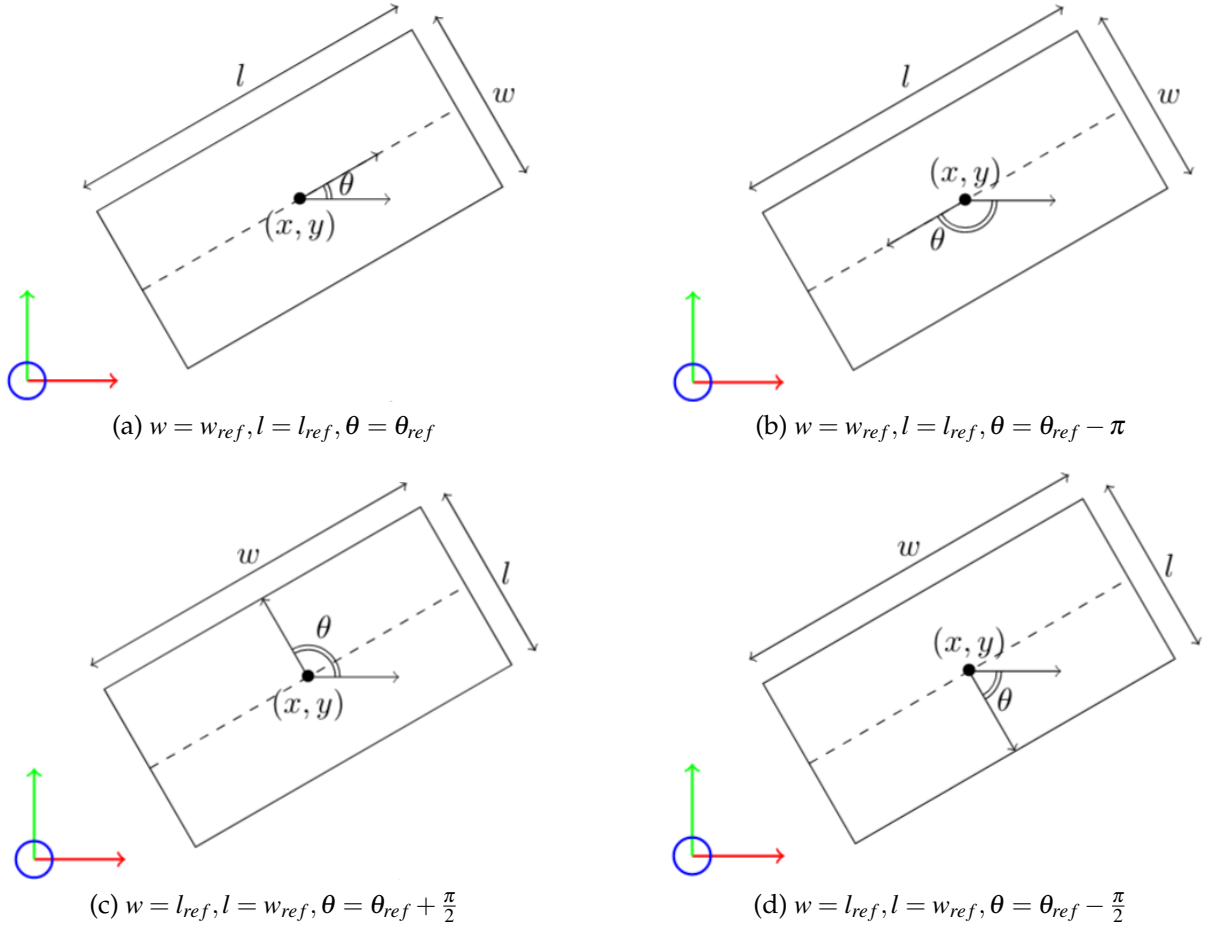
avec

$$a = \frac{\cos^2(\theta)}{2\sigma_w^2} + \frac{\sin^2(\theta)}{2\sigma_l^2},\ b = -\frac{\sin(2\theta)}{4\sigma_w^2} + \frac{\sin(2\theta)}{4\sigma_l^2},$$
$$c = \frac{\sin^2(\theta)}{2\sigma_w^2} + \frac{\cos^2(\theta)}{2\sigma_l^2},\ \sigma_w = \frac{w}{3},\ \sigma_l = \frac{l}{3}$$

,$\boldsymbol{\Sigma}$ étant définie positive.

Nous notons

$$F : \mathbb{R}^7 \to \mathbb{R}^2 \times \mathbb{R}^{2\times 2}$$
$$(x,y,z,h,w,l,\theta) \mapsto \boldsymbol{\mu},\boldsymbol{\Sigma}$$



(a) $w = w_{ref}, l = l_{ref}, \theta = \theta_{ref}$

(b) $w = w_{ref}, l = l_{ref}, \theta = \theta_{ref} - \pi$

(c) $w = l_{ref}, l = w_{ref}, \theta = \theta_{ref} + \frac{\pi}{2}$

(d) $w = l_{ref}, l = w_{ref}, \theta = \theta_{ref} - \frac{\pi}{2}$

FIGURE 5.5 – Ensembles de paramètres produisant la même fonction gaussienne. $w_{ref}, l_{ref}, \theta_{ref}$ sont les valeurs de référence.

la fonction transformant les paramètres $S = [x, y, z, h, w, l, \theta]^T$ en une loi normale bivariée.

Avec cette expression, chaque proposition peut être définie comme une fonction gaussienne. Cette représentation se focalise sur la forme de la gaussienne et, par conséquent, sur l'espace occupé par l'obstacle correspondant. Toutefois, elle ne fournit aucune information concernant l'orientation de l'obstacle. Ainsi, des ambiguïtés peuvent survenir du fait que plusieurs configurations de S peuvent produire la même gaussienne comme le montre la figure 5.5.

Ce phénomène est la raison de l'existence de la 3ème carte, la classification de l'orientation, qui permet de sélectionner de manière explicite l'orientation et ainsi d'opérer les modifications nécessaires sur les paramètres S. Pour la classification de voxels, les classes sont attribuées en fonction des annotations dans lesquels ils sont inclus.

5.4 Présentation des réseaux étudiés

Dans ce chapitre, nous décrivons 3 réseaux qui seront analysés au cours de cette étude. Un réseau ne traite que des nuages de points tandis que les deux autres utilisent également l'image RGB. La structure de ces trois réseaux est brièvement décrite sur la figure 5.6.

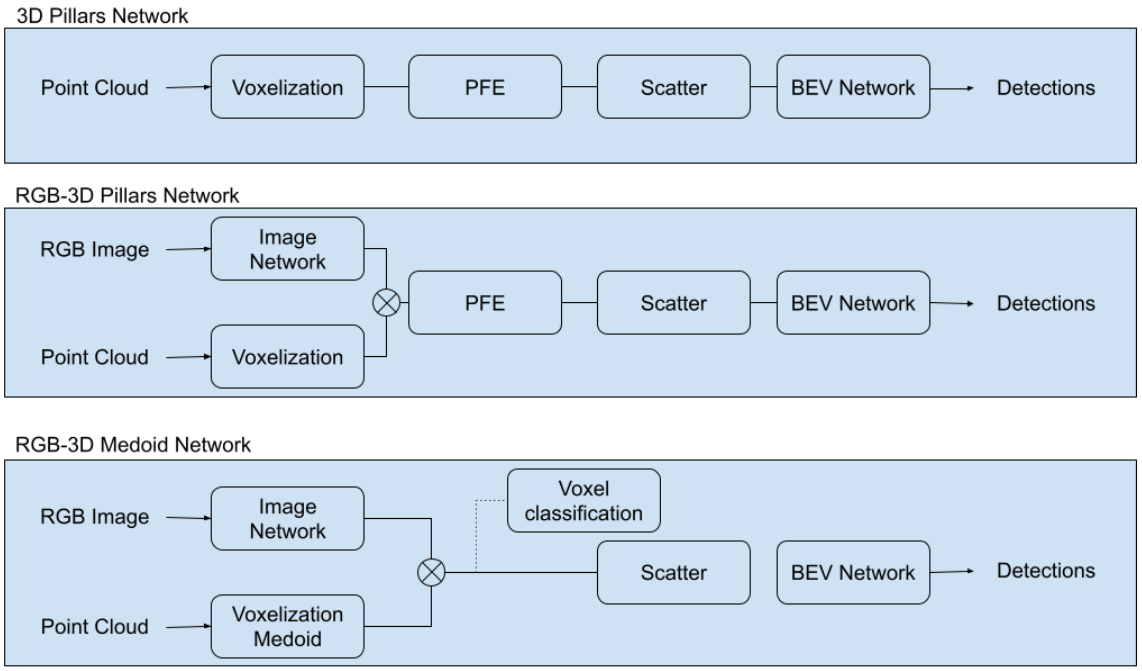


FIGURE 5.6 – Structure des trois réseaux étudiés.

On considère dans toute cette étude que l'axe x est orienté vers l'avant de l'ego-véhicule, l'axe y est vers la gauche et l'axe z vers le haut. Nous restreignons l'espace de recherche sur l'intervalle $[x_{min}, x_{max}]$ sur l'axe x et $[y_{min}, y_{max}]$ sur l'axe y. Les deux réseaux utilisant les images RGB sont des réseaux dits de fusion précoce (*early fusion*), c'est-à-dire que les caractéristiques images sont fusionnées aux informations 3D du nuage de point au sein du réseau. Le sous-réseau image fait partie intégrante du réseau de détection 3D. En effet, nous souhaitons observer la contribution de l'image lors de situations pour lesquelles les nuages de points comportent très peu de points. Nous n'étudions pas ici d'approches en fusion tardive (*late fusion*) pour lesquels plusieurs systèmes indépendants coopèrent (par exemple, dans le cas de la fusion des sorties de deux détecteurs différents).

La principale différence entre le réseau RGB-3D Pillars (que l'on notera CLVFE pour Caméra-LiDAR *Voxel Feature Encoder*) et le réseau Médoïde (CLMed pour Caméra-LiDAR Médoïde) provient de la manière dont sont encodées les cellules de la grille BEV. Dans le cas du premier réseau, tous les points 3D sont utilisés pour échantillonner la carte produite par l'image. Chaque cellule est alors décrite par un vecteur issu de la fusion des informations liées aux points de la cellule par un *Voxel Feature Encoder*. Dans le second cas, seuls les médoïdes sont utilisés pour échantillonner l'image. Chaque cellule ne dépend alors que d'un seul point. L'encodage de la cellule est directement lié à l'échantillonnage sur l'image.

5.4.1 Réseau 3D

Dans cette section, nous présentons le réseau n'utilisant que les nuages de points 3D. Le système utilisé est illustré sur la figure 5.7.

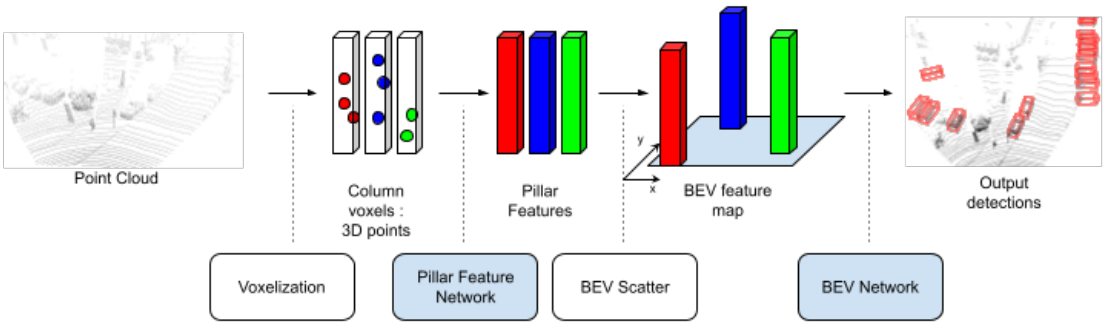


FIGURE 5.7 – Architecture globale du réseau.

Le processus reprend celui de PointPillars (LANG et al., 2019) : le nuage de points est d'abord discrétisé en voxels « colonnes ». Ces voxels sont traités par un *Pillar Feature Network* pour transformer chaque voxel en un vecteur caractéristique. Ces vecteurs sont positionnés au sein d'une carte BEV qui sera ensuite traitée pour produire les détections finales. Nous présentons les différents éléments du diagramme dans les paragraphes suivants.

Choix des données d'entrée

Lorsque l'on travaille sur un seul jeu de données ou une seule configuration de capteurs, on tend à exploiter le maximum d'informations. Néanmoins, certaines informations délivrées par deux modèles du même type de capteur peuvent grandement varier. Dans le cas des caméras, ces divergences sont nombreuses. On peut notamment citer la résolution de sortie, la sensibilité du capteur, le profil colorimétrique…. Ainsi, une même scène peut avoir un rendu très différent entre deux caméras. Dans le cas de capteurs LiDAR, ces différences sont plus faibles. Le rôle de ces capteurs est principalement de délivrer une représentation de la scène sous forme de nuage de points. La résolution des capteurs intervient alors dans la distribution des points dans la scène, bien que la géométrie sous-jacente, elle, ne varie pas, ce qui tend à « normaliser » les sorties. En outre, les capteurs ont des précisions et biais variables (diffusion du tir laser, incertitude sur la distance de l'impact...). Ces erreurs se réduisent avec l'évolution de la technologie afin de reproduire au mieux les surfaces de la scène. En revanche, l'un des aspects sur lequel les fabricants de LiDAR ne s'accordent pas concerne les informations supplémentaires renvoyées avec les points comme la réflectance. Les émetteurs et récepteurs ainsi que les longueurs d'onde et les modèles de calcul utilisés peuvent grandement varier d'un modèle à l'autre. Pour une même grandeur physique, les valeurs renvoyées diffèrent entre capteurs. De ce fait, nous n'utilisons que les informations géométriques, la réflectance n'est alors pas prise en compte.

Traitement des nuages de points

Comme énoncé précédemment, nous utilisons la méthode utilisée pour le réseau PointPillars. Dans une zone de recherche délimitée entre x_{min} et x_{max} sur l'axe x et y_{min} et y_{max} sur l'axe y, le nuage de points est discrétisé en cellules sans découpage vertical, de sorte à obtenir des « colonnes ». Ce processus est réalisé afin d'obtenir une grille creuse de $x_{num} \times y_{num}$ cellules. La taille des côtés de chaque cellule sur le plan xy est donné par $x_{res} = \frac{x_{max} - x_{min}}{x_{num}}, y_{res} = \frac{y_{max} - y_{min}}{y_{num}}$.

Réseau de neurones

Un réseau dont l'architecture est inspirée par celle de PointPillars (LANG et al., 2019), réputé rapide et relativement efficace, est mis en oeuvre. En outre, il n'est constitué que des opérateurs standard permettant l'export vers d'autres *frameworks* d'apprentissage profond à l'aide de ONNX (Open Neural Network eXchange) (*Site officiel de ONNX*).

Le réseau de neurones est donc constitué de deux parties :

- un sous réseau, le *Pillar Feature Encoder*, dont le rôle est de transformer l'ensemble des points 3D d'un voxel en un vecteur représentatif, chaque voxel étant traité indépendamment des autres ;
- un autre sous-réseau, le *BEV network*, qui est chargé de traiter la carte en BEV générée à partir des vecteurs associés aux voxels afin de produire des détections.

Pour un pilier/voxel constitué de N points, chaque point étant représenté par ses coordonnées cartésiennes $[x,y,z]^T \in \mathbb{R}^3$, on note (x_c,y_c,z_c) les distances sur chaque axe du point à la moyenne des points du voxel, et (x_p,y_p) les distances du point par rapport au centre du pilier sur le plan xy. Chaque point est donc représentable sous la forme d'un vecteur $p_{aug} = (x,y,z,x_c,y_c,z_c,x_p,y_p)^T$ de dimension $D = 8$.

Tous les voxels n'ont au départ pas le même nombre de points. Un prétraitement est réalisé afin de normaliser les voxels. Pour une valeur P fixée, si le nombre de points dans le voxel dépasse P, P points sont tirés aléatoirement de l'ensemble. Si le nombre de points est inférieur à P en revanche, certains points sont ré-échantillonnés afin d'atteindre les P points.

L'ensemble des voxels non vides est finalement représentable sous la forme d'un tenseur de taille (N,P,D) avec N le nombre de voxels non vides.

Le *Pillar Feature Encoder* consiste, comme dans l'article original en un réseau PointNet simplifié : une couche linéaire suivie d'un opérateur BatchNorm, d'une activation ReLU et d'un opérateur Max. Le tenseur de voxel est traité par ce réseau pour produire un nouveau tenseur de dimensions (N,P,D'). Un opérateur Max est alors appliqué uniquement sur la dimension des points pour produire un ensemble de vecteurs de dimensions (N,D'). Nous fixons $D' = 64$.

À partir de la position des voxels dans la scène, il est alors possible de générer une pseudo-image avec les vecteurs obtenus. Les pixels associés à des piliers vides ont des valeurs nulles.

Le réseau BEV est constitué d'un *backbone* directement issu du réseau PointPillars original et d'un bloc de sortie chargé de retourner les prédictions. La figure 5.8 décrit succinctement le processus de la carte générée aux prédictions finales.

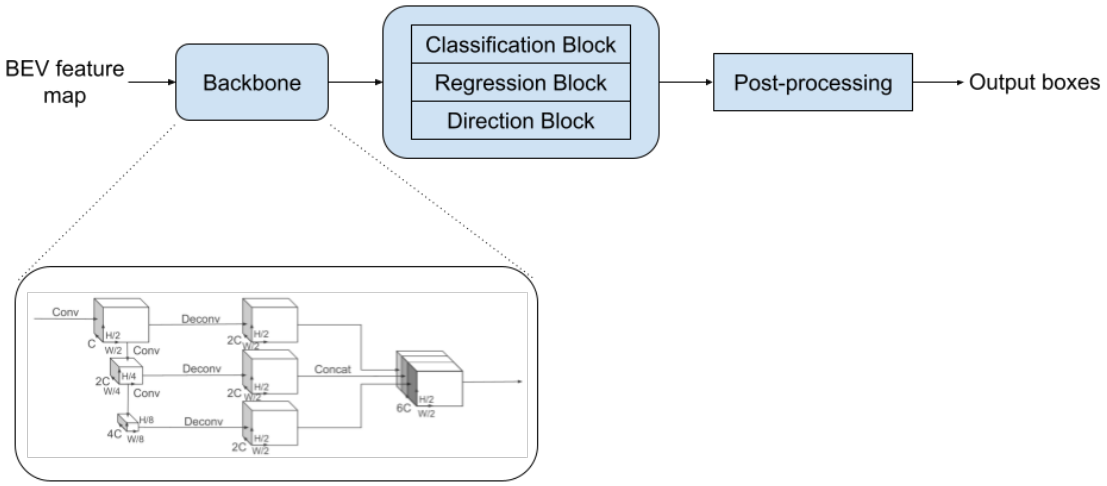


FIGURE 5.8 – Traitement de la carte BEV. L'image du backbone est issue de l'article de PointPillars (LANG et al., 2019).

Le *backbone* prend en entrée la pseudo-image de 64 canaux. On note $Bloc(L,F)$ un bloc constitué de L ensembles Convolution 2D − Batch Norm − ReLU et dont la sortie possède F canaux. On considère la pseudo-image de dimensions (H,W) de $D = 64$ canaux. Un premier $Bloc(4,64)$ génère une première carte de dimensions $(\frac{H}{2},\frac{W}{2})$. Cette carte alimente un second

Bloc$(6,128)$ pour générer une carte de dimensions $(\frac{H}{4},\frac{W}{4})$. Un troisième *Bloc*$(6,256)$ crée une carte de dimensions $(\frac{H}{8},\frac{W}{8})$. Les trois cartes sont redimensionnées à l'aide de convolutions transposées en tenseurs de dimensions $(\frac{H}{2},\frac{W}{2})$ et de 128 canaux puis sont concaténées pour produire un tenseur final de 384 canaux qui est transmis à un ensemble de blocs de sortie. Chacun des blocs en sortie du *backbone* (Classification, Régression, Direction) est une couche de convolution 1×1.

5.4.2 Réseau RGB-3D VFE

Ce réseau est une extension du réseau PointPillars présenté précédemment. En effet, en plus des informations géométriques contenues dans le nuage de points 3D, nous incorporons à chaque point des informations provenant de l'image.

Un premier réseau est chargé de transformer l'image RGB en une carte de caractéristiques. L'architecture de ce réseau est détaillée dans la figure 5.9

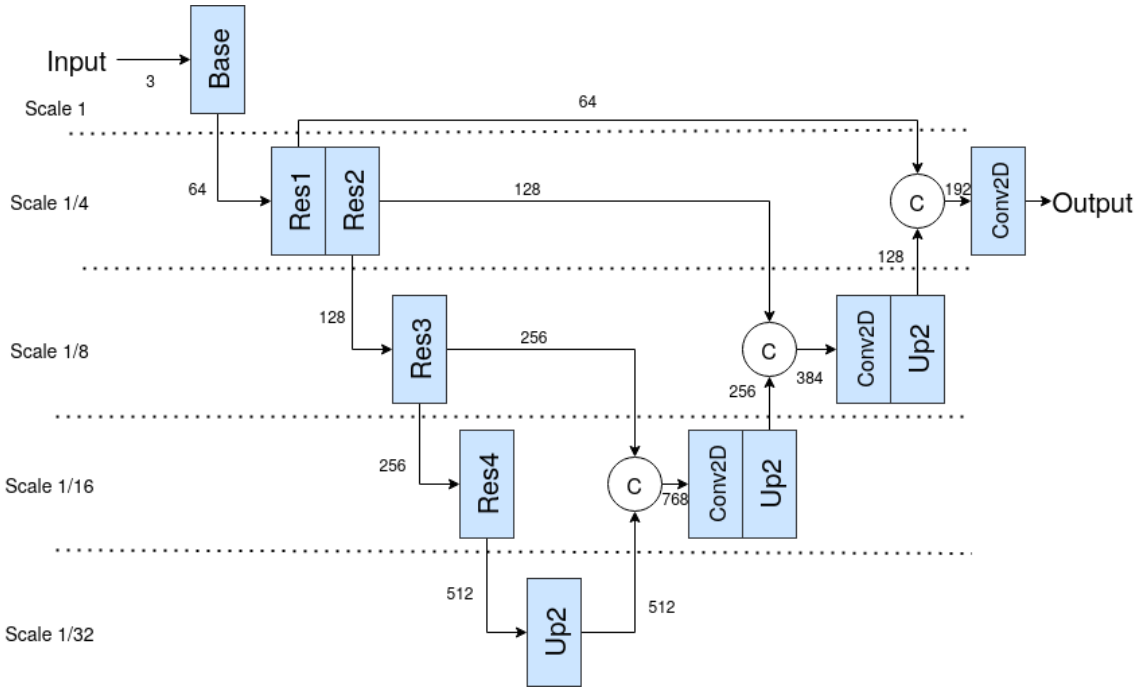


FIGURE 5.9 – Architecture dédié à l'image RGB.

Par la suite, Conv2D(c_{in}, c_{out}, k, s, p) désigne un opérateur de convolution 2D avec c_{in}, c_{out} le nombre de canaux d'entrée et de sortie, k la taille du noyau de convolution, s le pas (*stride*), p le remplissage au bords (*padding*). De même, MaxPool(k, s, p) désigne un opérateur de *Max Pooling*. Les blocs référencés Up désignent un redimensionnement, Linear(c_{in}, c_{out}) représente une couche linéaire avec c_{in}, c_{out} le nombre de canaux d'entrée et de sortie.

Le bloc nommé *Base*, utilisé dans l'extracteur de caractéristiques pour l'image RGB, est constitué d'une Conv2D(3, 64, 7, 2, 3) (pour les 3 canaux RGB), d'une *Batch Normamization*, d'une activation ReLU et d'un MaxPooling(3, 2, 1). Les blocs étiquetés Res_i proviennent d'une architecture ResNet18 (He et al., 2016). Le dernier bloc Conv2D renvoie une carte de 64 canaux.

Connaissant les projections de chaque point dans le plan image de la caméra, les caractéristiques sont extraites de la nouvelle carte et concaténées aux coordonnées 3D des points avant voxelisation.

5.4.3 Réseau RGB-3D Médoïde

Nous conservons la voxelisation colonne appliquée précédemment. Néanmoins, plusieurs modifications sont réalisées.

La principale modification concerne la représentation des voxels. En effet, ici nous représentons chaque voxel par un seul point, le médoïde. La gestion des informations issues de l'image RGB a principalement influencé ce choix. En réduisant le nombre de points d'échantillonnage sur l'image, le sous-réseau image doit apprendre à produire durant l'entraînement des informations plus représentatives du contenu de la région associée au voxel. En outre, lorsque l'on utilise un ensemble de vecteurs pour représenter un seul voxel, une variation des caractéristiques au niveau d'un seul point échantillonnage peut altérer le vecteur résultant. Pour un voxel non vide, nous notons x_c, y_c la position du centre du voxel sur le plan (Oxy). Le médoïde $p_{main} = [x_{main}, y_{main}, z_{main}]^T$ des points à l'intérieur du voxel est également calculé. Le médoïde a été choisi à la place de la moyenne car il n'est pas généré, mais appartient bien à l'ensemble de points originels. Comme nous considérons des voxels colonnes, ils sont plus représentatifs, notamment dans le cas de régions non connexes. Le voxel est alors encodé par un vecteur $\mathbf{v} = [x_c - x_{main}, y_c - y_{main}, z_{main}]^T \in \mathbb{R}^3$. Cette représentation permet de traiter chaque voxel indépendamment de sa position dans la scène afin que le réseau ne crée pas d'hypothèses sur la position des objets dans la scène au niveau du nuage de points. De plus, nous calculons la projection du point p_{main} dans l'image à l'aide des paramètres de calibration. Contrairement au réseau précédent qui utilisait l'intégralité du nuage pour récupérer des informations de l'image, seuls les points p_{main} sont utilisés ici pour extraire les caractéristiques de l'image comme le montre la figure 5.10.

Dans la transition entre la carte issue de la carte RGB et la carte BEV, nous intégrons également une couche pour la classification de voxels, uniquement à partir des informations RGB. Cela constitue en somme une segmentation sémantique « creuse » de l'image. Cette couche de classification consiste en une couche Linear(64, n_{cls}) avec n_{cls} le nombre de classes étudiées. Ces mêmes caractéristiques provenant de l'image sont concaténées aux informations 3D puis utilisés par une couche Linear(64+3, 64) pour être intégrées dans la carte BEV avant application du réseau BEV.

5.5 Entraînement

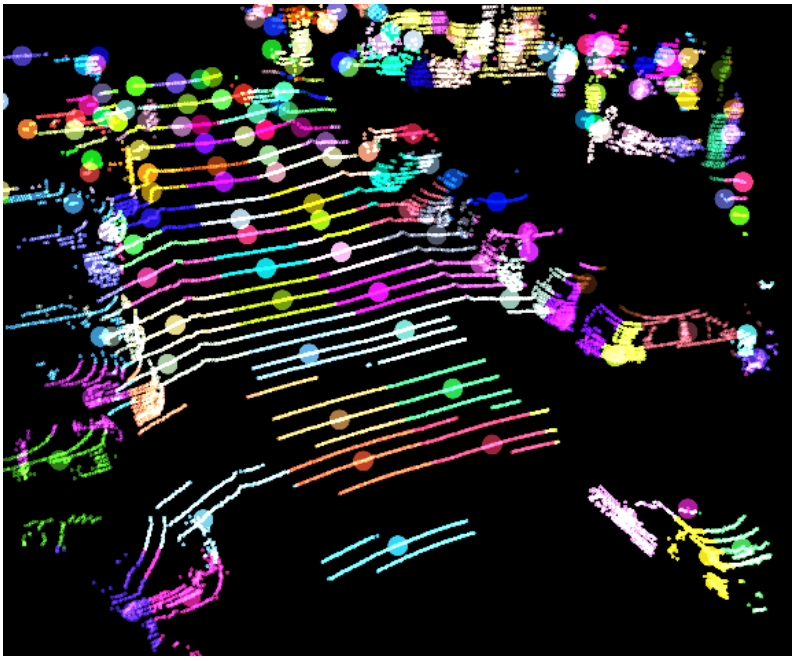
5.5.1 Sortie du réseau

Cette section s'applique aux trois réseaux étudiés. Le réseau renvoie trois cartes de mêmes dimensions : une carte pour la classification en BEV, une carte d'estimation des paramètres des boîtes et une carte de sélection de l'orientation. De plus, au niveau de la phase de transition, une classification des voxels est réalisée.

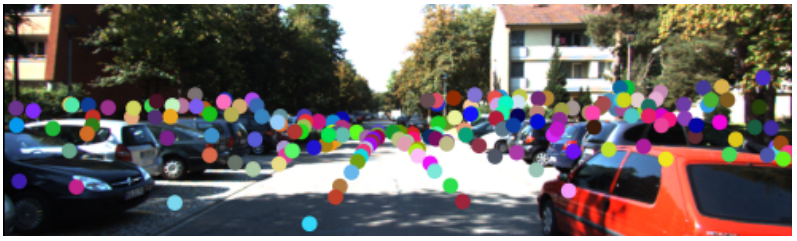
Nous adoptons une approche sans ancres inspirée par (Zhou, Wang et Krähenbühl, 2019) pour la représentation des obstacles sur les cartes de sortie. Chaque objet est représenté dans la carte de classification comme un seul pixel, ce pixel étant associé au voxel auquel appartient le centre de l'objet. Cette approche a été choisie afin de réduire le nombre d'hyperparamètres pouvant affecter l'entraînement. L'extraction des boîtes englobantes à partir de la carte de classification consiste d'une part en l'extraction des maxima dans la carte de classification, d'autre part à récupérer les paramètres aux positions de ces maxima.

5.5.2 Génération des vérités terrain

Chaque annotation i est définie par une classe et des paramètres de boîtes $[x_i, y_i, z_i, h_i, w_i, l_i, \theta_i]^T$. Les cartes de vérité terrain utilisées pendant l'entraînement sont générées suivant un processus inspiré par (Zhou, Wang et Krähenbühl, 2019) et (Ge et al., 2020). La carte de classification



(a) Visualisation de la voxelisation, chaque grand point représente un médoïde de voxel



(b) Projection des médoïdes dans l'image

FIGURE 5.10 – Relations entre les voxels dans l'image et les voxels dans la BEV. Un voxel est représenté par une même couleur dans les deux vues.

$M \in \mathbb{R}^{H \times W}$ est d'abord générée, H, W correspondant aux dimensions de la carte. Pour chaque annotation, notons (u_i, v_i) le pixel de M correspondant au centre de cette annotation. On génère ainsi une première carte dont les valeurs sont nulles sauf aux pixels liés aux annotations. Cette première carte permet de générer une transformée de distance euclidienne d pour chaque classe. Les valeurs de M sont alors définies pour l'objet i par :

$$M(u,v) = \left\{ \begin{array}{lll} 1 & \text{si} & d(u,v) = 0 \leftrightarrow (u,v) = (u_i, v_i) \\ 0.8 & \text{si} & d(u,v) = 1 \\ \frac{1}{d(u,v)} & \text{sinon} & \end{array} \right\}, \tag{5.3}$$

ces valeurs étant issues de (GE et al., 2020).

La figure 5.11 illustre un exemple de carte de classification générée à l'aide de cette méthode.

La carte d'estimation de paramètres est constituée de sept canaux, un par paramètre de boîte. On note Δx le premier canal, Δy le second canal, etc. Les voxels sont définis selon une grille avec positions fixes. Pour l'annotation i, son voxel positif associé, dont le centre est situé à $(\tilde{x}_i, \tilde{y}_i)$ dans l'espace métrique et à (u_i, v_i) dans les cartes, les valeurs visées pour la régression sont définies comme :

$$\begin{aligned} \Delta x = \tilde{x}_i - x_i,\ \Delta y = \tilde{y}_i - y_i,\ \Delta z = z_i,\ \Delta h = h_i, \\ \Delta w = w_i,\ \Delta l = l_i,\ \Delta \theta = \theta_i \end{aligned} \tag{5.4}$$

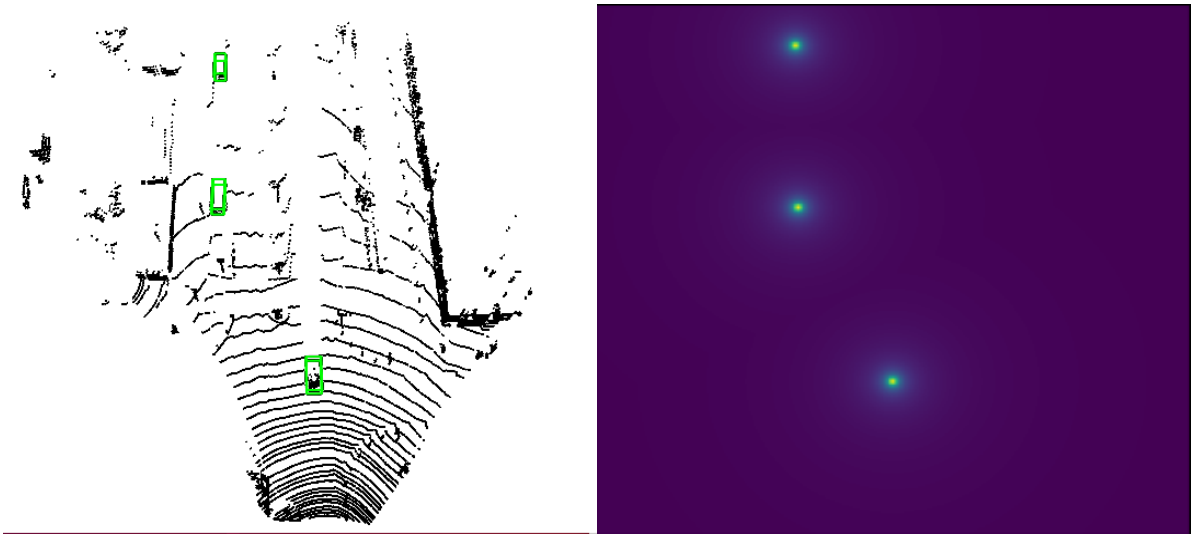


FIGURE 5.11 – Vérité terrain pour la classification/détection de points clés.

La carte de direction est une carte en quatre canaux, chaque canal représente l'un des quatre quadrants définis dans la figure 5.12. Comme énoncé plus tôt, cette carte permet la sélection de l'un des cas décrits figure 5.5.

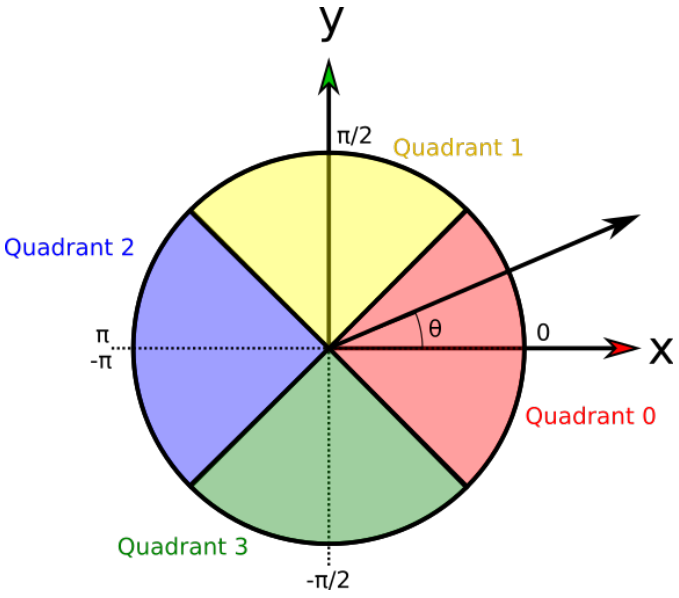


FIGURE 5.12 – Représentation des quatre quadrants utilisés pour la carte d'orientation.

5.5.3 Fonction de coût

La fonction de coût utilisée est une fonction multitâche :

$$L = \gamma_{cls} L^{cls} + \lambda_{reg} L^{reg} + \lambda_{ori} L^{ori} + \gamma_{rgbvox} L^{rgbvox} \tag{5.5}$$

avec $L^{cls}, L^{reg}, L^{ori}, L^{rgbvox}$ les fonctions relatives à la détection des centres des obstacles/points-clés en BEV illustrée dans la figure 5.11, à la régression des paramètres, au choix de l'orientation et à la classification des voxels non vides à partir des données RGB respectivement, λ_{cls}, λ_{reg}, $\lambda_{ori}, \gamma_{rgbvox}$ sont les poids respectifs.

Nous détaillons chacune de ces fonctions dans les sections suivantes.

Points clés BEV L^{cls} est la moyenne de *focal losses* appliquées à chaque pixel de la carte traitant des points clés/centres des obstacles.

Régression des paramètres de boîtes Inspiré par la méthode présentée par (Simonelli et al., 2019), les paramètres des boîtes sont séparés en trois ensembles :

- $S_c = \{x, y, z, h\}$,
- $S_{dim} = \{w, l\}$,
- $S_\theta = \{\theta\}$,

S_{dim} et S_θ étant ceux provoquant les ambiguïtés de rotation.

Grâce à l'approche décrite dans la section 5.3.2, nous disposons d'une représentation permettant l'usage de distances statistiques entre une prédiction et une vérité terrain. Ici, nous avons utilisé une distance de Bhattacharyya. Si P, Q deux lois normales multivariées $P = \mathcal{N}(\boldsymbol{\mu}_P, \boldsymbol{\Sigma}_P), Q = \mathcal{N}(\boldsymbol{\mu}_Q, \boldsymbol{\Sigma}_Q)$, la distance de Bhattacharyya s'écrit :

$$D_B(P,Q) = \frac{1}{8}(\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q) + \frac{1}{2}\ln\frac{\det\boldsymbol{\Sigma}}{\sqrt{\det\boldsymbol{\Sigma}_P \det\boldsymbol{\Sigma}_Q}}. \tag{5.6}$$

avec $\Sigma = \frac{\Sigma_P + \Sigma_Q}{2}$. Nous définissons la fonction de coût pour la régression :

$$\begin{aligned} L^{reg} = \frac{1}{n_{pos}} \sum_{pos} & \quad SmoothL1(S_c, \hat{S}_c) \quad + \\ & D_B(F(S_c, \hat{S}_{dim}, S_\theta), F(S_c, S_{dim}, S_\theta)) \quad + \\ & D_B(F(S_c, S_{dim}, \hat{S}_\theta), F(S_c, S_{dim}, S_\theta)) \end{aligned} \tag{5.7}$$

avec *pos* les positions des voxels positifs, n_{pos} le nombre de voxels positifs, S et $\hat{S}$ les vérités terrain et les prédictions respectivement, F la fonction permettant de passer des paramètres à la représentation probabiliste.

Sélection de l'orientation L_{ori} est une fonction d'entropie croisée appliquée uniquement aux positions des pixels positifs.

Classification de voxels L^{rgbvox} est la moyenne des *focal losses* appliquées à chaque voxel non vide (dont il existe une projection sur l'image RGB).

5.5.4 Paramètres d'entraînement

Pour la détection de véhicules, nous fixons $[x_{min}, x_{max}] \times [y_{min}, y_{max}] = [0.0\,\text{m}, 70.4\,\text{m}] \times [-35.2\,\text{m}, 35.2\,\text{m}]$ Pour les deux réseaux utilisant des Pillar Feature Encoder, $n_x = n_y = 440$. Pour le dernier réseau cependant, pour compenser la diminution de la quantité d'informations 3D due à l'utilisation de médoïdes uniquement, nous fixons $n_x = n_y = 704$.

A chaque itération et pour chaque nuage, les nappes du nuage de points sont retirées aléatoirement de sorte qu'entre 35% et 95% des nappes du nuage soient conservées.

Tous les modèles sont entraînés sur 80 époques à l'aide d'un optimiseur Adam (Kingma et Ba, 2014) et un planificateur de taux d'apprentissage dit « *one-cycle* » avec un taux d'apprentissage maximal fixé à 0.001.

Les valeurs attribuées à $(\lambda_{cls}, \lambda_{reg}, \lambda_{ori}, \lambda_{rgbvox})$ sont $(1.0, 1.0, 1.0, 0.5)$. En effet, les trois premières fonctions de coût sont directement liées à la détection des obstacles. Nous avons choisi d'attribuer une pondération identique aux trois objectifs. La quatrième fonction de coût est facultative. Néanmoins, elle peut être utilisée en tant que fonction auxiliaire pour conditionner

l'apprentissage du sous-réseau image. En effet, les tenseurs utilisés par cette fonction de coût sont calculés à partir de l'image RGB, les points 3D ne définissant à ce stade que les zones d'échantillonnage. La fonction de coût supplémentaire permet ainsi de diriger l'évolution des poids du réseau image de sorte que celui-ci remplisse les objectifs de segmentation image tout en étant utile à la détection des obstacles.

5.6 Expériences sur jeux de données annotés

Dans cette section, nous présentons les expériences menées ainsi que leurs résultats. Tous les réseaux sont entraînés sur le même ensemble de 3769 échantillons d'entraînement de la base KITTI. D'abord, nous évaluons les expériences réalisées sur le sous ensemble de validation de 3712 échantillons, sur nuages de points originaux (64 nappes) et fortement altérés (8 nappes aux latitudes régulièrement espacées). Ce découpage 3769/3712 est considéré comme standard par la plupart des auteurs de la littérature. Puis, nous utilisons les données de Pandaset et nuScenes comme données de test afin d'évaluer les capacités des réseaux sur des données totalement inconnues.

5.6.1 Méthodologie pour la comparaison

Dans cette section, nous étudions l'utilisation de réseaux pré-entraînés sur KITTI vers d'autres jeux de données. Nous nous focalisons sur deux jeux de données : nuScenes et Pandaset. Comme illustré dans le chapitre 2, les trois jeux de données sont équipés de capteurs différents, notamment au niveau des capteurs LiDAR. Nos travaux visent à transférer la complexité de la tâche d'identification des objets du LiDAR vers l'image. Jusqu'à présent, nous avons généré des situations dans lesquelles le nuage de points est altéré, mais le domaine image reste inchangé. L'utilisation d'autres jeux de données permet l'analyse sur d'autres environnements et conditions de capture.

Conditions expérimentales

Chaque jeu de données est le résultat de choix de conception propres et dispose de ses propres biais. De ce fait, pour une même tâche, les critères requis peuvent grandement varier d'un jeu à l'autre. Nous détaillons ici nos choix pour permettre la comparaison de résultats dans un même environnement.

Formatage des données Le jeu de données KITTI repose sur un LiDAR mécanique à 360° et des caméras avant. Toutefois, les annotations n'existent que dans le champ de vision caméra. Pour les autres jeux de données nous effectuons d'abord les transformations géométriques sur le nuage de points afin d'obtenir un repère similaire à celui de KITTI. De plus, bien qu'elles disposent de plusieurs caméras permettant l'acquisition des images sur 360°, nous nous limitons aux données des caméras frontales. Tout point 3D invisible par la caméra frontale est retiré du nuage d'entrée. Dans le cas de nuScenes reposant sur un système de *sweeps*, nous ne faisons pas d'accumulation de nuages de points et n'utilisons que le nuage synchronisé avec l'image.

Définition des annotations Chaque jeu de données définit sa propre représentation des obstacles. Ainsi, dans nuScenes, les rotations sont décrites à l'aide de quaternions, sans contraintes sur les axes de rotation. Des attributs désignant un état des obstacles est également ajouté. Ainsi, une voiture peut avoir un statut « En mouvement », « À l'arrêt » ou « En stationnement ». Pour l'étude, comme le réseau est entraîné sur KITTI, nous gardons le formalisme de cette base : classe et score, position 3D, dimensions 3D, angle autour de l'axe vertical.

Conditions d'entraînement Tous les modèles étudiés sont entraînés avec le nombre d'époques, le même nombre d'itérations et le même planificateur pour le taux d'apprentissage. Ce choix a été réalisé malgré le nombre de paramètres plus élevé dans les réseaux utilisant les images, rendant la convergence plus difficile. Bien que ces réseaux nécessitent des entraînements plus poussés, nous souhaitons d'abord observer les effets induits par les différents modificateurs appliqués plutôt que chercher à obtenir une performance maximale.

Évaluation Nous utilisons toujours l'AP définie par IoU pour l'évaluation. Toutefois, les difficultés définies pour le *benchmark* KITTI reposent sur des caractéristiques liées à l'image. Les caractéristiques caméra étant très différentes d'un jeu de données à l'autre, il n'est pas possible d'appliquer les mêmes critères de sélection de difficulté. De ce fait, nous avons fait le choix d'assembler toutes les détections dans un seul groupe. En outre, les dimensions aussi sont sujettes aux biais des annotateurs. Sachant que seule une partie des véhicules est visible, il peut y avoir de légères différences de dimensions sur un même modèle de véhicule. Pour cette raison, nous calculons l'AP sur une valeur de seuil d'IoU de 0.5 afin de compenser cette variabilité.

Distance maximale Les choix des concepteurs de la base conditionnent également la distance maximale jusqu'à laquelle les obstacles sont annotés. nuScenes par exemple prend en compte pour son évaluation propre les véhicules jusqu'à 50 m tandis que Pandaset peut fournir des annotations au-delà de 100 m. Cependant, les réseaux entraînés sur KITTI sont paramétrés pour une distance maximale de 80 m. Ainsi, nuScenes, la détection est d'abord réalisée sur 80 m puis toutes les cibles au-delà des 50 m sont retirées. Dans le cas de Pandaset, nous ne prenons pas en compte les annotations au-delà de 80 m.

5.6.2 Expériences menées

Pour chacun des réseaux étudiés, nous avons mené un certain nombre d'expériences. Chaque expérience est référencée selon la structure *Réseau-NuagePoints-Loss*. Pour la partie *Réseau*, on désigne par « L » le réseau LiDAR 3D, « CLVFE » le réseau caméra-LiDAR utilisant les VFE pour encoder les voxels, et « CLMed » le réseau caméra-LiDAR n'utilisant que les médoïdes pour représenter les voxels. L'entraînement des réseaux étant réalisé avec la base KITTI, « 64 » indique qu'aucune modification n'a été appliquée au nuage de points (donc 64 nappes). « Var » indique que des nappes sont retirées aléatoirement du nuage d'entrée au cours de l'entraînement tandis que « 8 » indique l'utilisation de nuages composés de 8 nappes réparties uniformément selon les latitudes. Enfin, la fonction de coût choisie est représentée par « L1 » si les paramètres (w, l, θ) sont estimés à l'aide d'une fonction *Smooth L1*, ou « B » si la distance de Bhattacharyya est utilisée. Dans le cas des réseaux « CLMed », la mention « VoxCls » indique l'intégration de la segmentation des médoïdes/voxels. La partie *NuagePoints* caractérise les modificateurs appliqués aux nuages de points.

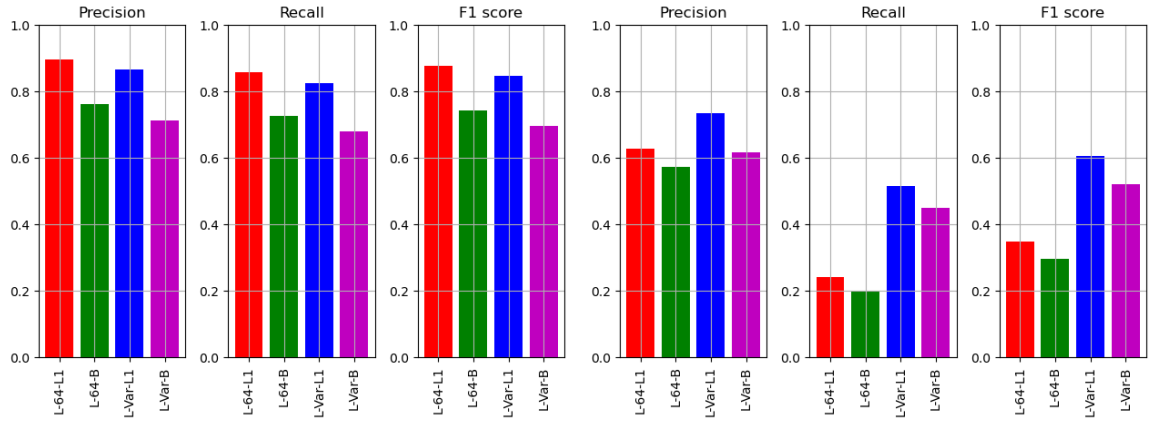
5.6.3 Expériences sur KITTI

Dans cette section, nous réalisons des études d'ablation afin d'évaluer la contribution des éléments proposés. Les études seront réalisées sur deux cas : le nuage original constitué de 64 nappes et le nuage altéré de sorte qu'il ne reste que huit nappes. Cette expérience permet de vérifier si le réseau est capable d'identifier des obstacles avec très peu de points. Comme nous utilisons des nuages très épars, nous nous focalisons sur la détection BEV. Les résultats sont présentés dans le tableau 5.1. Il est à noter que le réseau CLMed n'a pas été entraîné sur 8 nappes. En effet, il s'agissait du premier réseau développé pour cette étude. Nous n'avions pas à ce moment du développement pas songé à explorer cette piste.

TABLE 5.1 – Résultats pour la détection BEV sur le jeu de validation de KITTI. Les nuages de points ne sont pas altérés. Les valeurs sont calculées par *Average Precision* (%) et un seuil d'IoU fixé à 0.7.

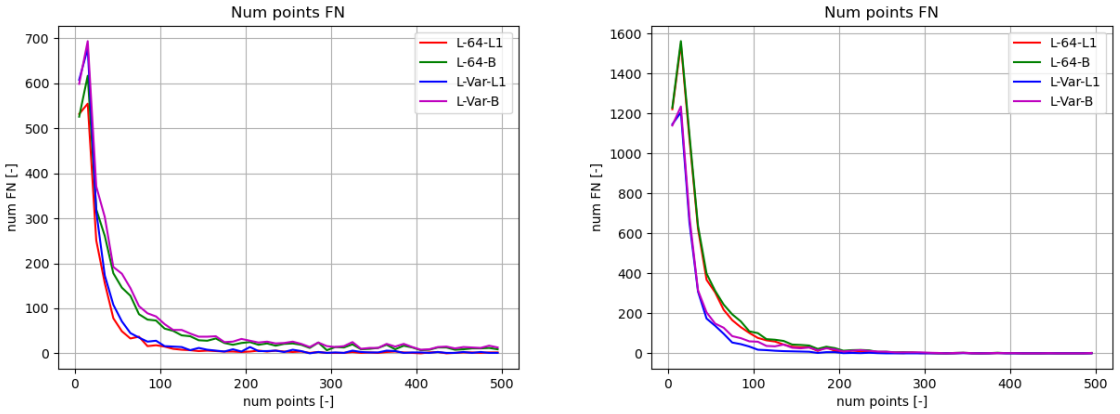
ID Exp.	64 nappes			8 nappes		
	Facile	Moyen	Difficile	Facile	Moyen	Difficile
CLMed-64-L1	87.80	69.41	71.31	37.08	25.88	25.53
CLMed-Var-L1	87.57	70.76	72.35	74.89	**55.04**	**55.36**
CLMed-64-B	87.68	68.42	70.57	48.31	33.56	28.40
CLMed-Var-B	87.11	69.37	71.29	74.58	53.78	54.12
CLMed-64-B+VoxCls	87.55	68.60	70.64	46.99	32.30	27.48
CLMed-Var-B+VoxCls	87.46	70.18	71.84	**75.92**	54.51	55.07
CLVFE-64-L1	75.02	59.22	54.71	41.81	29.48	27.29
CLVFE-64-B	72.07	53.10	53.31	42.11	29.59	27.47
CLVFE-Var-L1	77.46	58.49	54.50	61.57	40.46	38.83
CLVFE-Var-B	70.18	51.90	52.55	57.14	38.61	36.34
CLVFE-8-L1	71.55	58.72	54.57	65.30	46.96	41.10
CLVFE-8-B	65.55	48.19	47.92	64.34	45.12	40.06
L-64-L1	85.35	75.16	71.13	27.55	20.95	18.21
L-64-B	84.93	75.02	71.59	27.21	18.99	16.62
L-Var-L1	86.46	75.69	74.43	58.61	40.34	35.81
L-Var-B	85.90	75.45	72.30	58.16	40.56	35.48
L-8-L1	48.54	44.18	44.53	67.65	48.26	43.62
L-8-B	43.85	42.18	41.95	66.32	47.18	42.83
PointPillars (LANG et al., 2019)	89.65	87.17	84.37	46.99	32.34	27.85
PV-RCNN (SHI et al., 2020)	**90.26**	**88.04**	**87.39**	40.02	28.66	26.48

Nous avons effectué un ensemble d'analyses plus fines sur les expériences L-64-L1, L-64-B, L-Var-L1 et L-Var-B. La figure 5.13 représente les statistiques obtenues sur les données de *validation* de la base KITTI, la figure 5.14 se focalise sur le nombre d'obstacles manqués en fonction de leur nombre de points 3D et la figure 5.15 analyse les erreurs d'estimation des paramètres sur les vrais positifs.



(a) Résultats sur 64 nappes (b) Résultats sur 8 nappes

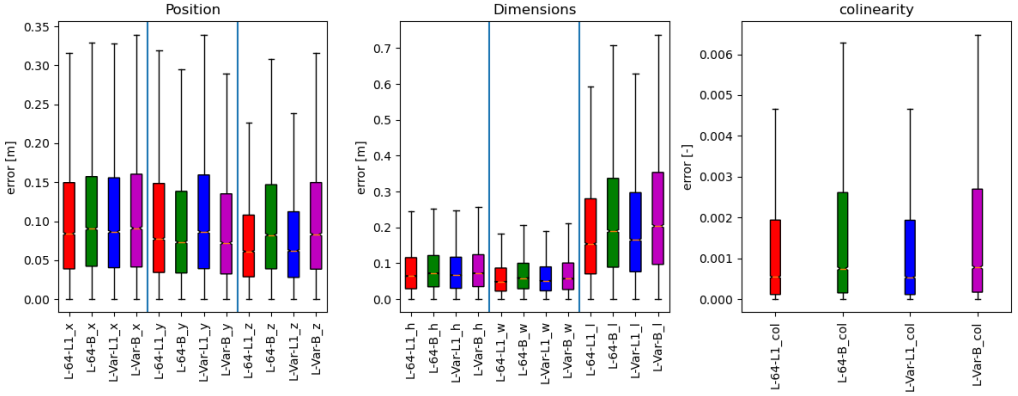
FIGURE 5.13 – Précision, rappel et F1-score obtenus sur le jeu de *validation* KITTI.



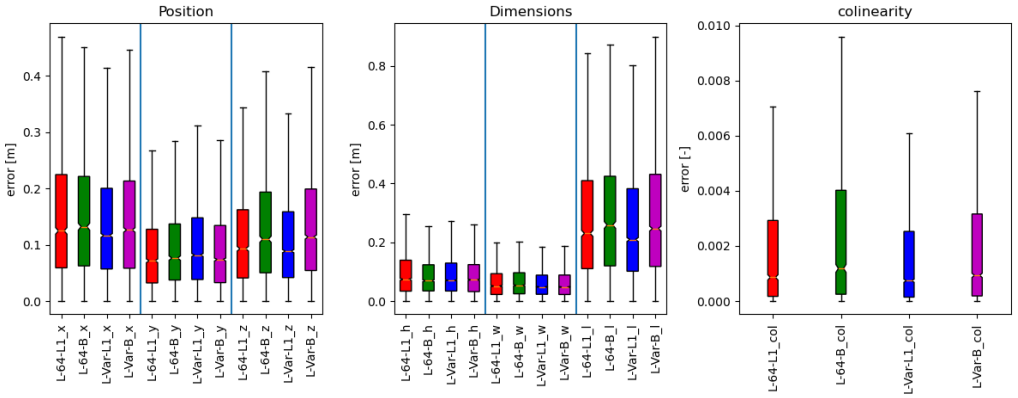
(a) Résultats sur 64 nappes (b) Résultats sur 8 nappes

FIGURE 5.14 – Nombre de faux négatifs manqués en fonction du nombre de points 3D inclus dans les annotations.

Comparaison globale entre modèles sur nuages à 64 nappes Nous étudions d'abord les résultats des différentes expériences dans les conditions standard d'utilisation (données non altérées du LiDAR Velodyne HDL-64E). Pour un entraînement standard et sur nuages de points non altérés, les scores de l'expérience CLVFE-64-L1 sont sensiblement inférieurs à ceux de L-64-L1 et de CLMed-64-L1. Tous les réseaux sont entraînés sur le même nombre d'époques. Alors que l'image est censée ajouter de l'information, la convergence s'est mieux réalisée sur L-64-L1 que sur CLVFE-64-L1. D'abord, l'ajout d'un réseau pour l'image accroît le nombre de paramètres, rendant l'apprentissage plus difficile, d'autant que ce sous-réseau image est situé loin de la sortie. En outre, contrairement à CLMed-64-L1, tous les points 3D sont utilisés pour échantillonnage de la carte de caractéristiques image, ce qui peut potentiellement accroître le nombre d'erreurs. Comme nous ne procédons pas à un découpage vertical, un voxel peut à la fois contenir le feuillage d'un arbre, une partie de la carrosserie d'une voiture et du sol, d'apparences très différentes mais regroupés sous le même voxel, ce qui peut grandement perturber les détections.



(a) Résultats sur 64 nappes



(b) Résultats sur 8 nappes

FIGURE 5.15 – Diagrammes en boites sur les erreurs d'estimation des paramètres des vrais positifs, plus basses sont les valeurs, meilleur est le détecteur. Chaque couleur correspond à une expérience. Le trait jaune indique la médiane des valeurs, les extrémités de la boite correspondent aux quartiles, les extrémités des lignes verticales noires correspondent aux erreurs minimales et maximales. Le terme « colinearity » est calculé avec $1 - |\cos(\theta_{gt} - \theta_{pred})|$.

Influence de la réduction de nappes pendant l'apprentissage Sur les nuages 64 nappes, on observe pour les trois réseaux de légères différences entre entraînements standard (64 nappes constantes) et entraînements avec réduction de nappes, La tendance globale est une légère dégradation des performances. Néanmoins, cette dégradation est compensée par le gain important dans des cas extrêmes où le nuage de points ne contient que 8 nappes. En effet, lors d'entraînements standard, le réseau devient plus spécialisé sur un seul type de distributions de points 3D mais toute autre distribution sera considéré comme une distribution inconnue qu'il ne saura pas correctement gérer. Lorsque le réseau observe des distributions différentes pour le même objectif, il établit un modèle plus versatile s'adaptant mieux. Ces observations sont également vérifiées lorsque l'on compare les valeurs de précision et de rappel des expériences L-64-L1 (rouge) et L-Var-L1 (bleu) sur la figure 5.13. Pour L-64-L1, le F1-score chute de 0.887 à 0.349 sur nuages à 8 nappes tandis que le F1-score de l'expérience L-Var-L1 passe de 0.846 à 0.606.

Nous constatons également que les réseaux utilisant les images semblent bénéficier au mieux de cette réduction du nombre de nappes. En effet, dans un nuage épars seul, il est difficile d'identifier les cibles. Avec un entraînement standard, bien qu'elles surpassent le réseau LiDAR, les valeurs restent assez faibles tandis qu'avec notre méthode, leurs performances augmentent très fortement sur les nuages très difficiles. Nous avons affiché les cartes obtenues après traitement des images RGB brutes par le réseau image sur la figure 5.16. Avec réduction de nappes, les cartes obtenues sont plus fines et détaillées. Le type d'entraînement a orienté la mise à jour des poids du réseau image afin d'acquérir des informations plus pertinentes pour améliorer les détections notamment lorsque les points 3D seuls ne suffisent plus à identifier une cible.

Enfin, nous avons tenté un entraînement sur des nuages épars uniquement (expériences L-8-L1 et CLVFE-8-L1). Dans le cas du réseau « L », l'expérience obtient de bons résultats sur sa distribution d'entraînement mais ne parvient plus à gérer les cas plus denses. En somme, comme pour l'entraînement à 64 nappes, le réseau a sur-appris sa distribution d'entraînement. Ce phénomène n'intervient cependant avec le réseau « CLVFE » grâce à l'utilisation de l'image pour la détection.

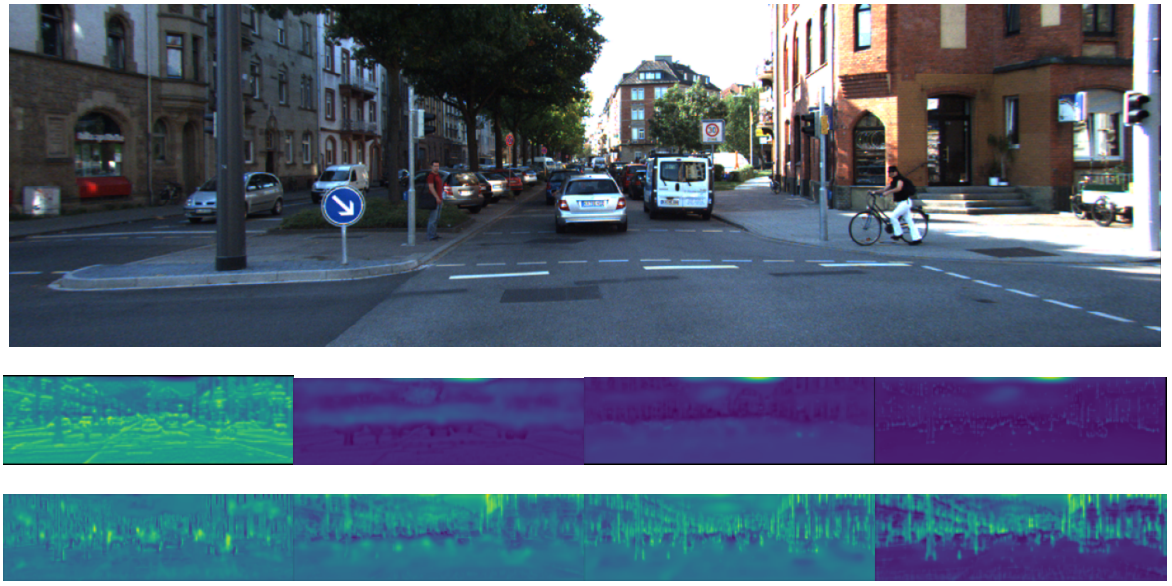


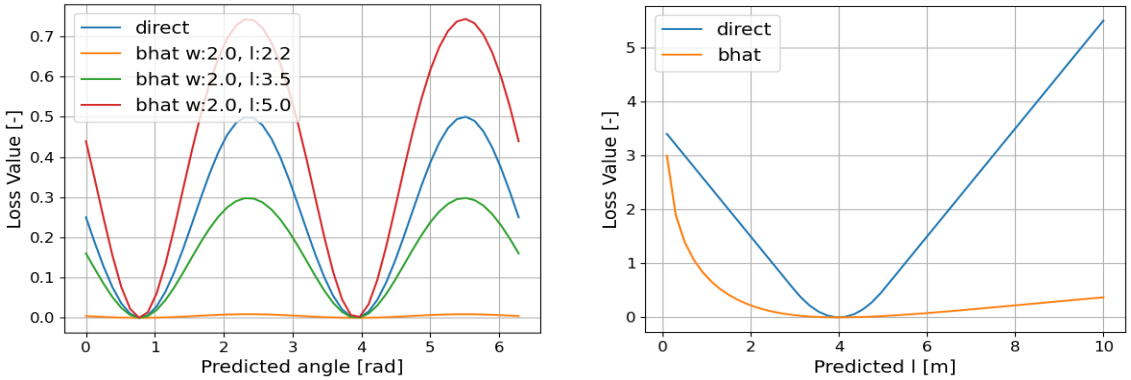
FIGURE 5.16 – Comparaison des cartes de caractéristiques issues de l'image RGB. Les valeurs sont normalisées entre 0 (violet) et 1 (jaune). Deuxième ligne : expérience CLMed-64-B, Troisième ligne : expérience CLMed-Var-B. Colonnes : canaux n° 0, 8, 16 et 32.

Influence de la représentation probabiliste Dans la majorité des cas, la représentation probabiliste tend à dégrader les performances des détecteurs. Cette dégradation est aussi bien

visible sur l'estimation des paramètres des boîtes comme l'indique la comparaison des expériences L-64-L1 et L-64-B sur la figure 5.15 que sur la classification, comme le montre le nombre accru de faux négatifs pour l'expérience L-64-B sur la figure 5.14.

La figure 5.17 illustre le profil de la fonction de coût de Bhattahcaryya D_B (Eq. 5.6) pour des variations de S_{dim} et S_θ. Ainsi, dans la figure 5.17a, tous les paramètres de boîtes sont fixes, excepté θ. La courbe a des variations similaires à la fonction sinus avec des minima globaux à 0 pour $\theta = k\pi, k \in \mathbb{Z}$. La principale spécificité de la fonction concerne l'influence des dimensions de l'objet visé sur la valeur maximale de la fonction de coût. Nous avons évalué D_B pour des cibles de différentes longueurs. Plus le ratio $\frac{l}{w}$ est proche de 1, plus les courbes de niveau de la gaussienne correspondante s'approche de cercles, moins l'espace occupé par la proposition change lorsque l'on applique une rotation sur cette dernière. Cela se traduit par une importance moindre des objets dont $w \to l$ dans la partie angulaire de la fonction de régression. Les objets allongés sont donc davantage ciblés sur la fonction de coût centrée sur θ. Concernant S_{dim}, le profil de D_B est affiché dans la figure 5.17b. La fonction de coût dispose d'un minimum global et la fonction est convexe. On note également que la valeur de la fonction est plus grande si la prédiction est plus petite que la boîte visée.

Néanmoins, comme on l'observe avec la figure 5.17a, les valeurs de la fonction de coût sont plus faibles que celles d'une fonction *Smooth L1* par exemple. Par conséquent, le réseau attribue moins d'importance à l'estimation des boîtes et est moins précis sur les paramètres, occasionnant ainsi la dégradation des performances lorsque le seuil d'IoU est élevé.



(a) Évolution en fonction de θ et de la taille de l'obstacle visé. $\theta_{gt} = \frac{\pi}{4}$rad

(b) Évolution en fonction de l.$l_{gt} = 4$m

FIGURE 5.17 – Profils de D_B sur un seul paramètre, les autres étant fixés à la vérité terrain. Les courbes étiquetées « direct » représentent les profils de la fonction *Smooth L1*.

Influence de la classification des voxels après le réseau RGB La motivation de la classification intermédiaire des voxels est de rendre l'extracteur de caractéristiques RGB indépendant. Si aucune contrainte n'est appliquée, les poids sont spécifiquement ajustés pour la tâche de détection 3D. Forcer une classification pousse le réseau RGB à apprendre des a priori relativement indépendants des points 3D. De plus, le résultat de la segmentation de l'image peut servir d'indicateur de la distance entre le domaine d'entraînement et le domaine d'utilisation effective.

Nous étudions ici les expériences CLMed-64-B, CLMed-Var-B, CLMed-64-B+VoxCls et CLMed-Var-B+VoxCls. Pour la procédure d'entraînement standard (expériences CLMed-64-B et CLMed-64-B+VoxCls), on observe de meilleurs résultats sur les nuages en haute et basse résolutions lorsque la classification de voxels est présente. Toutefois, avec la réduction de nappes (expériences CLMed-Var-B et CLMed-Var-B+VoxCls), l'expérience CLMed-Var-B+VoxCls est légèrement moins performante sur les nuages 64 nappes, mais gère mieux les cas de résolution plus réduite. L'utilisation d'une fonction de coût supplémentaire ainsi que la difficulté accrue

par la réduction des nappes rend la convergence plus difficile. Cependant, dans les cas difficiles à l'inférence, la segmentation aide à la meilleure identification des objets.

5.6.4 Études sur l'utilisation sur d'autres jeux de données

Dans cette section, nous étudions l'utilisation des réseaux précédents pré-entraînés sur KITTI sur d'autres jeux de données. Nous nous focalisons sur deux jeux de données : nuScenes et Pandaset. Comme illustré dans le chapitre 2, les trois jeux de données sont équipés de capteurs différents, caméras et LiDAR. Jusqu'à présent, nous avons généré des situations dans lesquelles le nuage de points est altéré, mais le domaine image reste inchangé. L'utilisation d'autres jeux de données permet l'analyse sur d'autres environnements et conditions de capture.

Le tableau 5.2 présente les résultats des différentes expériences sur ces deux jeux de données.

TABLE 5.2 – Évaluation sur nuScenes Mini et Pandaset. Les valeurs en en-tête correspondent au seuil d'IoU utilisé.

ID expérience	nuScenes 0.5	Pandaset 0.5
CLMed-64-L1	13.36	3.03
CLMed-Var-L1	30.73	39.18
CLMed-64-B	17.03	9.09
CLMed-Var-B	32.07	40.96
CLMed-64-B+VoxCls	22.61	4.38
CLMed-Var-B+VoxCls	33.82	39.76
CLVFE-64-L1	9.09	11.26
CLVFE-64-B	9.09	15.31
CLVFE-Var-L1	15.65	14.39
CLVFE-Var-B	16.61	7.41
CLVFE-8-L1	16.26	4.73
CLVFE-8-B	16.35	9.09
L-64-L1	37.69	49.35
L-64-B	50.12	**52.63**
L-Var-L1	47.86	49.00
L-Var-B	**66.91**	52.26
L-8-L1	55.04	16.82
L-8-B	44.43	11.58
PointPillars (LANG et al., 2019)	29.38	30.47
PV-RCNN (SHI et al., 2020)	32.96	39.73

Analyse des résultats sur nuScenes Mini Comme prévu, les performances ont grandement diminué sur le nouveau jeu de données. Les annotations ont, à l'origine, été conçues pour de l'accumulation de nuages de points. Par conséquent, n'utiliser qu'un seul nuage représente une difficulté importante. De plus, les nuages sont bien plus épars que ceux utilisés en entraînement.

Les réseaux utilisant les images ont subi une plus forte dégradation que le réseau « L » ce qui signifie que le réseau ne parvient pas à exploiter les nouvelles images. Les artefacts introduits par la nouvelle caméra, les nouvelles conditions d'enregistrement et les environnements inconnus différent grandement du contenu observé au cours de l'apprentissage.

La réduction du nombre de nappes au cours de l'entraînement améliore les performances du réseau LiDAR sur les nuages très épars de nuScenes. De plus, contrairement aux tests sur KITTI, la représentation gaussienne, elle aussi, améliore les performances. En effet, sur KITTI, la classification atteignait un palier, les erreurs provenaient principalement de l'estimation des paramètres des boîtes. Sur les nouveaux jeux de données, la problématique est plus axée vers l'obtention de scores de classification suffisants. Les erreurs sont plus dues à des erreurs dans la détection des obstacles. La représentation gaussienne a permis l'apprentissage d'une représentation des véhicules plus poussée, au détriment de la précision sur les boîtes. En outre, comme énoncé dans la section 5.6.1, si les biais entre annotateurs de différentes bases de données varient grandement, l'exactitude sur la partition d'entraînement peut introduire des erreurs dans la partition de test.

La figure 5.18 expose visuellement certains résultats de l'expérience CLMed-Var-B+VoxCls sur les données de nuScenes. La première colonne illustre une scène de nuit sur laquelle apparaissent des artefacts conséquents : flou de mouvement, bruit, reflets, etc. KITTI ne contient pas de scènes de nuits, donc les feux de circulation et les réverbères peuvent facilement provoquer des erreurs dans le traitement d'image, impactant à la fois la classification de voxels et la détection 3D.

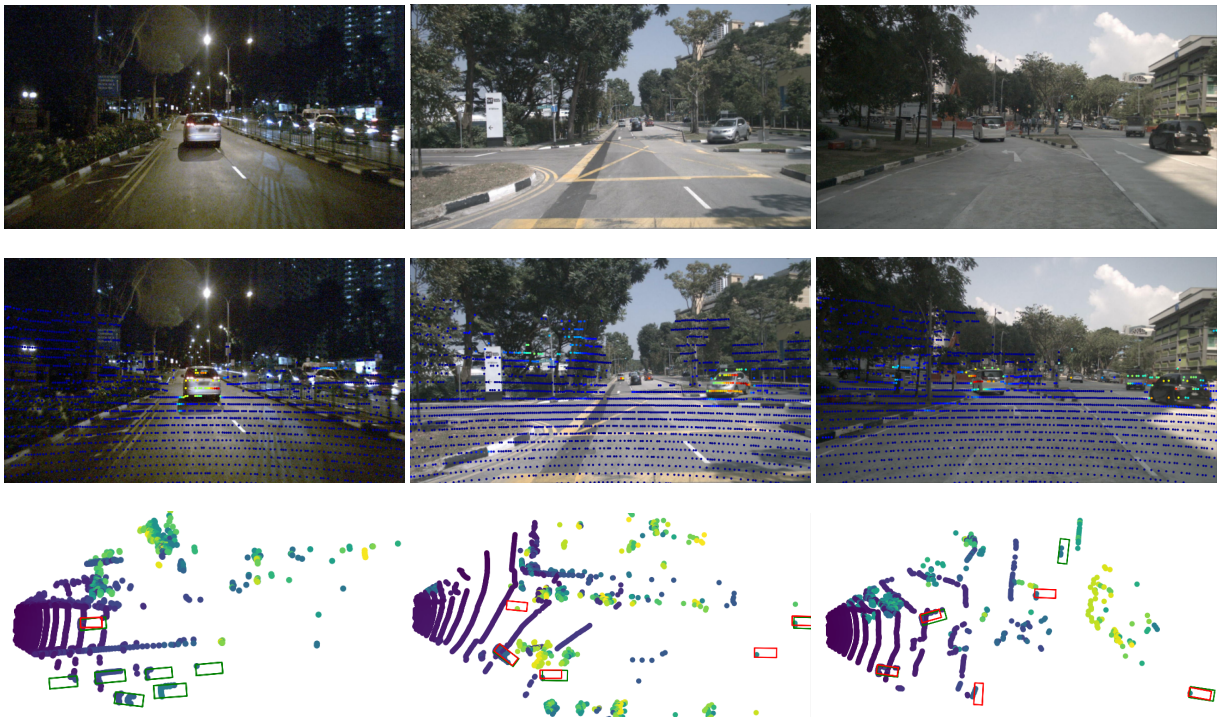


FIGURE 5.18 – Sorties de l'expérience CLMed-Var-B+VoxCls sur nuScenes. Première ligne : l'image d'entrée ; deuxième ligne : superposition avec la segmentation des voxels (bleu : 0, rouge : 1) ; troisième ligne : nuage de points, la vérité terrain en vert, les prédictions en rouge.

Analyse des résultats sur Pandaset

Les nuages de points de Pandaset ont une densité similaire aux nuages de KITTI mais pas les mêmes répartitions. Les expériences étiquetées CLVFE ont le plus de mal à généraliser. En effet, comme pour nuScenes, les artefacts de caméra ainsi que les nouveaux environnements génèrent des cartes de caractéristiques très différentes, dont les valeurs sont propagées au sein des *Pillar Feature Encoders* puis dans le réseau BEV. Les expériences CLMed-Var-L1, CLMed-Var-B et CLMed-Var-B+VoxCls, qui sont entraînées avec réduction du nombre de nappes parviennent

tout de même à fournir des scores plus élevés. En effet, la carte obtenue à partir de l'image se doit d'être plus explicite car le nombre de points d'échantillonnage avec les médoïdes est très inférieur au nombre de points dans le nuage entier. En somme, la sortie du réseau image est plus informative grâce à un échantillonnage plus limité.

La figure 5.19 présente les résultats du réseau produits par l'expérience CLMed-Var-B+VoxCls sur certaines scènes de Pandaset. On observe que malgré la différence de contexte, le réseau est tout de même capable de renvoyer des détections pertinentes. On note également que les scènes sont bien plus denses en obstacles que sur les autres bases.

En outre, la distribution des modèles de véhicules présents sur les routes peut grandement varier entre deux villes. Ainsi, dans les scènes de Pandaset, les véhicules larges comme les pickups sont bien plus présents que dans les scènes enregistrées en Allemagne pour la base KITTI. En somme, le réseau n'explore qu'une faible partie de la variabilité intra-classe.

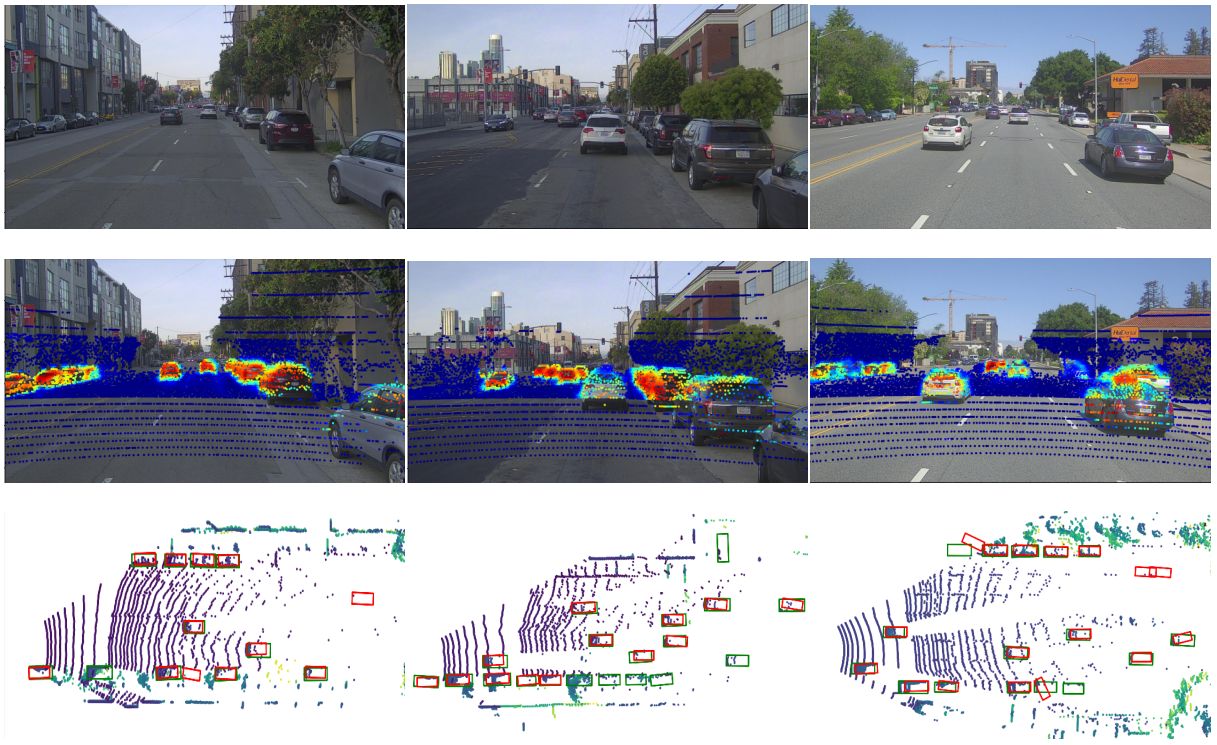


FIGURE 5.19 – Sorties de l'expérience CLMed-Var-B+VoxCls sur Pandaset. Première ligne : l'image d'entrée ; seconde ligne : superposition avec la segmentation de voxels (bleu : 0, rouge : 1) ; troisième ligne : nuage de points, vérité terrain en vert et prédictions en rouge.

5.7 Application sur données non annotées

Dans cette section, nous analysons les sorties obtenues par l'un des réseaux entraînées sur KITTI sur des données LiDAR non annotées.

Nous explorons deux jeux de données :

– les données de la base Robocar produite par l'Université de Technologie de Belfort-Montbéliard (UTBM). Ces données ont été à l'origine conçues pour de l'odométrie par LiDAR ou du SLAM (*Simultaneous Localization And Mapping*) ;

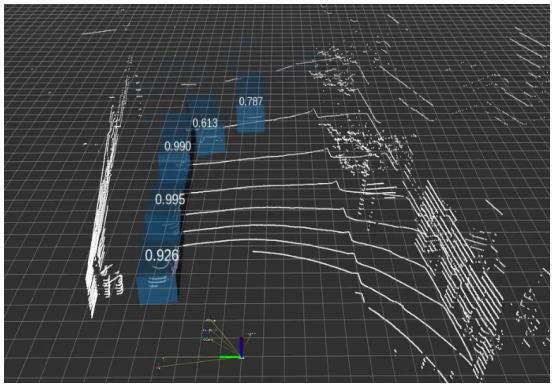
– le Ford AV Dataset, qui est une collection d'enregistrements réalisés par le constructeur automobile Ford dans la ville de Détroit.

Les véhicules des deux bases sont équipés respectivement de 2 et 4 LiDAR 32 nappes Velodyne HDL-32E. Cependant, pour les expériences, nous n'utiliserons que les capteurs avant coté conducteur. L'absence de calibration précise dans la base Robocar nous pousse à utiliser un réseau LiDAR. Nous utiliserons l'expérience L-Var-B, qui a donné les meilleurs résultats globaux. L'absence d'annotations ne mènera dans cette section qu'à des analyses qualitatives.

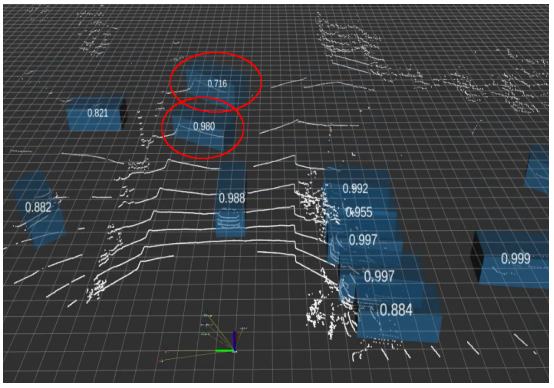
5.7.1 Tests sur UTBM Robocar

Les images de la figure 5.20 illustrent les résultats obtenus par le réseau. Globalement, le réseau parvient à bien identifier les voitures visibles. Avec le capteur et la configuration utilisée, les voitures sont visibles jusqu'à 25 m. Toutefois, la configuration utilisée produit un nombre assez important de faux positifs. Le phénomène est accentué lorsque aucun véhicule n'est présent dans la scène. Ces faux positifs sont en général sporadiques et instantanés et surviennent davantage lorsque l'ego-véhicule est à l'arrêt.

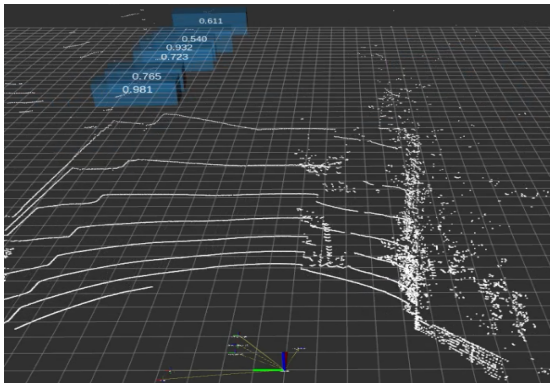
Certains jeux, comme celui de l'UTBM, ne disposent pas de paramètres de calibrage suffisamment précis pour réaliser des tests avec la fusion caméra-LiDAR. De ce fait, les résultats présentés dans ce chapitre ne concerneront que le détecteur n'opérant qu'avec le capteur LiDAR. Les données de Robocar sont fournies sous forme de rosbag. À noter que les données images et non images d'une même scène sont fournies sur deux rosbags distincts.



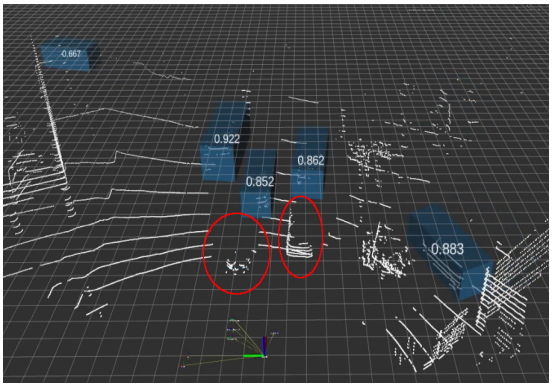
(a) Bonnes détections



(b) Points du sol confondus avec des voitures



(c) Nombre de points insuffisant pour déterminer s'il s'agit de véhicules ou d'infrastructures



(d) Véhicules proches non détectés

FIGURE 5.20 – Exemples de sorties sur les données UTBM Robocar.

5.7.2 Tests sur Ford Autonomous Vehicle Dataset

La figure 5.21 illustre quelques-uns des résultats obtenus sur des scènes représentant le campus de l'Université du Michigan-Dearborn et une route de zone commerciale américaine. Ici encore, le réseau parvient à détecter la grande majorité des voitures de tourisme, tout en respectant la contrainte temps-réel imposée par la fréquence du capteur. Dans les zones bétonnées, le détecteur commet peu d'erreurs. Toutefois, lorsque de la végétation est présente, le réseau tend à renvoyer des faux positifs un peu plus fréquemment dans ces zones.

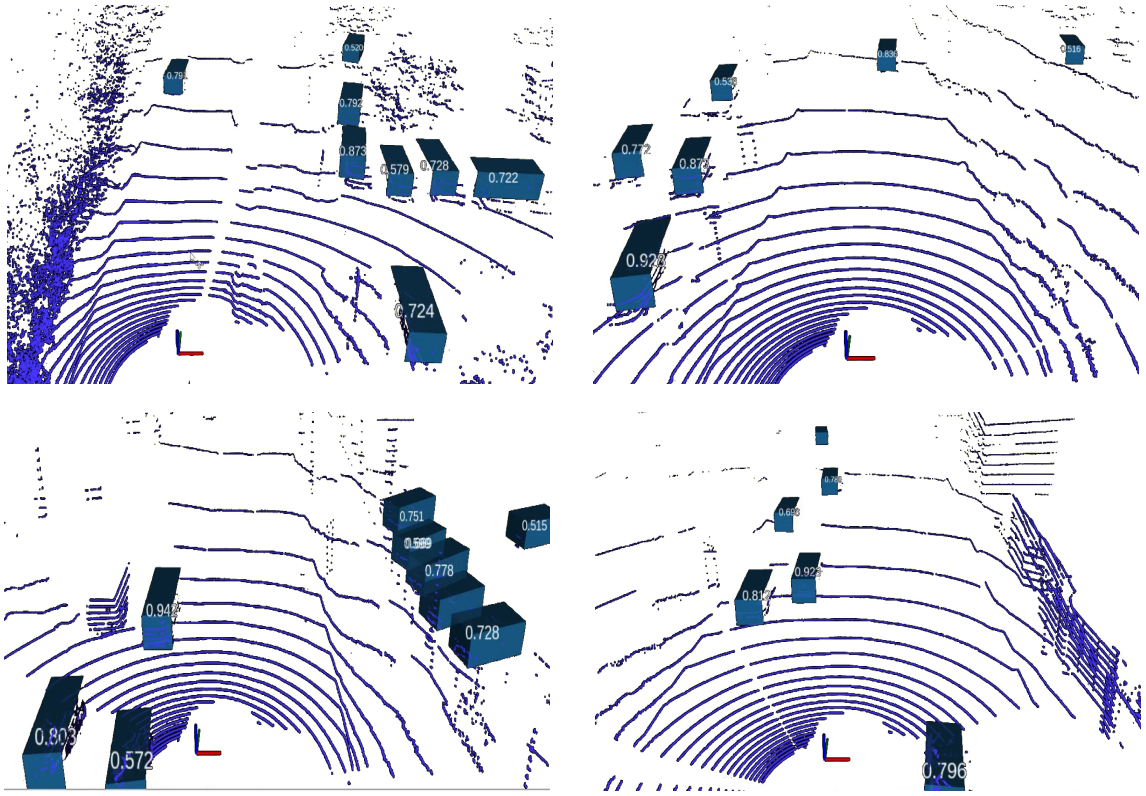


FIGURE 5.21 – Exemples de sorties sur les données Ford AV Dataset.

5.8 Conclusion et synthèse

Nous avons présenté dans cette section une étude concernant l'utilisation de détecteurs caméra-LiDAR et LiDAR sur des données provenant de capteurs et d'enregistrements très différents de la base d'apprentissage. Nous avons également présenté deux techniques permettant une meilleure résistance aux changements de cibles sans hypothèses préalables. La première technique consiste à réaliser un retrait aléatoire de nappes dans les nuages de points au cours de l'entraînement. Cela permet d'accroître la variabilité des données observées par les réseaux. De plus, dans le cas de réseaux utilisant également les images RGB, cette technique peut conditionner l'ajustement des poids au sein du réseau chargé de traiter ces images. De plus, nous définissons une nouvelle représentation des obstacles permettant une meilleure estimation des boîtes englobantes lorsque le nuage utilisé contient peu de points. Au lieu d'estimer directement les paramètres des boîtes, les obstacles sont représentés comme des lois normales permettant ainsi l'usage de distances statistiques. Ces deux techniques ont été mises à contribution sur trois réseaux. Quelles que soient les performances des méthodes exploitant les deux capteurs, nous avons montré grâce aux tests sur nuScenes et Pandaset que ces catégories de réseaux sont difficiles à appliquer sur des données de distributions différentes, ces réseaux étant notamment

soumis aux artefacts des nouvelles caméras et aux conditions d'enregistrement inconnues. Nous avons finalement mis à l'épreuve le réseau LiDAR entraîné uniquement sur KITTI avec nos deux techniques sur des données non annotées afin de valider notre approche.

Chapitre 6

Conclusion générale et perspectives

Sommaire

6.1 Contributions

La détection 3D à partir de nuages de points est un domaine très étudié dans la littérature. Cependant, de par la rareté des bases et de la non-uniformité des procédés d'annotation et d'évaluation, la très grande majorité des méthodes de la littérature ne sont entraînées et testées que sur un seul jeu de données. La première approche proposée est une méthode graphique permettant de représenter les véhicules sous forme d'ellipses directement à la sortie d'un réseau de neurones. Cette carte peut être utilisée en tant que pseudo-carte d'occupation. Toutefois, elle ne permet pas directement d'isoler les obstacles indépendamment les uns des autres. Pour combler cette lacune, une méthode d'ajustement d'ellipses est appliquée en sortie du réseau pour extraire les obstacles de la scène. La méthode est rapide et simple à mettre en place tout en restant efficace avec un AP 0.7 de 82.01% sur les cas les plus difficiles des données de validation. Cependant, deux inconvénients demeurent. Tout d'abord l'encodage utilisé convient aux véhicules et aux objets de grande taille, mais il n'est pas suffisamment représentatif pour d'autres cibles de plus petites tailles. Ensuite, dans le cas où les cibles sont représentées par peu de points 3D, les ellipses censées identifier les véhicules peuvent ne pas avoir de contours proprement définis, causant ainsi des erreurs sur l'estimation des dimensions.

Nous nous sommes focalisés ensuite sur l'aspect portabilité entre jeux de données. L'utilisation d'autres jeux de données fait l'objet dans la littérature de techniques visant à réadapter un système préexistant à de nouvelles données. En détection 3D, la recherche est souvent axée vers l'adaptation aux nuages de points. Toutefois, cela implique en général des connaissances définies au préalable par l'utilisateur ou bien acquises par le biais d'un nouvel apprentissage. Nous avons défini deux techniques appliquées lors de l'apprentissage. La première technique consiste en la réduction aléatoire du nombre de nappes dans le nuage de points 3D pendant l'entraînement. Cette technique permet de diversifier les données vues par le réseau pendant l'apprentissage et ainsi ne pas sur-apprendre une distribution unique de points. La seconde technique est un changement de représentation des obstacles. Chaque obstacle est représenté sous la forme d'une loi normale bivariée. Ce choix autorise alors l'utilisation de distances statistiques comme fonctions de coût plutôt que d'estimer les paramètres des boîtes englobantes indépendamment les uns des autres. Ces deux techniques ont été testées sur un réseau LiDAR et deux réseaux caméra-LiDAR entraînées sur l'ensemble d'entraînement de KITTI. Les trois réseaux adoptent une approche inspirée de CenterNet (Duan et al., 2019) pour la suppression des ancres. Le premier incrément a permis une amélioration des résultats visible directement sur l'ensemble de validation de KITTI. On a pu observer une performance accrue sur des nuages de points de faible résolution tout en conservant de bonnes capacités de détections sur des nuages de points bien plus denses. La seconde augmentation cause une dégradation des performances sur le jeu de validation de KITTI. Néanmoins, lorsque l'on a appliqué les réseaux « LiDAR » entraînés sur des nuages issus d'autres bases de données, nous avons pu constater de meilleurs scores de détections alors que ces données n'ont jamais été observées par les réseaux durant leur apprentissage. Concernant les réseaux caméra-LiDAR toutefois, nous avons observé de très faibles performances sur les jeux de données nuScenes et Pandaset. En effet, bien que nous ayons altéré les nuages de points au cours de l'entraînement, aucune modification n'a été réalisée sur les images RGB. Par conséquent, les réseaux ont sur-appris certains artefacts de la caméra ou certains phénomènes visibles uniquement dans la base KITTI. Enfin, pour valider notre approche, nous avons appliqué notre réseau n'utilisant que des nuages de points sur des données non annotées issues des enregistrements UTBM Robocar et Ford AV Dataset.

6.2 Perspectives

6.2.1 Intégration temporelle, suivi et prédiction

Les détections réalisées dans nos travaux sont élaborées échantillon par échantillon, sans mise en relation temporelle. D'un nuage de points au suivant, les résultats renvoyés par le système sont donc indépendants, ce qui peut provoquer des incohérences entre les sorties à l'instant $t-1$ et celles à l'instant t. Pour pallier ce problème, il est intéressant d'intégrer dans la chaîne de traitement l'information temporelle. Celle-ci est notamment représentée par les trajectoires et les paramètres dynamiques (vitesse longitudinale, vitesse de lacet, accélération longitudinale, etc.). L'approche la plus standard de suivi multiobjets (MOT pour *Multiple Object Tracking*) est le suivi par détection : un ensemble de détections (représentées par des boîtes englobantes identifiant les objets) est assemblé en pistes, chaque piste représente un seul et unique objet le long de l'enregistrement et est représenté par un identifiant unique. Selon (Ciaparrone et al., 2020), la majorité des algorithmes de MOT suivent partiellement ou complètement la décomposition suivante :

- une phase de détection : un algorithme de détection extrait à chaque instant les objets de la scène sous forme de boîtes englobantes ;
- une phase d'extraction de caractéristiques : un algorithme analyse le contenu des détections et des pistes existantes pour en extraire des caractéristiques ;
- une phase de calcul d'affinités : à partir des caractéristiques extraites, les distances entre détections et pistes sont calculées ;
- une phase d'association : les mesures de distance/similarités sont utilisées pour associer les détections aux pistes représentant les mêmes cibles.

À l'heure actuelle, un grand nombre de méthodes existent dans la littérature, principalement pour la détection 2D, l'un des aspects les plus étudiés étant la gestion des superpositions dues aux occultations pouvant provoquer des changements d'identifiants de piste. Dans le cadre de la détection 3D et BEV pour la conduite autonome, les occultations impactent surtout les quantités de points disponibles pour identifier un obstacle ce qui rend la détection plus difficile. Une superposition pour la tâche de suivi a plus de chances de représenter un accident entre deux véhicules. En marge du suivi qui va évaluer la position des obstacles entre les instants t_1 à t_n, il peut être intéressant de prédire les trajectoires futures sur un ou plusieurs horizons définis (par exemple sur 1 s) des obstacles détectés afin de mieux planifier la trajectoire de l'ego-véhicule.

6.2.2 Simulation et apprentissage multibases

L'annotation des données automobiles est difficile et chronophage. De plus, les capteurs LiDAR étant onéreux, peu de types de capteurs sont en général étudiés pour l'entraînement. La simulation de scènes est une piste de recherche intéressante dans la mesure où elle permettrait la synthèse de données et donc l'agrandissement de bases de données exploitables par des réseaux de neurones. Toutefois, l'un des aspects critiques de cette approche concerne le réalisme et/ou l'adéquation aux caractéristiques réelles des capteurs. Dans le cas de l'utilisation de caméras par exemple, le simulateur doit être capable de réaliser des rendus photoréalistes et de simuler physique de phénomènes optiques. Le cas inverse, le réseau entraîné ne sera pas directement transposable à des données réelles, car le domaine d'entraînement ne correspond pas au domaine d'application. En outre, comme expliqué plus tôt, la très grande majorité des méthodes n'est entraînée que sur un jeu de données précis couvrant un certain nombre de situations (géographiques, météorologiques, lumineuses, etc.). L'apprentissage séquentiel sur plusieurs bases de données peut être une piste d'exploration, notamment si ces nouvelles bases sont bien plus petites. Cela permettrait ainsi l'inclusion de nouvelles connaissances à coût réduit en se reposant sur les poids déjà appris. Cependant, cela peut également occasionner un

phénomène dit de *catastrophic interference* ou *catastrophic foregetting* signifiant l'écrasement des anciennes connaissances par les nouvelles.

6.2.3 Confiance, incertitude

La très grande majorité des réseaux de détection retournent un ensemble de boîtes englobantes et des scores d'appartenance aux classes. Toutefois, ces valeurs sont les résultats de calculs entre une entrée et les poids du réseau, au départ tirés aléatoirement. Des erreurs peuvent toujours survenir, aussi bien sur les scores de classification que sur les paramètres des boîtes. L'une des principales critiques faites aux réseaux de neurones traditionnels vient de leur incapacité à renvoyer des valeurs d'incertitude permettant de quantifier la fiabilité d'un résultat. Dans des domaines liés à la robotique, pour lesquels l'estimation du risque est nécessaire, cela constitue un important manquement, ralentissant ainsi l'implémentation des réseaux de neurones dans des contextes dits critiques. Une approche comme le *test time augmentation* peut permettre d'obtenir la dispersion d'un réseau « standard » à moindre coût. Cette approche consiste en une altération d'un même donnée d'entrée lors de l'inférence. Le réseau doit donc faire preuve de stabilité par rapport au bruit ajouté sur les différentes entrées. D'autres approches développent une nouvelle branche de recherche, les réseaux de neurones bayésiens, dans lesquels les poids de chaque unité sont remplacés par des distributions de probabilité (Jospin et al., 2020). Ces réseaux nécessitent de nouveaux frameworks et sont réputés lourds en termes de calcul d'inférence.

6.2.4 Interprétabilité et explicabilité

Un système de détection n'est pas infaillible et certaines erreurs peuvent s'avérer critiques dans des situations ou des domaines relevant de la sécurité. Par exemple si un obstacle proche de l'ego-véhicule et bien défini n'est pas relevé par le système de détection, il faut être capable expliquer la raison de cette erreur. Bien qu'il soit utile de réaliser des analyses supplémentaires telles que la répartition des erreurs dans l'espace, elles permettent uniquement de caractériser statistiquement le réseau utilisé. Elle ne permet pas de remonter aux causes des erreurs ou de justifier les résultats obtenus dans une disposition précise. Le plus souvent, les approches se focalisent sur la visualisation de variables intermédiaires. En classification, les *Class Activation Maps* (CAM) constituent l'une des approches les plus courantes permettant d'évaluer les régions de l'image des plus contributrices pour une classe précise. D'autres approches comme (Ben-Younes et al., 2020) tentent de retranscrire la justification d'un choix sous la forme d'une phrase compréhensible par l'homme.

Bibliographie

Agarwal, Siddharth, Ankit Vora, Gaurav Pandey, Wayne Williams, Helen Kourous et James McBride (2020). « Ford Multi-AV Seasonal Dataset ». In : *arXiv preprint arXiv :2003.07969*.

Ali, Waleed, Sherif Abdelkarim, Mahmoud Zidan, Mohamed Zahran et Ahmad El Sallab (2018). « Yolo3d : End-to-end real-time 3d oriented object bounding box detection from lidar point cloud ». In : *Proceedings of the European Conference on Computer Vision (ECCV)*, p. 0-0.

Badrinarayanan, Vijay, Alex Kendall et Roberto Cipolla (2017). « Segnet : A deep convolutional encoder-decoder architecture for image segmentation ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12, p. 2481-2495.

Bello, Saifullahi Aminu, Shangshu Yu, Cheng Wang, Jibril Muhmmad Adam et Jonathan Li (2020). « deep learning on 3D point clouds ». In : *Remote Sensing* 12.11, p. 1729.

Beltran, Jorge, Carlos Guindel, Francisco Miguel Moreno, Daniel Cruzado, Fernando Garcia et Arturo De La Escalera (2018). « BirdNet : a 3D Object Detection Framework from LiDAR information ». In : *21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, p. 3517-3523.

Ben-Younes, Hédi, Éloi Zablocki, Patrick Pérez et Matthieu Cord (2020). « Driving Behavior Explanation with Multi-level Fusion ». In : *arXiv preprint arXiv :2012.04983*.

Bottou, Léon (2010). « Large-scale machine learning with stochastic gradient descent ». In : *Proceedings of COMPSTAT'2010*. Springer, p. 177-186.

Brazil, Garrick et Xiaoming Liu (2019). « M3D-RPN : Monocular 3D Region Proposal Network for Object Detection ». In : *Proceeding of International Conference on Computer Vision*. Seoul, South Korea, p. 9287-9296.

Cadot, Julien (2018). *Voiture autonome, autopilote, assistance... : quelles différences entre les niveaux de conduite ?* url : https://www.numerama.com/tech/212551-voiture-autonome-autopilote-assistance-a-la-conduite-quelles-differences.html (visité le 25/02/2021).

Caesar, Holger, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan et Oscar Beijbom (2019). « nuScenes : A multimodal dataset for autonomous driving ». In : *arXiv preprint arXiv :1903.11027*.

Cai, Zhaowei, Quanfu Fan, Rogerio S Feris et Nuno Vasconcelos (2016). « A unified multi-scale deep convolutional neural network for fast object detection ». In : *European Conference on Computer Vision*. Springer, p. 354-370.

Canziani, Alfredo, Adam Paszke et Eugenio Culurciello (2016). « An analysis of deep neural network models for practical applications ». In : *arXiv preprint arXiv :1605.07678*.

— *Analysis of deep neural networks*. url : https://culurciello.medium.com/analysis-of-deep-neural-networks-dcf398e71aae (visité le 28/04/2021).

Carballo, Alexander, Abraham Monrroy, David Wong, Patiphon Narksri, Jacob Lambert, Yuki Kitsukawa, Eijiro Takeuchi, Shinpei Kato et Kazuya Takeda (2020a). « Characterization of Multiple 3D LiDARs for Localization and Mapping using Normal Distributions Transform ». In : *arXiv preprint arXiv :2004.01374*.

Carballo, Alexander, Jacob Lambert, Abraham Monrroy, David Wong, Patiphon Narksri, Yuki Kitsukawa, Eijiro Takeuchi, Shinpei Kato et Kazuya Takeda (2020b). « LIBRE : The Multiple 3D LiDAR Dataset ». In : *arXiv preprint arXiv :2003.06129*. (accepted for presentation at IV2020).

Chabot, Florian, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teuliere et Thierry Chateau (2017). « 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*, p. 2040-2049.

Chang, Jia-Ren et Yong-Sheng Chen (2018). « Pyramid stereo matching network ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 5410-5418.

Chardenon, Aude. *Navya Beti*. url : https://www.usine-digitale.fr/article/la-drome-va-tester-la-premiere-navette-autonome-en-milieu-rural.N1000324 (visité le 25/02/2021).

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy et Alan L Yuille (2017a). « DeepLab : Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4, p. 834-848.

Chen, Liang-Chieh, Yukun Zhu, George Papandreou, Florian Schroff et Hartwig Adam (2018). « Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation ». In : *ECCV*.

Chen, Qi, Lin Sun, Zhixin Wang, Kui Jia et Alan Yuille (2019a). « Object as Hotspots : An Anchor-Free 3D Object Detection Approach via Firing of Hotspots ». In : *arXiv preprint arXiv :1912.12791*.

Chen, Xiaozhi, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler et Raquel Urtasun (2016). « Monocular 3d object detection for autonomous driving ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 2147-2156.

Chen, Xiaozhi, Huimin Ma, Ji Wan, Bo Li et Tian Xia (2017b). « Multi-View 3D Object Detection Network for Autonomous Driving ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 1907-1915.

Chen, Yilun, Shu Liu, Xiaoyong Shen et Jiaya Jia (2019b). « Fast Point R-CNN ». In : *The IEEE International Conference on Computer Vision (ICCV)*.

Ciaparrone, Gioele, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri et Francisco Herrera (2020). « Deep learning in video multi-object tracking : A survey ». In : *Neurocomputing* 381, p. 61-88.

Dalal, Navneet et Bill Triggs (2005). « Histograms of oriented gradients for human detection ». In : *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. T. 1. Ieee, p. 886-893.

Dataset BigRedLiDAR. url : https://bigredlidar.github.io/ (visité le 02/03/2021).

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li et Li Fei-Fei (2009). « Imagenet : A large-scale hierarchical image database ». In : *2009 IEEE conference on computer vision and pattern recognition*. Ieee, p. 248-255.

Duan, Kaiwen, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang et Qi Tian (2019). « Centernet : Keypoint triplets for object detection ». In : *Proceedings of the IEEE International Conference on Computer Vision*, p. 6569-6578.

Felzenszwalb, Pedro F, Ross B Girshick, David McAllester et Deva Ramanan (2009). « Object detection with discriminatively trained part-based models ». In : *IEEE transactions on pattern analysis and machine intelligence* 32.9, p. 1627-1645.

Felzenszwalb, Pedro, David McAllester et Deva Ramanan (2008). « A discriminatively trained, multiscale, deformable part model ». In : *2008 IEEE conference on computer vision and pattern recognition*. IEEE, p. 1-8.

Fitzgibbon, Andrew, Maurizio Pilu et Robert B Fisher (1999). « Direct least square fitting of ellipses ». In : *IEEE Tr. on PAMI* 21.5, p. 476-480.

Fu, Huan, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich et Dacheng Tao (2018). « Deep ordinal regression network for monocular depth estimation ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 2002-2011.

Fukushima, Kunihiko (1975). « Cognitron : A self-organizing multilayered neural network ». In : *Biological cybernetics* 20.3, p. 121-136.

Gandhi, Rohith. *Object Detectors*. url : https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e (visité le 25/02/2021).

Ge, Runzhou, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang et Yuan Li (2020). « Afdet : Anchor free one stage 3d object detection ». In : *arXiv preprint arXiv :2006.12671*.

Geiger, Andreas, Philip Lenz, Christoph Stiller et Raquel Urtasun (2013). « Vision meets Robotics : The KITTI Dataset ». In : *International Journal of Robotics Research (IJRR)*.

Girshick, Ross (2015). « Fast r-cnn ». In : *Proceedings of the IEEE international conference on computer vision*, p. 1440-1448.

Girshick, Ross, Jeff Donahue, Trevor Darrell et Jitendra Malik (2014). « Rich feature hierarchies for accurate object detection and semantic segmentation ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 580-587.

Graham, Ben (2015). « Sparse 3D convolutional neural networks ». In : *arXiv preprint arXiv :1505.02890*.

Graham, Benjamin (2014). « Spatially-sparse convolutional neural networks ». In : *arXiv preprint arXiv :1409.6070*.

Graham, Benjamin et Laurens van der Maaten (2017). « Submanifold Sparse Convolutional Networks ». In : *arXiv preprint arXiv :1706.01307*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren et Jian Sun (2016). « Deep Residual Learning for Image Recognition ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 770-778.

— (2015). « Spatial pyramid pooling in deep convolutional networks for visual recognition ». In : *IEEE transactions on pattern analysis and machine intelligence* 37.9, p. 1904-1916.

Hendrycks, Dan et Kevin Gimpel (2016). « Gaussian error linear units (gelus) ». In : *arXiv preprint arXiv :1606.08415*.

Huang, Gao, Zhuang Liu, Laurens Van Der Maaten et Kilian Q Weinberger (2017). « Densely connected convolutional networks ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 4700-4708.

Huang, Xinyu, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin et Ruigang Yang (2018). « The apolloscape dataset for autonomous driving ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, p. 954-960.

Huang, Xinyu, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng et Ruigang Yang (2019). « The apolloscape open dataset for autonomous driving and its application ». In : *IEEE transactions on pattern analysis and machine intelligence* 42.10, p. 2702-2719.

Ioffe, Sergey et Christian Szegedy (2015). « Batch normalization : Accelerating deep network training by reducing internal covariate shift ». In : *International conference on machine learning*. PMLR, p. 448-456.

Jadon, Shruti. *Activation functions - Medium*. url : https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092 (visité le 25/02/2021).

Jiang, Peng et Srikanth Saripalli (2020). « LiDARNet : A Boundary-Aware Domain Adaptation Model for LiDAR Point Cloud Semantic Segmentation ». In : *arXiv preprint arXiv :2003.01174*.

Jospin, Laurent Valentin, Wray Buntine, Farid Boussaid, Hamid Laga et Mohammed Bennamoun (2020). « Hands-on Bayesian Neural Networks–a Tutorial for Deep Learning Users ». In : *arXiv preprint arXiv :2007.06823*.

Kasmi, Abderrahim, Johann Laconte, Romuald Aufrère, Ruddy Theodose, Dieumet Denis et Roland Chapuis (2020). « An Information Driven Approach For Ego-Lane Detection Using Lidar And OpenStreetMap ». In : *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, p. 522-528.

Kingma, Diederik P et Jimmy Ba (2014). « Adam : A method for stochastic optimization ». In : *arXiv preprint arXiv :1412.6980*.

Kirillov, Alexander, Kaiming He, Ross Girshick, Carsten Rother et Piotr Dollár (2019). « Panoptic segmentation ». In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 9404-9413.

Krizhevsky, Alex, Ilya Sutskever et Geoffrey E Hinton (2012). « Imagenet classification with deep convolutional neural networks ». In : *Advances in neural information processing systems* 25, p. 1097-1105.

Ku, Jason, Melissa Mozifian, Jungwook Lee, Ali Harakeh et Steven L Waslander (2018). « Joint 3D Proposal Generation and Object Detection from View Aggregation ». In : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, p. 1-8.

Kuang, Hongwu, Bei Wang, Jianping An, Ming Zhang et Zehan Zhang (2020). « Voxel-FPN : Multi-Scale Voxel Feature Aggregation for 3D Object Detection from LiDAR Point Clouds ». In : *Sensors* 20.3. issn : 1424-8220. doi : 10.3390/s20030704. url : https://www.mdpi.com/1424-8220/20/3/704.

Lang, Alex H., Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang et Oscar Beijbom (2019). « PointPillars : Fast Encoders for Object Detection From Point Clouds ». In : *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Langer, Ferdinand, Andres Milioto, Alexandre Haag, Jens Behley et Cyrill Stachniss (2020). « Domain transfer for semantic segmentation of LiDAR data using deep neural networks ». In : *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, p. 8263-8270.

Law, Hei et Jia Deng (2018). « Cornernet : Detecting objects as paired keypoints ». In : *Proceedings of the European Conference on Computer Vision (ECCV)*, p. 734-750.

LeCun, Yann et al. (1989). « Generalization and network design strategies ». In : *Connectionism in perspective* 19, p. 143-155.

LeCun, Yann, Léon Bottou, Yoshua Bengio et Patrick Haffner (1998). « Gradient-based learning applied to document recognition ». In : *Proceedings of the IEEE* 86.11, p. 2278-2324.

LeCun, Yann, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard et Lawrence D Jackel (1990). « Handwritten digit recognition with a back-propagation network ». In : *Advances in neural information processing systems*, p. 396-404.

Li, Fei-Fei and Johnson Justin and Yeung Serena. *Cours 11 de Li, Johnson et Yeung (Université de Stanford)*. url : http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf (visité le 02/03/2021).

Li, Peixuan, Huaici Zhao, Pengfei Liu et Feidao Cao (2020). « RTM3D : Real-time Monocular 3D Detection from Object Keypoints for Autonomous Driving ». In : *arXiv preprint arXiv :2001.03343*.

Liang, Ming, Bin Yang, Shenlong Wang et Raquel Urtasun (2018). « Deep Continuous Fusion for Multi-Sensor 3D Object Detection ». In : *Proceedings of the European Conference on Computer Vision (ECCV)*, p. 641-656.

Liang, Ming, Bin Yang, Yun Chen, Rui Hu et Raquel Urtasun (2019). « Multi-task multi-sensor fusion for 3d object detection ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 7345-7353.

Liang, Zhidong, Ming Zhang, Zehan Zhang, Xian Zhao et Shiliang Pu (2020). « RangeRCNN : Towards Fast and Accurate 3D Object Detection with Range Image Representation ». In : *arXiv preprint arXiv :2009.00206*.

Liao, Yiyi, Jun Xie et Andreas Geiger (2021). « KITTI-360 : A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D ». In : *arXiv.org* 2109.13410.

Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan et Serge Belongie (2017a). « Feature Pyramid Networks for Object Detection ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 2117-2125.

Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He et Piotr Dollár (2017b). « Focal loss for dense object detection ». In : *Proceedings of the IEEE International Conference on Computer Vision*, p. 2980-2988.

Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu et Alexander C Berg (2016). « Ssd : Single shot multibox detector ». In : *European conference on computer vision*. Springer, p. 21-37.

Liu, Zechen, Zizhang Wu et Roland Tóth (2020). « SMOKE : single-stage monocular 3d object detection via keypoint estimation ». In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, p. 996-997.

Long, Jonathan, Evan Shelhamer et Trevor Darrell (2015). « Fully Convolutional Networks for Semantic Segmentation ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 3431-3440.

Ma, Wenping, Qifan Yang, Yue Wu, Wei Zhao et Xiangrong Zhang (2019). « Double-branch multi-attention mechanism network for hyperspectral image classification ». In : *Remote Sensing* 11.11, p. 1307.

Maas, Andrew L, Awni Y Hannun et Andrew Y Ng (2013). « Rectifier nonlinearities improve neural network acoustic models ». In : *Proc. icml*. T. 30. 1. Citeseer, p. 3.

Manhardt, Fabian, Wadim Kehl et Adrien Gaidon (2019). « Roi-10d : Monocular lifting of 2d detection to 6d pose and metric shape ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 2069-2078.

Mousavian, Arsalan, Dragomir Anguelov, John Flynn et Jana Kosecka (2017). « 3d bounding box estimation using deep learning and geometry ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 7074-7082.

Outil d'annotation Supervisely. url : https://supervise.ly/lidar-3d-cloud/ (visité le 25/02/2021).

Pandaset by Hesai and Scale. url : https://pandaset.org/ (visité le 13/07/2021).

Pang, Su, Daniel Morris et Hayder Radha (2020). « CLOCs : Camera-LiDAR Object Candidates Fusion for 3D Object Detection ». In : *arXiv preprint arXiv :2009.00784*.

Patole, Sujeet Milind, Murat Torlak, Dan Wang et Murtaza Ali (2017). « Automotive radars : A review of signal processing techniques ». In : *IEEE Signal Processing Magazine* 34.2, p. 22-35.

Pendleton, Scott Drew, Hans Andersen, Xinxin Du, Xiaotong Shen, Malika Meghjani, You Hong Eng, Daniela Rus et Marcelo H Ang (2017). « Perception, planning, control, and coordination for autonomous vehicles ». In : *Machines* 5.1, p. 6.

Perceptron multicouche - Wikipédia. url : https://fr.wikipedia.org/wiki/Perceptron_multicouche (visité le 25/02/2021).

Piewak, Florian, Peter Pinggera et Marius Zöllner (2019). « Analyzing the Cross-Sensor Portability of Neural Network Architectures for LiDAR-based Semantic Labeling ». In : *IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, p. 3419-3426.

Produits Ouster. url : https://ouster.com/products (visité le 25/02/2021).

Produits Velodyne. url : https://velodynelidar.com/products/ (visité le 25/02/2021).

Pröve, Paul-Louis. *Convolutions in Deep Learning*. url : https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d (visité le 25/02/2021).

Qi, Charles R, Wei Liu, Chenxia Wu, Hao Su et Leonidas J Guibas (2018). « Frustum PointNets for 3D Object Detection from RGB-D Data ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 918-927.

Qi, Charles R, Hao Su, Kaichun Mo et Leonidas J Guibas (2017a). « PointNet : Deep Learning on Point Sets for 3D Classification and Segmentation ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 652-660.

Qi, Charles Ruizhongtai, Li Yi, Hao Su et Leonidas J Guibas (2017b). « PointNet++ : Deep Hierarchical Feature Learning on Point Sets in a Metric Space ». In : *Advances in Neural Information Processing Systems*, p. 5099-5108.

Qian, Ning (1999). « On the momentum term in gradient descent learning algorithms ». In : *Neural networks* 12.1, p. 145-151.

Redmon, Joseph et Ali Farhadi (2017). « YOLO9000 : better, faster, stronger ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 7263-7271.

— (2018). « YOLOv3 : An Incremental Improvement ». In : *arXiv preprint arXiv :1804.02767*.

Redmon, Joseph, Santosh Divvala, Ross Girshick et Ali Farhadi (2016). « You Only Look Once : Unified, Real-Time Object Detection ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 779-788.

Ren, S., K. He, R. Girshick et J. Sun (2017). « Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6, p. 1137-1149.

Ren, Shaoqing, Kaiming He, Ross Girshick et Jian Sun (2015). « Faster r-cnn : Towards real-time object detection with region proposal networks ». In : *arXiv preprint arXiv :1506.01497*.

Roddick, Thomas, Alex Kendall et Roberto Cipolla (2018). « Orthographic Feature Transform for Monocular 3D Object Detection ». In : *arXiv preprint arXiv :1811.08188*.

Ronneberger, Olaf, Philipp Fischer et Thomas Brox (2015). « U-Net : Convolutional Networks for Biomedical Image Segmentation ». In : *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, p. 234-241.

Rosebrock, Adrian. *Intersection over union (IOU) for object detection*. url : https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (visité le 25/02/2021).

Rosenblatt, Frank (1958). « The perceptron : a probabilistic model for information storage and organization in the brain. » In : *Psychological review* 65.6, p. 386.

Roynard, Xavier, Jean-Emmanuel Deschaud et François Goulette (2018). « Paris-Lille-3D : A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification ». In : *The International Journal of Robotics Research* 37.6, p. 545-557. doi : 10.1177/0278364918767506. eprint : https://doi.org/10.1177/0278364918767506. url : https://doi.org/10.1177/0278364918767506.

Saleh, Khaled, Ahmed Abobakr, Mohammed Attia, Julie Iskander, Darius Nahavandi, Mohammed Hossny et Saeid Nahvandi (2019). « Domain adaptation for vehicle detection from bird's eye view lidar point cloud data ». In : *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, p. 0-0.

Sallab, Ahmad El, Ibrahim Sobh, Mohamed Zahran et Nader Essam (2019). « LiDAR sensor modeling and data augmentation with GANs for autonomous driving ». In : *arXiv preprint arXiv :1905.07290*.

Shah, Anish, Eashan Kadam, Hena Shah, Sameer Shinde et Sandip Shingade (2016). « Deep residual networks with exponential linear unit ». In : *Proceedings of the Third International Symposium on Computer Vision and the Internet*, p. 59-65.

Shan, Tixiao, Jinkun Wang, Fanfei Chen, Paul Szenher et Brendan Englot (2020). « Simulation-based lidar super-resolution for ground vehicles ». In : *Robotics and Autonomous Systems* 134, p. 103647.

Shi, Shaoshuai, Xiaogang Wang et Hongsheng Li (2019). « PointRCNN : 3D Object Proposal Generation and Detection From Point Cloud ». In : *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shi, Shaoshuai, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang et Hongsheng Li (2020). « PV-RCNN : Point-Voxel Feature Set Abstraction for 3D Object Detection ». In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 10529-10538.

Shi, Shaoshuai, Zhe Wang, Xiaogang Wang et Hongsheng Li (2019). « Part-A^2 Net : 3D Part-Aware and Aggregation Neural Network for Object Detection from Point Cloud ». In : *arXiv preprint arXiv :1907.03670*.

Shrivastava, Shubham et Punarjay Chakravarty (2020). « CubifAE-3D : Monocular Camera Space Cubification on Autonomous Vehicles for Auto-Encoder based 3D Object Detection ». In : *arXiv preprint arXiv :2006.04080*.

Simon, Martin, Stefan Milz, Karl Amende et Horst-Michael Gross (2019). « Complex-YOLO : An Euler-Region-Proposal for Real-Time 3D Object Detection on Point Clouds ». In : *Computer Vision – ECCV 2018 Workshops*. Sous la dir. de Laura Leal-

Taixé et Stefan Roth. Cham : Springer International Publishing, p. 197-209. isbn : 978-3-030-11009-3.

Simonelli, Andrea, Samuel Rota Bulo, Lorenzo Porzi, Manuel Lopez-Antequera et Peter Kontschieder (2019). « Disentangling Monocular 3D Object Detection ». In : *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Simonyan, Karen et Andrew Zisserman (2014). « Very deep convolutional networks for large-scale image recognition ». In : *arXiv preprint arXiv :1409.1556*.

Sindagi, Vishwanath A, Yin Zhou et Oncel Tuzel (2019). « MVX-Net : Multimodal voxelnet for 3D object detection ». In : *International Conference on Robotics and Automation (ICRA)*. IEEE, p. 7276-7282.

Site officiel de ONNX. url : https://onnx.ai/ (visité le 13/07/2021).

Sun, Pei, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine et al. (2020). « Scalability in perception for autonomous driving : Waymo open dataset ». In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 2446-2454.

Sécurité Routière 2019. url : https://www.onisr.securite-routiere.gouv.fr/etat-de-l-insecurite-routiere/bilans-annuels-de-la-securite-routiere/bilan-2019-de-la-securite-routiere (visité le 25/02/2021).

Thomas, Hugues, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette et Leonidas J Guibas (2019). « Kpconv : Flexible and deformable convolution for point clouds ». In : *Proceedings of the IEEE/CVF International Conference on Computer Vision*, p. 6411-6420.

Tornado Mobility. url : https://www.tornado-mobility.com/index.php/en/home-2/ (visité le 25/02/2021).

Triess, Larissa T, David Peter, Christoph B Rist, Markus Enzweiler et J Marius Zöllner (2019). « CNN-based synthesis of realistic high-resolution LiDAR data ». In : *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, p. 1512-1519.

Van de Sande, Koen EA, Jasper RR Uijlings, Theo Gevers et Arnold WM Smeulders (2011). « Segmentation as selective search for object recognition ». In : *2011 International Conference on Computer Vision*. IEEE, p. 1879-1886.

Viola, Paul et Michael Jones (2001). « Rapid object detection using a boosted cascade of simple features ». In : *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. T. 1. IEEE, p. I-I.

Vora, Sourabh, Alex H. Lang, Bassam Helou et Oscar Beijbom (2020). « PointPainting : Sequential Fusion for 3D Object Detection ». In : *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, Dingkang, Connor Watkins et Huikai Xie (2020). « MEMS Mirrors for LiDAR : A review ». In : *Micromachines* 11.5, p. 456.

Wang, Jingdong, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang et al. (2020). « Deep high-resolution representation learning for visual recognition ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, Mei et Weihong Deng (2018). « Deep visual domain adaptation : A survey ». In : *Neurocomputing* 312, p. 135-153.

Wang, Yan, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell et Kilian Q Weinberger (2019a). « Pseudo-lidar from visual depth estimation : Bridging the gap in 3d object detection for autonomous driving ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 8445-8453.

Wang, Yue, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein et Justin M Solomon (2019b). « Dynamic graph cnn for learning on point clouds ». In : *Acm Transactions On Graphics (tog)* 38.5, p. 1-12.

Wang, Zhixin et Kui Jia (2019). « Frustum ConvNet : Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection ». In : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, p. 1742-1749.

Wu, Bichen, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue et Kurt Keutzer (2019). « Squeezesegv2 : Improved model structure and unsupervised domain adaptation for road-object segmentation from a LiDAR point cloud ». In : *International Conference on Robotics and Automation (ICRA)*. IEEE, p. 4376-4382.

Wu, Yuxin et Kaiming He (2018). « Group normalization ». In : *Proceedings of the European conference on computer vision (ECCV)*, p. 3-19.

Xiang, Yu, Roozbeh Mottaghi et Silvio Savarese (2014). « Beyond PASCAL : A Benchmark for 3D Object Detection in the Wild ». In : *IEEE Winter Conference on Applications of Computer Vision (WACV)*.

Xie, Jun, Martin Kiefel, Ming-Ting Sun et Andreas Geiger (2016). « Semantic Instance Annotation of Street Scenes by 3D to 2D Label Transfer ». In : *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xu, Danfei, Dragomir Anguelov et Ashesh Jain (2018). « PointFusion : Deep Sensor Fusion for 3D Bounding Box Estimation ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 244-253.

Xu, Yifan, Tianqi Fan, Mingye Xu, Long Zeng et Yu Qiao (2018). « Spidercnn : Deep learning on point sets with parameterized convolutional filters ». In : *Proceedings of the European Conference on Computer Vision (ECCV)*, p. 87-102.

Yan, Yan, Yuxing Mao et Bo Li (2018). « SECOND : Sparsely Embedded Convolutional Detection ». In : *Sensors* 18.10. issn : 1424-8220. doi : 10.3390/s18103337. url : https://www.mdpi.com/1424-8220/18/10/3337.

Yan, Zhi, Li Sun, Tomas Krajnik et Yassine Ruichek (2020). « EU Long-term Dataset with Multiple Sensors for Autonomous Driving ». In : *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Yang, Zetong, Yanan Sun, Shu Liu, Xiaoyong Shen et Jiaya Jia (2018). « IPOD : Intensive Point-based Object Detector for Point Cloud ». In : *arXiv preprint arXiv :1812.05276*.

Yi, Li, Boqing Gong et Thomas Funkhouser (2021). « Complete & label : A domain adaptation approach to semantic segmentation of LiDAR point clouds ». In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 15363-15373.

Yin, Tianwei, Xingyi Zhou et Philipp Krähenbühl (2020). « Center-based 3D Object Detection and Tracking ». In : *arXiv :2006.11275*.

Yohanandan, Shivy. *mean Average Precision*. url : https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2 (visité le 25/02/2021).

You, Yurong, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell et Kilian Q Weinberger (2019). « Pseudo-lidar++ : Accurate depth for 3d object detection in autonomous driving ». In : *arXiv preprint arXiv :1906.06310*.

Zhao, Hengshuang, Jianping Shi, Xiaojuan Qi, Xiaogang Wang et Jiaya Jia (2017). « Pyramid scene parsing network ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 2881-2890.

Zhou, Xingyi, Dequan Wang et Philipp Krähenbühl (2019). « Objects as points ». In : *arXiv preprint arXiv :1904.07850*.

Zhou, Xingyi, Jiacheng Zhuo et Philipp Krahenbuhl (2019). « Bottom-up object detection by grouping extreme and center points ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 850-859.

Zhou, Yin et Oncel Tuzel (2018). « Voxelnet : End-to-end learning for point cloud based 3d object detection ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 4490-4499.

Zhu, Jun-Yan, Taesung Park, Phillip Isola et Alexei A Efros (2017). « Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks ». In : *Computer Vision (ICCV), 2017 IEEE International Conference on*.

Zou, Zhengxia, Zhenwei Shi, Yuhong Guo et Jieping Ye (2019). « Object detection in 20 years : A survey ». In : *arXiv preprint arXiv :1905.05055*.

nuScenes. url : https://www.nuscenes.org/nuscenes#data-collection (visité le 13/07/2021).