Estimation des Disparités par Adaptation de Domaine pour la Détection d'Objets 3D à partir d'Images stéréoscopiques

par

Lucas HUYGHUES-BEAUFOND

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

MONTRÉAL, LE 21 FÉVRIER 2023

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Marco Pedersoli, directeur de mémoire
génie des systèmes à l'École de technologie supérieure

M. José Dolz, président du jury
génie logiciel et des TI à l'École de technologie supérieure

M. Matthew Toews, membre du jury
génie des systèmes à l'École de technologie supérieure

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

LE 8 FÉVRIER 2023

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Ce mémoire a pu être réalisé grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma gratitude.

Je tiens à exprimer toute ma reconnaissance à mon directeur de recherche Marco Pedersoli, professeur à l'ETS, pour m'avoir encadré, conseillé tout au long de ma recherche et pour sa patience et disponibilité durant la relecture et correction du mémoire.

J'adresse mes sincères remerciements à toutes l'équipe du projet *dista* de l'école André-Laurendeau (Montréal) qui par leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions. Je remercie en particulier David Beaulieu, responsable du projet dista, pour m'avoir accordé sa confiance et une large indépendance dans l'exécution de ma recherche. Un grand merci également à Christian Thériault, enseignant-chercheur en IA sur le projet, pour m'avoir guidé et supervisé au début de ma recherche, en m'apportant les outils techniques et méthodologiques indispensables à la conduite de mon mémoire.

Enfin, je remercie mes très chers parents pour leurs aides et encouragements tout au long de ma scolarité.

Estimation des Disparités par Adaptation de Domaine pour la Détection d'Objets 3D à partir d'Images stéréoscopiques

Lucas HUYGHUES-BEAUFOND

RÉSUMÉ

Les enjeux écologiques ont aujourd'hui une place importante dans la politique mondiale. Pour lutter contre le réchauffement climatique, de plus en plus de personnes décident d'utiliser le vélo comme alternative à la voiture, qui est l'un des moyens de transport les plus polluants. Cependant, la cohabitation vélo-voiture est difficile et provoque chaque année de nombreux accidents graves, ce qui nécessite la mise en place de solutions efficaces assurant la sécurité des cyclistes. C'est dans ce contexte de sécurité routière que nous nous sommes intéressés à développer un système employant la stéréovision et une méthode de détection 3D en apprentissage profond, dans le but d'estimer la distance de sécurité entre un vélo et un véhicule lors d'un dépassement.
Cependant, les méthodes actuelles de détection 3D par images stéréoscopiques ont des performances très limitées, dont les erreurs proviennent principalement de la méthode de disparité permettant de générer la représentation 3D de la scène avant d'effectuer la détection. En effet, obtenir des cartes de disparité réelles densément annotées est une tache fastidieuse, et la faible quantité de données réelles disponibles ne permet pas d'entraîner efficacement ces méthodes.
Dans ce mémoire, nous allons nous intéresser à la conception d'un modèle de disparité rapide, inspiré du réseau 2D DispNetC, dont on cherchera à améliorer sa précision et ses capacités de généralisation. De plus, dans le but de résoudre le problème de manque de données réelles, nous allons adapter le modèle au domaine d'une manière non-supervisée par le principe d'apprentissage adverse, permettant de réduire l'écart entre les domaines synthétique (annoté) et réel (sans annotation). Un CycleGAN va translater les données d'apprentissage synthétiques vers le domaine réel, et un discriminateur de caractéristiques va rendre invariant au domaine les représentations internes au réseau. La stratégie d'apprentissage proposée permettra d'augmenter la robustesse du modèle face au changement de domaine, et d'améliorer ses performances sur le domaine réel dans le cas non-supervisé. Le modèle final sera associé à une méthode de détection 3D par stéréo pour mesurer sa capacité à produire l'information 3D pour la détection. On l'évaluera sur la base KITTI pour la détection 3D et on validera les résultats des expériences en les comparant avec les méthodes de la littérature.

Mots-clés: détection 3D, stéréovision, disparité, adaptation de domaine, apprentissage adverse

Disparity Estimation and Domain Adaptation for Stereo-Based 3D Object Detection

Lucas HUYGHUES-BEAUFOND

ABSTRACT

Today's world is confronted to several devastating consequences due to pollution. Increasingly, people want to adopt an eco-friendly behavior and find a substitution to the car is the first step to reduce pollution. And the bycicle is the means of transport the most voted in. However, the coexistence between bicycles and cars is difficult and causes many serious accidents every year, which requires effective solutions to ensure the safety of cyclists. In this context of road safety, we have developed a system using stereovision and a 3D detection method in deep learning, with the aim of estimating the safe distance between a bike and a vehicle when passing.
However, the actuals stereo-image based 3D detection methods have limited performance, mostly due to the disparity estimation methods used to produce 3D representation of the scene before detection. Indeed, obtaining real disparity map densly annoted is a tedious task, and the few dataset available does not allow an efficient training of the model.
In this thesis, we will focus on the design of a fast disparity model, inpired by the 2D network DispNetC, whom we will seek to improve its accuracy and generalization. Moreover, in order to solve the problem of lack of real labeled data, the model will be adapted to the real domain in a unsupervied manner by the adversarial learning principle, allowing to reduce the gap between synthetic (annotated) and real (without annotation) domains. The CycleGAN network will translate synthetic learning data into the real domain, and a feature discriminator will make the internal representations of the network invariant to the domain. The proposed learning strategy will increase the robustness of the model facing domain shift, and improve its performance on the real domain in the unsupervised case. The final model will be combined with a image-stereo based 3D detection method to measure its ability to produce 3D informations for detection. It will be evaluated on the KITTI dataset for 3D detection and the results of the experiments will be validated by comparison with literature methods.

Keywords: detection 3D, disparity, domain adaptation, adversarial learning, stereovision

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AP	Average Precision
BEV	Bird's-eye-view
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSR	Code de la Sécurité Routière
DO3D	Détection d'Objet 3D
ETS	École de Technologie Supérieure
FP	False Positive
FPS	Frame per Second
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HOG	Histogram of Gradients
IA	Intelligence Artificielle
IoU	Intersection over Union
LBP	Local Binary Patterns
mAP	mean Average Precision
MB	MIDDLEBURY
NMS	Non-Max-Suppression
P	Précision
PSV	Plan-Sweep Volume
R	Rappel (Recall)
RoIs	régions d'intérêt
RPN	Réseau de proposition de régions

SIFT	Scale Invariant Feature Transform
SPP	Spatial Pyramid Pooling
TP	True Positive

LISTE DES SYMBOLES ET UNITÉS DE MESURE

b	Boite encadrante estimée
B	Baseline caméra stéréoscopique
f	distance focale
h	distance entre le système de caméra et la scène
$\hat{b}$	Boite encadrante réelle (label)
ε	Seuil de IoU
ε_d	Seuil D1
Ghz	Gigahertz
m	mètre
Mhz	Megahertz
ms	milliseconde
s	seconde
TFLOP	Teraflop
y_D	Carte de disparité estimée
$\hat{y_D}$	Carte de disparité réelle (label)
z	Carte de profondeur

INTRODUCTION

0.1 Motivation

L'année 2021 a comptabilisé un total de 27 888 personnes accidentées sur les routes au Québec (Canada), dont 1 574 ont été tuées ou gravement blessées. On observe ainsi une augmentation d'environ 4,9% d'accidents graves et mortelles par rapport à l'année précédente. Parmi ces décès, 54,8% correspondent à des occupants d'une voiture ou d'un camion, 19,3% étaient des occupants d'une moto et 4,6% d'un vélo. Ce qui représente une hausse de 27,4% pour les motocyclistes et de 56,9% chez les cyclistes par rapport à la moyenne entre 2016 à 2020 (chiffres issus de la SAAQ (2021)).

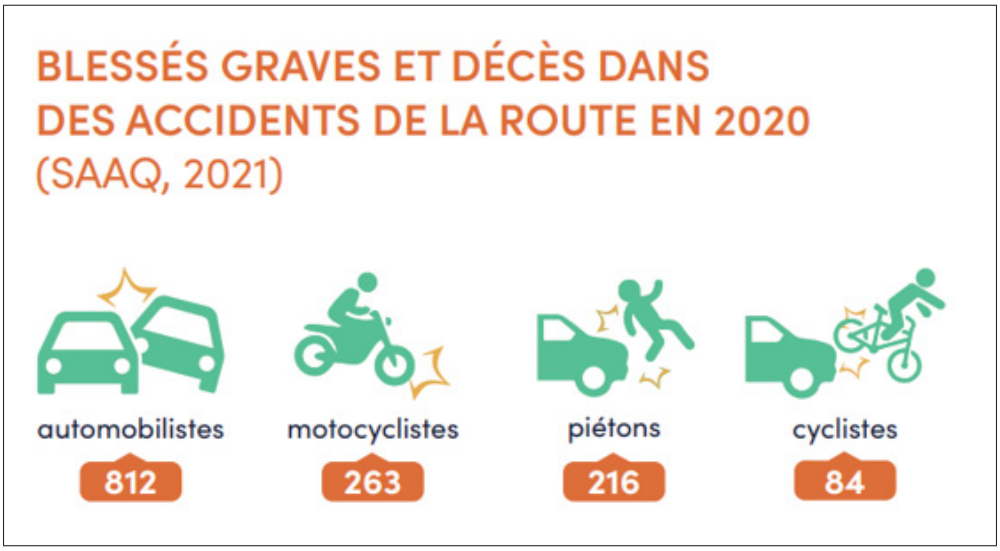

FIGURE 0.1 Nombre de blessés grave et décès dans des accidents de la route au Québec, Canada, en 2020 selon le moyen de transport
Tirée de Vélo Québec (2021)

La majorité des décès chez les cyclistes sont dus à des collisions avec des véhicules motorisés, et surviennent en milieu urbain. Ainsi au Canada, on estime une moyenne de 74 accidents mortels chaque année, dont 56% se déroulent en ville et 73% impliquent un véhicule motorisé

(Statistique Canada (2019)). En effet, les zones urbaines sont plus touchées par des collisions concernant les cyclistes du fait de la forte densité du trafic routier, de l'augmentation du nombre de cycliste (au moins 300 000 de plus qu'en 2015 au Québec, selon Vélo Québec (2021)) mais aussi à cause du non respect des règles de sécurité routière (qui concerne un tiers des accidents mortels au Canada, selon Statistique Canada (2019)).

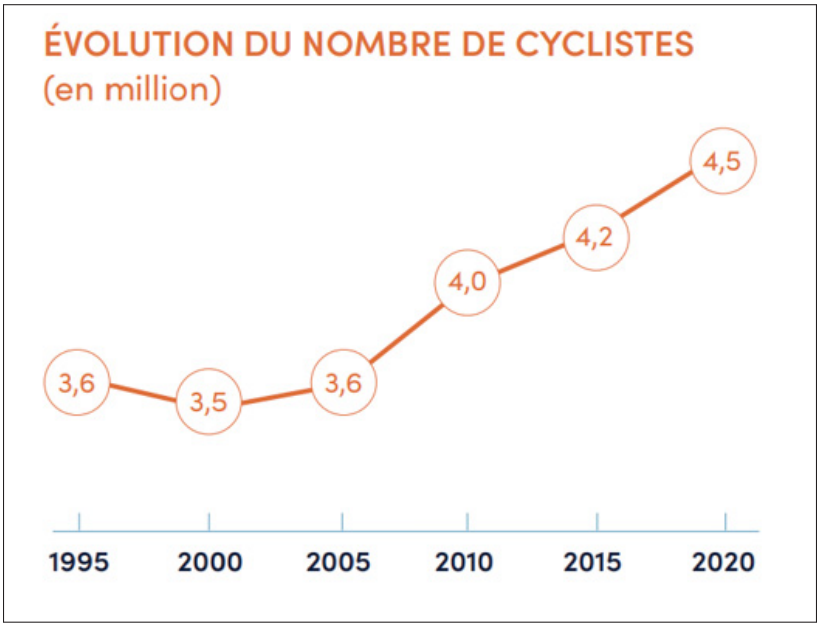

FIGURE 0.2 Évolution du nombre de cycliste (en million) au Québec (Canada) entre 1995 et 2020
Tirée de Vélo Québec (2021)

Ces statistiques poussent les villes à mettre en place des mesures pour assurer la sécurité des cyclistes par exemple en développant des infrastructures cyclables (pistes, bandes, chaussées désignées...), ou en modérant la circulation automobile (radars de vitesse, dos d'âne allongés, avancées de trottoir, ralentisseurs...) (d'après Vélo Québec (2021)).

Cependant, certaines infractions au Code de la Sécurité Routière (CSR) sont difficiles à contrôler sans avoir un dispositif adapté à la situation. Le dépassement d'une bicyclette par un véhicule motorisé sans le respect de la distance de sécurité est un problème récurrent dans les zones

où on ne peut implanter de voie cyclable (voir image 0.3). Le CSR exige une distance de 1m sur les routes où la vitesse affichée est de 50 km/h et moins ; et de 1,5 m sur les routes où la vitesse affichée est de plus de 50 km/h. Mais ces distances sont très peu respectées : on observe une augmentation de 92,3% d'infraction à cette règle entre 2015 et 2019 au Québec (selon Tardif (2021)). A noter que ce pourcentage représente seulement les cas observés, et donc de nombreuses infractions ne sont pas comptabilisées.

FIGURE 0.3 Illustration dépassement d'un vélo par une voiture sur une même voie en zone urbaine
Tirée de SPVM (2022)

Il peut être difficile pour un conducteur d'estimer à l'oeil la bonne distance de sécurité lors d'un dépassement d'un cycliste. De même, il difficile pour les forces de l'ordre, plus que toute autre infraction, d'appliquer la loi avec certitude. En effet, ils doivent en général se baser sur leur jugement, et il faut que l'infraction constatée ne soit pas contestable à la Cour (Radio-Canada (2016)). En 2019, il a été testé à Montréal (Québec) un nouveau radar mesurant avec précision la distance latéral d'un véhicule dépassant un policier à vélo (Maisonneuve & Lemieux (2018)). Ce dispositif, développé aux États-Unis, est coûteux (1500 $US) et doit être installé sur le guidon du vélo du policier (voir image 0.4). Bien que son but est de contrôler et prévenir les mauvais

dépassement, on peu voir que les deux contraintes énoncées rendent difficile son utilisation à grande échelle.

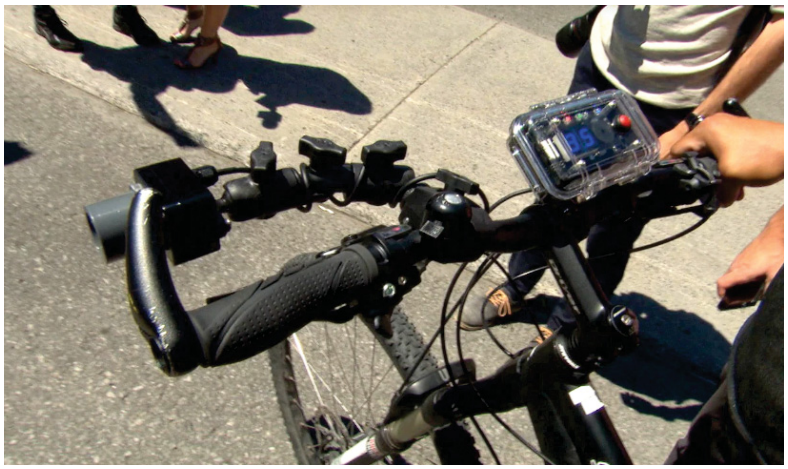

FIGURE 0.4 Image d'un radar pour mesurer la distance lors d'un dépassement d'un vélo. Le radar, à émission laser, est installé sur le guidon du vélo du policier
Tirée de Maisonneuve & Lemieux (2018)

C'est dans ce contexte qu'un groupe de chercheurs de l'école André Laurendeau à Montréal ont cherché à développer un système plus efficace, nommé "Dista", permettant de mesurer avec précision la distance de dépassement d'un véhicule avec un vélo, et d'avertir l'automobiliste lors d'une infraction, le tout en temps réel sans l'intervention de l'homme, à l'aide de capteurs optiques. Ce système doit pouvoir être installé facilement dans une zone urbaine très fréquentées (au niveau d'un feu, d'une intersection), et être à une hauteur permettant d'avoir une vue sur l'ensemble du trafic routier (poteau de signalisation).

Le capteur employé doit être léger, peu coûteux et avoir des données facilement exploitable pour permettre son déploiement dans plusieurs zones d'une ville. Mais il doit aussi pouvoir capturer les informations 3D de la scène : en effet, l'orientation et les dimensions de deux objets font varier la distance latéral les séparant, ce qui est difficilement observable en 2D. C'est pourquoi les chercheurs ont décidé d'utiliser un système stéréoscopique pour le flux de données.

L'analyse automatique de données issues d'un capteur et l'extraction des informations indispensables au problème traité, est une tache qui demande l'utilisation d'un algorithme complexe. L'émergence de l'intelligence artificielle (IA) ces dernières années, a permis de développer des méthodes répondant à des problèmes en vision par ordinateur très complexes. En particulier, un sous-domaine appelé *Deep Learning* (ou Apprentissage Profond) a permis de construire des modèles puissants, les réseaux de neurones convolutionnels (CNN), apprenant à partir d'un grand nombre de données des informations complexes et sémantiques, pour résoudre tous types de tache dans un large panel de domaine d'application.

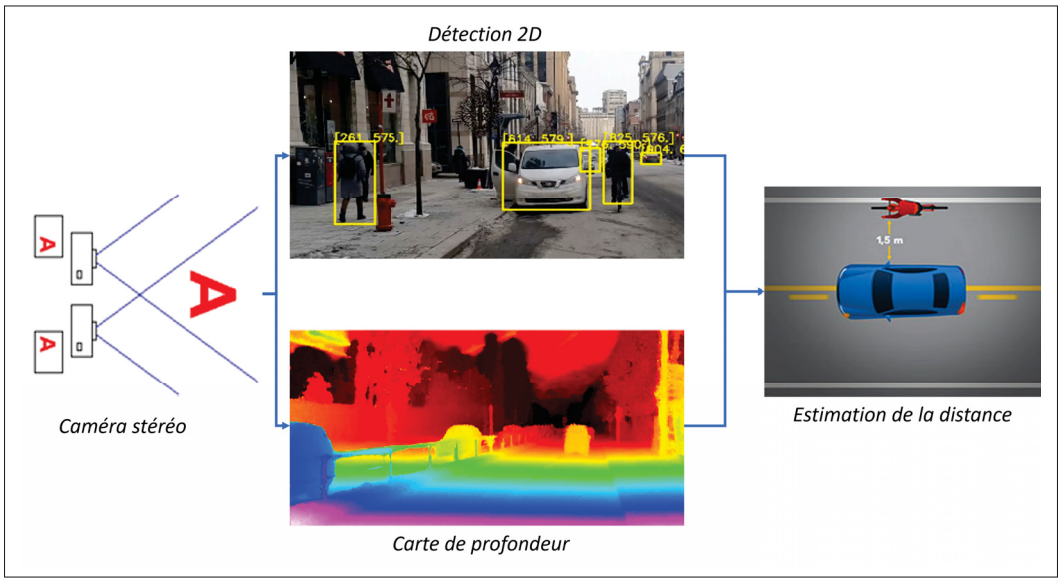

FIGURE 0.5 Schéma de l'approche "Dista" pour l'estimation de la distance lors d'un dépassement d'un vélo

Les chercheurs de l'école André Laurendeau ont donc proposé le système suivant (voir figure 0.5) :

Ils proposent d'utiliser la vision binoculaire par une caméra stéréoscopique permettant de capturer deux images de la scène, mais aussi de déterminer la carte de disparité à l'aide d'un algorithme de *stereo-matching* (le SGBM de Hirschmueller (2008)). Une carte de disparité est un rassemblement des distances de correspondance entre les pixels de l'image gauche et

droite, et permet de générer une carte de profondeur de la scène. En parallèle, un algorithme en *deep learing* pour la détection et reconnaissance d'objet 2D (le EfficientDet de Tan, Pang & Le (2020)) permet de situer dans l'image le cycliste et l'automobiliste réalisant le dépassement. La position 3D et l'orientation des objets est ensuite estimées grâce aux informations précédentes, afin de produire des boites les encadrant en trois dimensions. Finalement, la distance latéral entre les boites 3D du cycliste et du véhicule est prédite.

L'ensemble du système est construit à partir d'un ordinateur d'IA compact Jetson Nano de la marque NVIDIA, permettant de gérer des algorithmes *deep learning* sur des processeurs graphiques (GPU) de haute puissance de calcul. Cet ordinateur a la particularité d'être peu cher, petit et très léger, mais d'avoir une faible mémoire disponible (voir figure 0.6).

FIGURE 0.6 Image d'une Jetson Nano de NVIDIA
Tirée de www.nvidia.com

Le *deep learning* appliqué à la vision par ordinateur a permis de grandes avancées dans le domaine de la détection d'objet 2D et ainsi de concevoir des méthodes ayant une précision supérieur à 90% (He, Zeng, Huang, Hua & Zhang (2020)) sur des bases de données références comme KITTI de Menze, Heipke & Geiger (2015), avec des vitesses de prédiction sur GPU inférieur à 60ms par image (seuil pour avoir l'illusion d'un mouvement fluide). C'est pourquoi un réseau de détection 2D a été choisi pour le projet "Dista". Cependant, l'algorithme permettant

de déterminer la profondeur de la scène, le SGBM (Semi-Global Matching), ne peut fonctionner sur GPU et n'a pas la capacité à s'adapter au type de scène observée, ce qui entraîne de très faibles performances (classé 264ème sur la base KITTI avec une vitesse de prédiction de 1,1s par image). Ces limitations vont affecter fortement l'estimation de la distance latérale.
Le travail présent va donc explorer une nouvelle approche plus performante, conçue sur des modèles en IA, permettant la détection 3D et l'estimation des disparités à partir d'images stéréoscopiques.

0.2 Contributions

Avec le succès grandissant du *deep learning* pour la détection et reconnaissance d'objets, et les challenges à surmonter pour concevoir un système de perception automatique performant, ce mémoire a été initié par les chercheurs de l'école André-Laurendeau dans le but de contribuer à la conception d'une technologie basée sur une IA et la vision par ordinateur, pour mesurer en temps réel la distance latérale lors d'un dépassement.
L'objectif principale est de développer l'architecture d'un modèle en *deep learning* pour la détection 3D d'objets à partir d'images stéréoscopiques, permettant d'obtenir les informations essentielles pour déterminer la distance latérale entre deux objets.

En cherchant la méthode adaptée à notre situation, nous avons dû faire face à plusieurs problèmes. Tout d'abord, la tache de détection d'objet 3D par images stéréos ("stéréo" comme diminutif de stéréoscopique) est un domaine qui a été traité depuis peu (Chen *et al.* (2015) sont parmi les premiers), et dont à peine une vingtaine de papiers existe contre plus de 2 000 pour la tache de détection 2D [1]. Cette différence est due à la complexité de la tache de détection 3D qui nécessite d'introduire une troisième dimension, estimée généralement en passant par la profondeur (mono) ou les disparités (stereo).

[1] selon https://paperswithcode.com/

De plus, on observe un écart très important des performances (vitesse et précision) entre les méthodes de détection 2D et 3D par images, ce qui rend l'utilisation de ces dernières encore limitées. A ce jour, les méthodes 3D les plus performantes sont celles exploitant des nuages de points 3D lidar en entrée (Arnold, Al-Jarrah & al. (2019), Menze *et al.* (2015)).
Ensuite, contrairement aux méthodes 2D qui sont alimentées par de vaste bases de données regroupant des images en accès libre sur internet, les méthodes 3D nécessitent des bases d'images stéréos qui sont plus difficiles à obtenir et à annoter (boites 3D, catégories d'objet, cartes de disparité dense). Une seule base de données d'images stéréos réelles est à ce jour employée par l'ensemble des méthodes de la littérature (la base KITTI qui contient 3 000 données pour la détection 3D et 200 pour l'estimation des disparités), et seulement l'objet "voiture" est détecté. Ce qui pousse les chercheurs à développer leur modèle à l'aide de base de données stéréos synthétiques, plus facile à générer (par exemple grâce à des simulateurs 3D de jeux vidéo). Cependant, on observe que malgré la disponibilité d'une grande quantité de données synthétiques, les méthodes n'arrivent pas à performer suffisamment dans des conditions réelles : les modèles ont des difficultés à généraliser les informations 3D apprises sur le domaine synthétique vers le réel. En effet, de nombreuses contraintes sont à prendre en compte dans le monde réel qu'on ne retrouve pas dans un environnement simulé (couleurs, textures, régions répétitives, surfaces réflectives, luminosité, météo...).
Ainsi, ce mémoire ne va pas se concentrer directement sur la détection 3D, mais sur l'étape de disparité qui est essentielle à la construction de la représentation 3D de la scène observée à partir d'images stéréos. On va chercher à optimiser la précision, et les capacités d'adaptation sur des données réelles de cette étape.

La première contribution est la réalisation, à notre connaissance, de la toute première revue de littérature recensant les méthodes basées sur des images stéréos pour : la détection d'objet 3D et l'estimation des disparités. Mais nous allons aussi présenter le principe d'adaptation de domaine non-supervisé appliqué à la détection et l'estimation des disparités. C'est à l'aide de l'adaptation

de domaine non-supervisée que nous allons chercher à freiner la chute des performances des méthodes de détection 3D et de disparité sur des images stéréo réelles, provoquée par le manque d'annotations.

Dans un deuxième temps, nous allons montrer que les méthodes de détection 3D par stéréo employant une méthode estimant les cartes de disparité, obtiennent les meilleurs performances. Nous allons nous concentrez sur les sources principales d'erreurs lors de la détection (sur des images réelles), et plus précisément sur la méthode estimant les disparités. Selon plusieurs critères, nous allons choisir parmi l'ensemble des méthodes de la littérature le réseau DispNetC de Mayer & Ilg (2016), qui est la méthode proposant les performances et l'architecture (encodeur-décodeur) les plus adaptées à notre problème.
Puis à l'aide des progrès observés pour la structure encodeur-décodeur dans le domaine de la segmentation sémantique, nous allons proposer une nouvelle architecture du DispNetC, le DispNetCv4, permettant d'améliorer considérablement les performances du modèle de base, et assurant un très bon compromis précision-vitesse. Nous allons pour cela étudier l'ajout de modules d'agrégation spatiale et locale, et des modules d'attention spatial, de channel et de résolution.

On a pu voir que la faible disponibilité des bases d'images stéréo réelles avec cartes de disparité ne permettaient pas de couvrir l'ensemble des données observable dans le monde réel. Ainsi, concevoir un modèle de *deep learning* avec peu de données entraîne une chute importante des performances lors du déploiement sur le terrain.
Pour notre troisième contribution, nous allons chercher à adapter au domaine d'une manière non-supervisée le réseau de disparité proposé, pour réduire la perte de précision lorsque le modèle est appliqué sur des données réelles. Pour cela nous allons considérer comme domaine source annotée la base synthétique SceneFlow de Mayer & Ilg (2016), et comme domaine cible la base réel KITTI de Menze *et al.* (2015) dont on suppose que les cartes de disparité sont

inconnues. Nous allons entraîner le modèle de disparité par adaptation de domaine selon les principes de *adversarial learning* (apprentissage adverse) et de reconstruction d'image. Nous allons employer un réseau de translation d'images non-appariées, le CycleGAN de Zhu, Park, Isola & Efros (2017), pour translater les images synthétiques vers le réel, puis entraîner le réseau de disparité sur ces images "pseudo-réel" avec annotations. Un discriminateur de caractéristiques intermédiaire robuste, inspiré de Kundu, Uppala, Pahuja & Babu (2018), sera placé à une couche optimale du modèle, pour que les représentations internes soient invariantes au domaine.
Les expériences vont permettre de conclure sur l'efficacité de la stratégie d'apprentissage proposée à améliorer les performances de la méthode de disparité lorsqu'elle est employée sur des données réelles dont les annotations sont inconnues, sachant qu'elle a été entraînée seulement avec des cartes de disparité synthétiques.

Finalement, nous allons évaluer les performances du réseau de disparité proposé, sans et avec adaptation de domaine, pour la tache de détection 3D. Nous allons employer la méthode de détection stéréo Pseudo-Lidar de Wang, Chao & al. (2019) qui est l'approche proposant l'architecture la plus adaptée à notre problème, tout en ayant des performances état de l'art.
En comparant les résultats obtenus avec ceux de la littérature, on démontre l'efficacité du DispNetCv4 à représenter plus finement la scène 3D tout en assurant une vitesse et empreinte mémoire raisonnable, lors de la prédiction. La méthode d'adaptation de domaine permet de diminuer considérablement la chute de performance dans l'estimation des disparités et dans la détection 3D, lorsque le modèle est appliqué sur le domaine réel.

0.3 Organisation

Le mémoire est divisé en 4 chapitres et une conclusion :

1. Le chapitre 1 concerne la revue de la littérature pour introduire les principes d'apprentissage profond, de détection et de stéréovision. Nous présentons les méthodes de détection 3D par stéréo existantes, les méthodes de disparité et les méthodes employant l'adaptation de domaine non-supervisée pour le sujet traité. Le chapitre liste leurs avantages et limitations, afin de justifier les contributions réalisées ;
2. Le chapitre 2 présente la méthode de disparité : les raisons qui ont poussé au choix de la méthode de détection 3D par stéréo et de la méthode de disparité. La structure du réseau de base est introduite, puis chaque amélioration apportée est soigneusement présentée, ainsi que leur impact sur les performances globales ;
3. Le chapitre 3 est dédié à l'adaptation de domaine non-supervisée pour l'estimation des disparités sur le domaine réel non annoté. On y introduit les principes d'apprentissage adverse et de translation d'image. Chaque étape de la stratégie d'apprentissage par adaptation est détaillée : le CycleGAN est employé comme réseau de translation, le DispNetCv4 estime les disparités et un discriminateur de caractéristiques intermédiaire confond les représentations interne synthétique et réelle ;
4. Le Chapitre 4 sert à évaluer l'approche global pour la détection 3D en associant notre réseau de disparité avec un détecteur 3D par lidar comme proposé par le Pseudo-Lidar. Notre méthode de détection 3D est testée sur KITTI d'une manière supervisée, et non-supervisée en utilisant le réseau de disparité entraîné par adaptation ;
5. Le dernier chapitre permet de conclure sur le travail effectué en résumant les résultats principaux, le travail restant et les recommandations.

CHAPITRE 1

REVUE DE LITTÉRATURE

1.1 Mise en contexte

1.1.1 Principe d'apprentissage

L'Intelligence Artificielle (IA) est par définition le regroupement de principes et techniques permettant à une machine d'imiter l'intelligence humaine. Elle permet de répondre à des problèmes pouvant être décrit par un ensemble de fonctions et lois mathématiques, mais aussi des problèmes beaucoup plus complexes à formuler.
On retrouve entre autres dans l'IA les domaines d'apprentissage machine (*machine learning*) et d'apprentissage profond (*deep learning*) (figure 1.1).

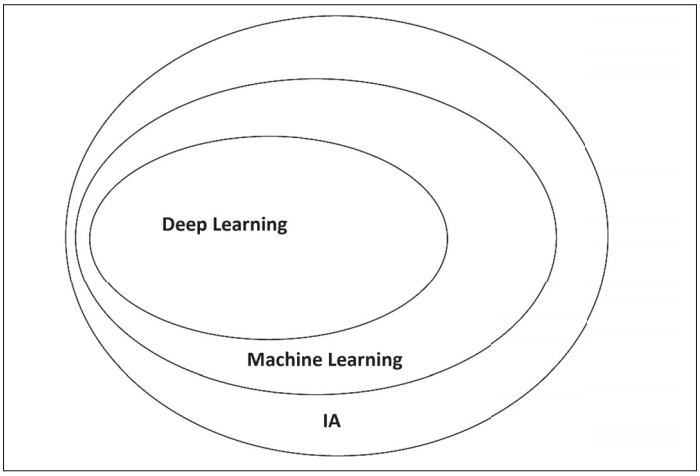

FIGURE 1.1 Place de l'apprentissage machine et profond au sein de l'IA
Adaptée de Goodfellow *et al.* (2015)

L'apprentissage machine (*Machine Learning*) est un sous-ensemble de l'IA donnant la capacité à une machine d'apprendre sans être explicitement programmée. L'algorithme exploite des données d'apprentissage pour en acquérir des connaissances et utilise ces connaissances pour

réaliser la tâche demandée (Goodfellow *et al.* (2015)). Les méthodes conventionnelles en apprentissage machine étaient conçues pour le premier type de problème énoncé plus haut, tels que la classification de données, le *clustering* (regroupement de données), traitement de données bancaire, reconnaissance de formes... Leurs performances dépendent fortement de comment les données en entrée sont pré-traitées (Goodfellow *et al.* (2015)) : les modèles encodent les informations (appelées caractéristiques ou *features*) contenues dans la donnée ou dans une image dans un vecteur caractéristique (*feature vector*), qui va garder seulement celles les plus importantes (projetant la donnée dans un espace de plus faible dimension).
Le processus d'extraction des caractéristiques de la donnée était conçu par des experts « à la main » et était lié au problème à traiter. Ainsi, ces méthodes sont dépendantes de la tâche considérée, elles demandent des bases de données très structurées (donnée avec label) et deviennent difficiles à concevoir quand le problème n'est plus formulable mathématiquement.

1.1.2 Apprentissage profond

Pour faire face aux limites rencontrées en apprentissage machine, l'apprentissage profond (*Deep Learning*) a été proposé afin d'acquérir des connaissances plus complexes, de répondre plus efficacement à un problème, le tout d'une manière automatique (Goodfellow *et al.* (2015)). L'algorithme est composé de plusieurs « couches » : les caractéristiques les plus simples et élémentaires sont apprises en premières et vont permettre aux couches profondes d'apprendre des caractéristiques plus abstraites et complexes pour réaliser la tâche demandée (voir figure 1.2). Un modèle profond se construit de A à Z (*end-to-end*) grâce à des exemples d'apprentissage en entrée et leurs vérités terrain (ou *ground truth* en anglais) en sortie. La vérité terrain d'une donnée (appelé aussi label ou annotation) correspond au vrai résultat attendu en sortie de l'algorithme. Il permet d'évaluer l'erreur pour ensuite optimiser automatiquement les paramètres du modèle. Contrairement aux méthodes d'apprentissage machine, l'algorithme profond nécessite beaucoup moins d'expertise humaine : il n'est pas spécifique à un problème et peut être adapté pour différentes tâches. De plus, les bases de données employées ne sont pas forcément structurées.

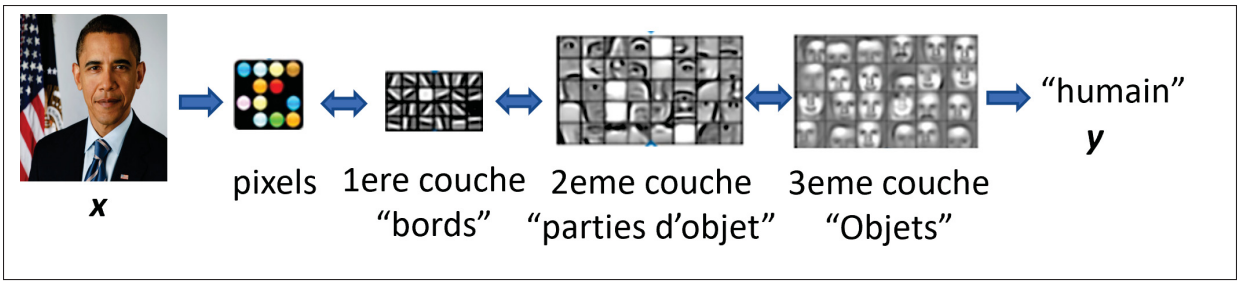

FIGURE 1.2 Illustration d’un modèle profond pour la classification d’images*
Adaptée de Kordon (2020)

*L’algorithme apprend d’abord les informations de couleur et bord de l’image x, puis celles pertinentes de l’objet pour estimer sa classe y

Selon la disponibilité des *ground truth* dans une base de données, il existe différents types de supervision d’un modèle. L’apprentissage dit supervisé est le cas trivial : on aide le modèle à faire la bonne prédiction lors de son apprentissage en lui fournissant le vrai résultat soit l’annotation $\hat{y}$ associées à l’entrée x. Lorsque les annotations ne sont pas disponibles, on est dans le cas non-supervisé. Les modèles profonds ont de meilleures capacités à apprendre des caractéristiques importantes, que les approches d’apprentissage machine. La figure 1.3 est un exemple de 3 types de supervision pour la tâche de segmentation. Le principe est de détermine le contour et la catégorie de chaque objet.

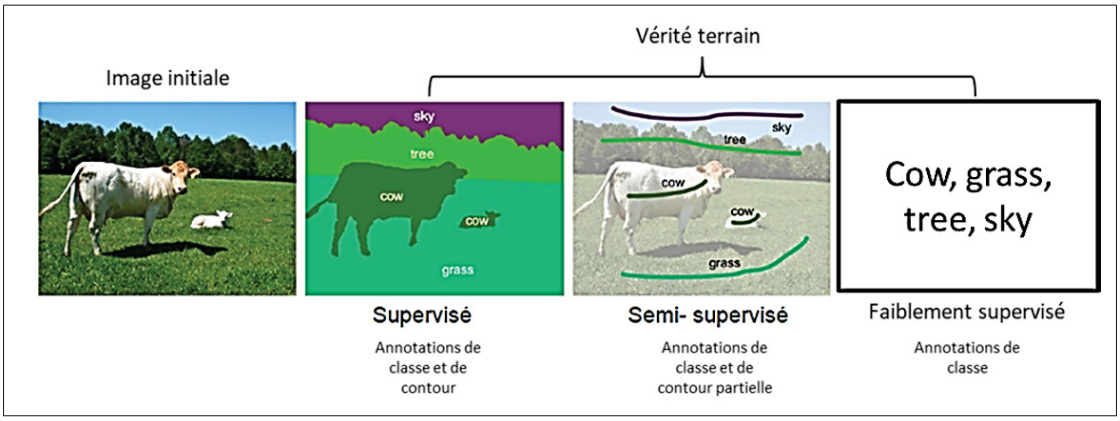

FIGURE 1.3 Vérités terrain selon le type de supervision
Adaptée de Ben Ayed (2019)

De manière générale, un modèle en apprentissage profond fonctionne comme suit (figure 1.2) :

1. **Lors de l'apprentissage.** A partir d'une base de données $\{x, \hat{y}\}_i$ où x est la donnée d'entrée associée à une annotation $\hat{y}$, le modèle profond va apprendre par lui-même les informations et caractéristiques importante de x lui permettant de prédire en sortie une valeur $y \approx \hat{y}$. L'erreur de prédiction entre y et $\hat{y}$ est utilisée pour optimiser ses paramètres ;
2. **Lors de la prédiction.** Le modèle reçoit en entrée la donnée x, et grâce aux connaissances apprises durant l'apprentissage, il va prédire la valeur y sans connaître $\hat{y}$.

Les modèles les plus utilisés en apprentissage profond sont appelés des réseaux de neurones (ANN ou *Artificial Neural Networks*). Ils sont composés de neurones artificiels appelés perceptrons (voir figure 1.4) répartis sur plusieurs couches. Ils proviennent de l'idée du réseau de milliards de neurones biologiques et de connections contenu dans notre cerveau (Pedersoli (2020)). Un perceptron réalise la somme pondérée entre le vecteur en entrée et un vecteur poids correspondant aux paramètres du neurone. Une fonction d'activation non-linéaire est appliquée à la sortie, puis une classification binaire est effectuée (Ben Ayed (2019)).

Les premiers réseaux profond étaient les MLP (Multi Layer Perceptron) dont l'idée est de cumuler les perceptrons sur des couches et de créer plusieurs couches interconnectées se succédant afin de résoudre des problèmes à plusieurs classes et plus complexes (figure 1.5) (Pedersoli (2020)). Cependant les MLP ne sont pas adaptés pour les données de très grande dimension comme les images ou vidéos. Pour cette raison, des réseaux spécialisés ont vu le jour : les réseaux de convolution (CNNs ou *Convolutional Neural Networks*) (figure 1.6). Ils sont aussi composés de plusieurs couches mais les neurones sont connectés entre eux par des opérations de convolution. Ils sont inspirés de la façon dont le cortex visuel des animaux fonctionne : il pré-traite de petite quantité d'information se chevauchant au lieu de traiter la vision dans son ensemble (Chane (2021)). En supplément des couches de convolution, différentes fonctions (d'activation, de régularisation, de normalisation...) sont ajoutées pour améliorer les performances des CNNs.

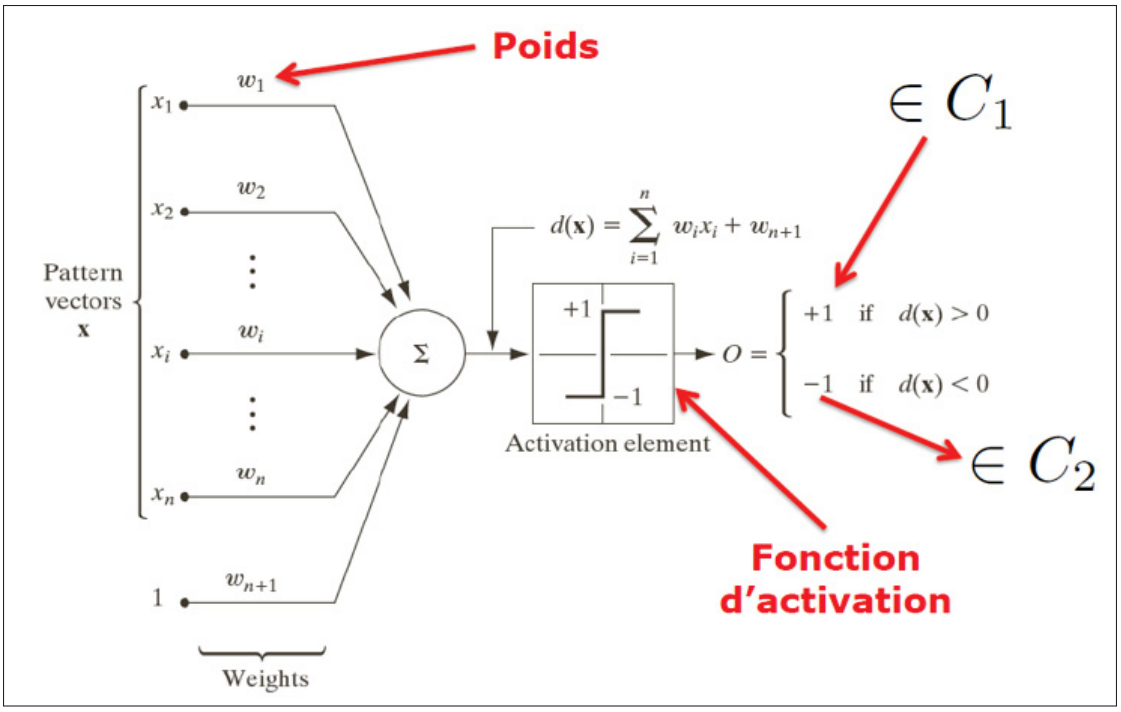

FIGURE 1.4 Illustration d'un Perceptron*
Tirée de Ben Ayed (2019)

*Il réalise la somme pondérée d'un vecteur entrée x à n dimension à l'aide des poids w_i, puis applique une fonction d'activation non-linéaire pour effectuer la classification.

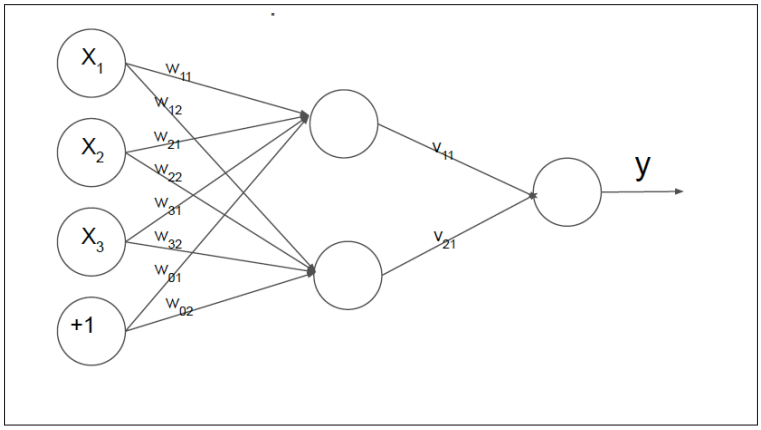

FIGURE 1.5 Structure d'un MLP*
Tirée de Pedersoli (2020)

*Chaque couche est composée par multiples neurones.

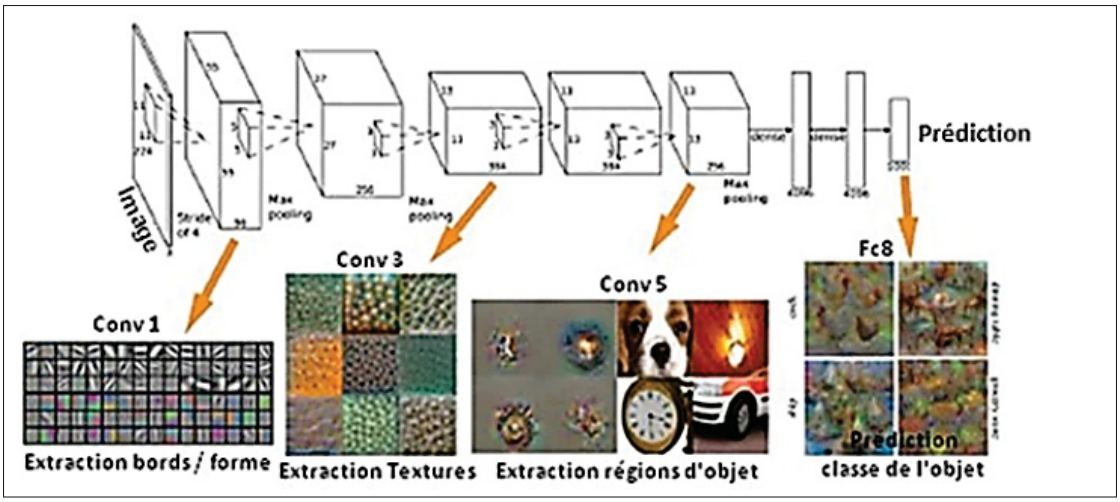

FIGURE 1.6 Illustration d'un CNN*
Adaptée de Ben Ayed (2019)

*Plus la couche est profonde et plus l'information apprise est importante pour la prédiction.

L'apprentissage profond a permis de résoudre des problèmes très complexes, et a accéléré les progrès dans différents domaine tels que la reconnaissance de voix, le traitement de texte et la vision par ordinateur. Dans la suite, nous allons nous intéresser à la détection d'objet, qui est une application très étudiée dans les domaines de vision par ordinateur et d'apprentissage profond.

1.1.3 Détection d'objet

La vision par ordinateur (*Computer Vision*) est un champ d'étude qui consiste à développer des techniques permettant à des machines de « voir » et comprendre le contenu d'images ou vidéo (Wikipedia (2006)). Cela est très simple pour l'œil humain mais très complexe à appliquer à un ordinateur. La détection d'objet est un problème fondamental en vision par ordinateur qui suscite de nombreuses recherches depuis plusieurs années. Elle est ainsi présente dans un large panel d'application, comme la vision robotique, la surveillance et sécurité, le tracking et la conduite autonome. La complexité de cette tâche et son importance dans un grand nombre d'application en vision par ordinateur obligent des améliorations sans cesse des méthodes de détection.

Le principe de détection d'objet se définit comme suit : à partir d'une image, il faut prédire si un ou des objets appartenant à une classe prédéfinie sont présents, et si c'est le cas, il faut retourner

leurs positions spatiales ou les boites encadrant les objets (figure 1.7) (Liu, Ouyang & al. (2019)). Il est donc nécessaire tout d'abord d'être capable de reconnaître le ou les objets dans l'image. Cette tâche, se nommant la classification d'objets, cherche à déterminer les classes des objets présents dans l'image (figure 1.7). La classification d'objets est un problème qui a été fortement traité en vision par ordinateur, et l'ajout de l'information de position des objets complique le problème (Liu *et al.* (2019)).

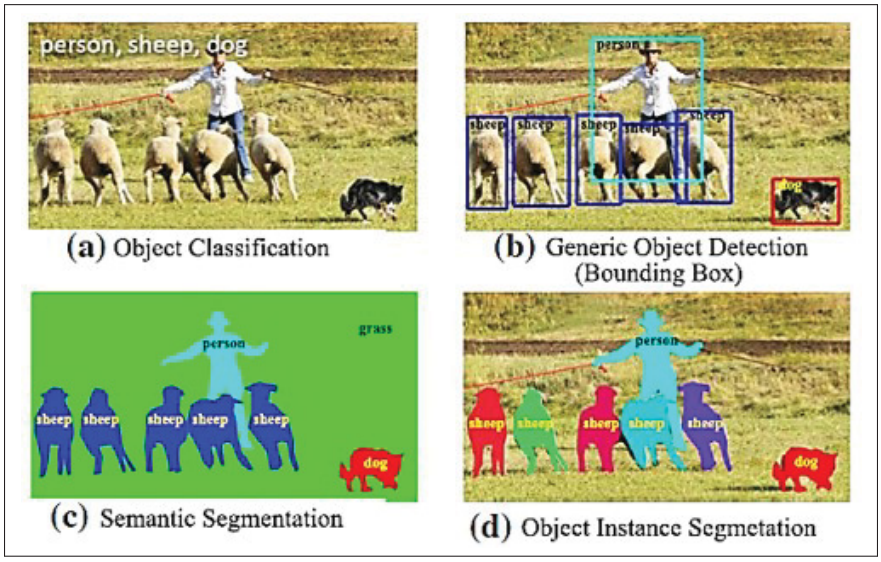

FIGURE 1.7 **a** Classification d'objets, **b** détection 2D d'objets générique, **c** segmentation sémantique au niveau des pixels, **d** segmentation sémantique des classes objets
Tirée de Liu *et al.* (2019)

Les objets à détecter peuvent être issus de classes spécifiques c'est à dire des classes représentant un objet unique (comme le visage de Barack Obama, la Statue de la Liberté. . .) ou bien ils peuvent aussi être issus de catégories génériques (voitures, chiens, vélos. . .) (figure 1.8) (Liu *et al.* (2019)). Ce dernier type de classe représente un défi important dans le domaine de la vision par ordinateur : en effet, une même classe va représenter un large éventail d'objet ayant des couleurs, textures, matériaux, tailles, poses, formes variées. Ces variations au sein même d'une classe et le nombre de classe considéré rendent la tâche de discrimination très difficile.

FIGURE 1.8 Les objets à détecter peuvent être des objets spécifiques (haut) ou bien des catégories d'objets (bas)

Il faut ajouter à cela l'influence de facteurs externes qui vont complexifier le problème. Les images considérées peuvent être issues d'un environnement contrôlé mais aussi non-contrôlé (image vidéo). L'apparence d'un objet est alors modifiée par les conditions météorologiques (pluie, brume), l'éclairage (jour, nuit, ombre), sa position dans l'espace (proche, en hauteur), la présence d'occlusions (un objet bloque la vue d'un autre), l'arrière-plan, la caméra (faible résolution, focus, flou, 1.9) (Liu *et al.* (2019)). Il faut ainsi être en mesure d'apprendre une représentation complexe des classes pour réaliser une frontière efficace entre elles.

Connaître la position spatiale d'un objet à l'aide d'une image peut être réalisé de plusieurs manières : à l'aide d'une boite 2D ou 3D (rectangle ou parallélépipède encadrant étroitement l'objet) ; ou bien grâce à la segmentation de chaque objet (assigne à chaque pixel une classe d'objet sémantique permettant de réaliser le contour des objets) (Rahman, Tan & al. (2020)). Dans le cas de la détection 2D ou 3D, l'algorithme prédit la classe de l'objet, les coordonnées du centre de la boite ainsi que la largeur et hauteur. Un objet pouvant avoir différentes apparences

dans une image, la création de la boite délimitante est une tâche complexe. L'utilisation d'une boite 3D demande d'estimer en supplément des informations de profondeur par rapport à la détection 2D : le détecteur doit connaître, en plus de la position dans l'espace de la boite 3D, son orientation et ses dimensions (Rahman *et al.* (2020)).

Un algorithme développé en détection d'objets doit assurer une haute précision, robustesse et efficacité. Il doit être capable d'identifier et localiser différents objets dans une image fixe ou image vidéo, en prenant en compte les variations d'apparences au sein d'une classe, mais aussi le grand nombre de classe-objet possible, qu'il va devoir distinguer. Pour de nombreuse applications, il est nécessaire que la détection soit en temps réel, et que les ressources mémoires et de calculs soient le plus faible possible afin de fonctionner sur des systèmes embarqués.

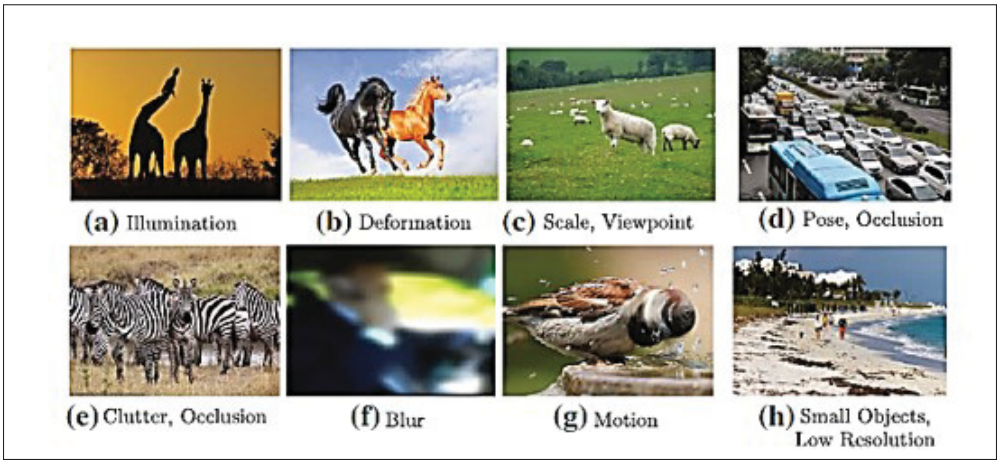

FIGURE 1.9 (**a-h**) Variations des conditions d'imagerie entraînant une modification de l'apparence d'objets issus d'une même classe
Tirée de Liu *et al.* (2019)

1.1.4 Apprentissage profond et détection

Les premières méthodes de reconnaissance et détection d'objet étaient basées sur les informations géométriques (contour, forme) de l'objet limitant leurs emplois à des objets spécifiques. Puis les chercheurs se sont intéressés à l'utilisation de classificateurs statistiques, basés sur des caractéristiques encodant les informations d'apparence de l'objet. Ces méthodes associaient

un extracteur de caractéristiques locales invariantes conçu par des experts (tels que *Scale Invariant Feature Transform* (SIFT) de Lowe (1999) ou *Local Binary Patterns* (LBP) de Ojala, Pietikäinen & Mäenpää (2002)) avec un classificateur réalisant la prédiction (comme *Support Vector Machine* (SVM) de Osuna, Freund & Girosit (1997)).

L'émergence considérable des techniques en apprentissage profond, dont les réseaux convolutionnels, ont permis de développer ces dernières années des modèles de reconnaissance et détection d'objet puissants résolvant des problèmes complexes. Les méthodes en apprentissage profond ont la particularité d'apprendre automatiquement des représentations robustes et abstraites à partir d'une grande quantité de données d'entraînement, grâce à leur large structure de millions de paramètres. La tâche de reconnaissance et détection 2D en apprentissage profond a été fortement étudiée par les chercheurs, et cela depuis plusieurs années. On est ainsi capable d'atteindre de très hautes performances (rapidité et précision) dans la détection d'objet 2D. Les méthodes se divisent en deux catégories principales :

- les méthodes de détection à deux étapes, associant un réseau de proposition des régions d'intérêts (RPN) avec un réseau de détection (RCNN de Girshick, Donahue, Darrell & Malik (2014) ou Faster-RCNN de Ren, He, Girshick & Sun (2015)) ;
- les méthodes de détection à une étape, effectuant la détection directement à partir de l'image initiale (SSD de Liu *et al.* (2016)).

A contrario, les méthodes de détections 3D ont été beaucoup moins traitées dans la littérature, entraînant un écart de performance signifiant avec les méthodes 2D (Arnold *et al.* (2019)).

La compréhension d'une scène en 3D est un problème en vision par ordinateur important et crucial pour beaucoup d'application (comme par exemple éviter des obstacles en conduite autonome). La détection 3D fournie des informations de profondeur, de pose et d'occlusion qu'on ne peut obtenir par détection 2D. Cependant, les méthodes de détection 3D en apprentissage profond vont être plus complexes à concevoir, plus coûteuses en ressource de calculs et dans la génération de la base de données, qui nécessite des annotations 3D .
Il existe 3 grandes familles de méthodes de détection 3D (Arnold *et al.* (2019)). Ces familles

se différencient selon les capteurs utilisés pour générer les données d'apprentissage (voir table 1.1) :

1. **Caméra.** Elle produit des images RGB avec des informations de couleurs, formes et textures sous forme d'intensité de pixel. Ce capteur a la particularité d'être à faible prix et facilement accessible. Mais ses performances sont affectées par les régions répétitives et sans textures, par les conditions de temps et conditions lumineuses (par exemple lors d'entrée et sortie d'un tunnel). La prédiction de la boite 3D d'un objet à l'aide d'une image est réalisée en associant une méthode de détection 2D et des contraintes géométriques 3D. Cependant, l'information de profondeur n'est pas fournie explicitement ce qui n'assure pas de bonne précision. On peut récupérer une information de profondeur plus précise en utilisant une caméra stéréoscopique fournissant deux images de points de vue différents. La méthode de *stereo-matching* (appariement pixellique) est alors utilisée pour obtenir une carte de profondeur de bonne densité ;

2. **Lidar.** C'est un capteur émettant des rayons laser permettant de mesurer la distance des obstacles au système, grâce au temps écoulé entre l'émission et la réception du rayon. Le lidar va alors générer un nuage de points 3D de la scène. Il permet des mesures précises des distances mais la densité du nuage de points va dépendre du lidar utilisé. Contrairement à une caméra, il n'est pas affecté par les conditions lumineuses (très efficace durant la nuit) mais ne fournit pas d'information de texture. Son coût est élevé ce qui limite son accessibilité. Les méthodes de détection 3D utilisant les nuages de points d'un lidar vont soit transformer en voxels les points 3D ou bien projeter ces points dans un plan image 2D pour réaliser la détection. Le temps de traitement et le coût de calcul de ces méthodes sont très élevés ;

3. **Fusion.** La dernière famille de méthode de détection 3D correspond aux méthodes dites Fusion, combinant l'image RGB de la caméra et le nuage de points 3D lidar de la scène. Ces méthodes sont robustes et réalisent les meilleures performances en détection 3D. Elles associent les informations 3D du nuage de points avec les informations de texture et couleur d'une image. Elles nécessitent une calibration précise entre les capteurs et demandent beaucoup de ressources informatiques.

Tableau 1.1 Comparaison des différentes familles de détection 3D.
Adapté de Arnold *et al.* (2019)

Capteur	**Méthodes**	**Limitations**
Caméra monoculaire	Utilise une image RGB pour prédire la boite 3D à l'aide de la boite 2D et de contrainte géométrique 3D	Manque une information explicite de profondeur : faible performance de localisation
Lidar	Utilise un nuage de points brut issu d'un capteur LiDAR. Les points 3D peuvent être transformés en voxels ou projetés dans une image 2D pour réaliser la détection 3D	La considération du nuage de points brut entier augmente le temps de traitement et le coût de calcul
Caméra-stéréo	Utilise une paire d'image stéréo pour obtenir la carte de disparité afin de réaliser la détection 3D	Le *stereo-matching* demande du temps et un coût de calculs important. Les erreurs de disparité diminuent fortement la précision de la détection
Fusion	Combine une ou des images RGB avec un nuage de points pour réaliser une détection 3D robuste. Réalise les meilleures performances	Demande un coût de calculs important. Nécessité d'une calibration précise entre les capteurs.

Nous allons nous intéresser par la suite aux méthodes de détection 3D par image stéréo. Pour cela, la partie suivante va introduire le principe de stéréovision.

1.1.5 Stéréovision

La stéréoscopie regroupe l'ensemble des techniques permettant de reconstruire la profondeur à partir d'une image stéréoscopique composée de deux images planes de la même scène (Wikipedia (2014)). Elle s'appuie sur le principe de la perception humaine du relief grâce aux images perçues en simultané par chaque œil.

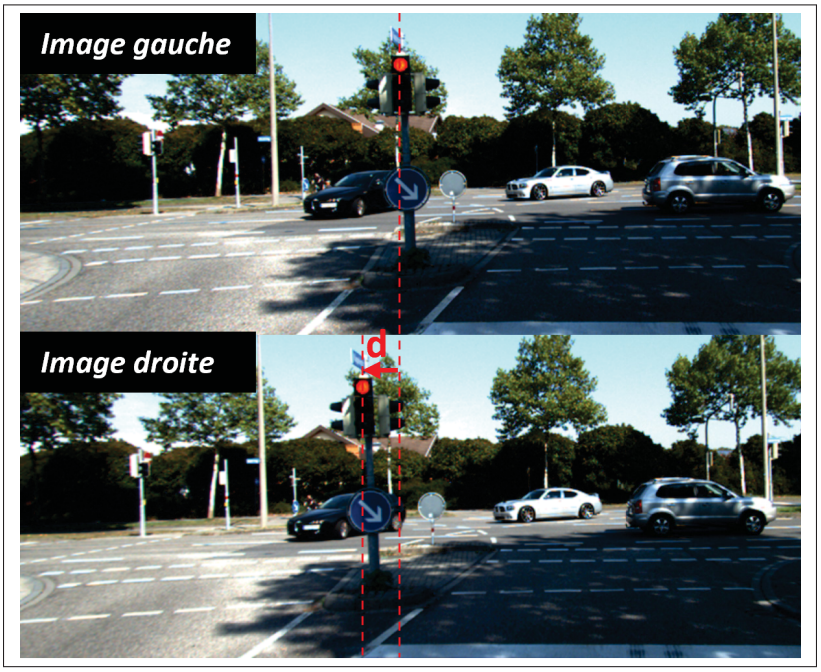

FIGURE 1.10 Illustration des disparités sur une image stéréoscopique*
Adaptées de Menze *et al.* (2015)

*Le feu rouge de l'image gauche se retrouve déplacé vers la gauche dans l'image droite, d'une distance d représentant la disparité.

Son fonctionnement est le suivant (Zhou, Meng & al. (2020)) :

À partir d'un système optique composé de deux caméras (une caméra gauche de centre optique C_1, et distance focale f_1 et une caméra droite de centre optique C_2 et distance focale f_2) à la même hauteur, séparées d'une distance B (*baseline*) et à une distance h de la scène, deux images d'une même scène vont être obtenues en simultané. Ces deux images (nommées image gauche et droite) étant acquises avec des points de vue différents, la profondeur de la scène crée une

disparité géométrique entre elles. Plus simplement, un point de la scène sur la première image sera présent sur la seconde image mais décalé d'une distance représentant la disparité (voir figure 1.10). Plus un point est proche et plus sa valeur de disparité sera grande. Ainsi, à partir de l'image stéréoscopique, on est capable de reconstruire la carte de disparité qui va indiquer pour chaque point du plan de la scène, la valeur de disparité associée. La disparité peut être mesurée en distance ou en pixel.

FIGURE 1.11 Exemple d'images stéréoscopiques et de la carte de disparité
Adaptées de Mayer *et al.* (2016)

La stéréovision consiste à mesurer les informations 3D de la scène issue de l'image stéréoscopique. Notre cerveau à la capacité de mesurer la disparité et de l'employer pour créer la sensation de profondeur (Marr (1982)). Les algorithmes de stéréovision utilisent les disparités géométriques pour obtenir la profondeur dans une image et déterminer les dimensions, les formes ou les positions d'objets. La disparité est la différence horizontale entre les coordonnées d'un pixel de l'image gauche avec celui de l'image droite correspondant au même point de la scène. On obtient la profondeur en un point $X(x, y, z)$ par la fonction suivante :

$$Z(X) = \frac{f \times B}{y(X)} \tag{1.1}$$

où f est distance focale du système de caméra stéréo, B est la baseline et $y(X)$ la fonction de disparité en X.

La figure 1.12 est un schéma du principe de stéréovision pour le calcul des disparités selon

l'équation 1.1. On peut ainsi voir que la disparité est inversement proportionnelle à la profondeur. Une carte de profondeur indique en chaque pixel la distance subjective perçue par l'observateur vers le point.

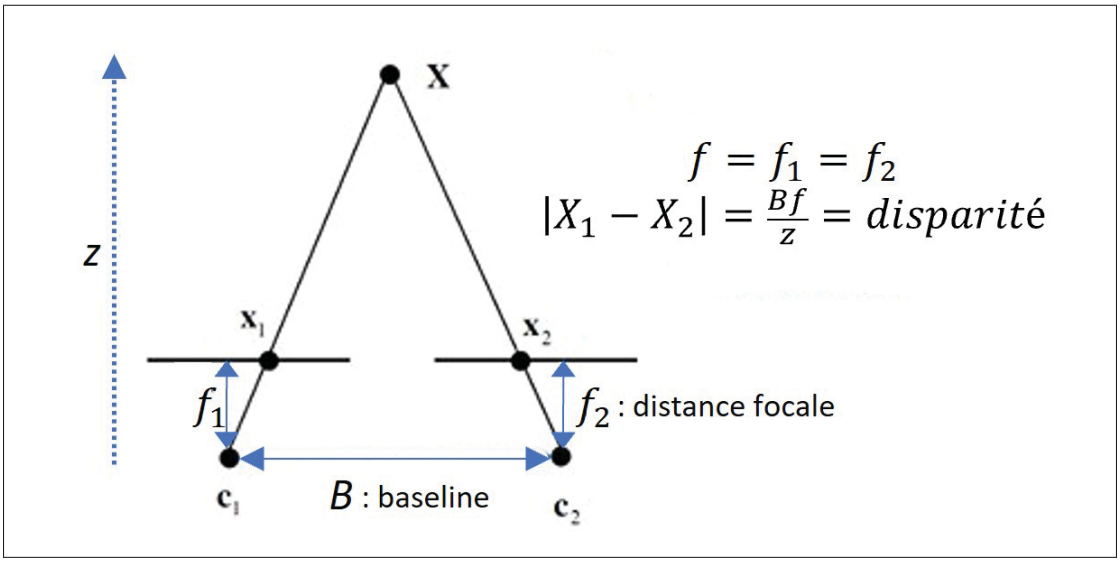

FIGURE 1.12 Schéma du principe de la stéréovision*

*Un point X de la scène est représenté en X_1 sur l'image de la caméra gauche de centre optique C_1 et de focale f_1, et en X_2 sur l'image de la caméra droite de centre optique C_2 et de focale f_2. Connaissant la disparité en X on peut connaître la profondeur, et inversement.

Afin de générer une carte de disparité à partir de l'image stéréoscopique, on applique une méthode dite de *stereo-matching* (appariement pixellique). Le but de l'algorithme est de matcher les pixels de l'image gauche avec leurs correspondants de l'image droite pour déterminer les disparités. Du fait que les deux images représentent une même scène, elles possèdent des relations géométriques pouvant être décrites par une géométrie épipolaire. Et lorsque qu'une image stéréoscopique est rectifiée, la contrainte épipolaire devient rectiligne, et l'espace de recherche des pixels correspondant dans les images gauche et droite devient ainsi limité à une ligne horizontale (problème à une dimension). Il est ainsi plus pratique de travailler avec des images rectifiées (Zhou *et al.* (2020)).

Pour rectifier une image, il est nécessaire de calibrer les caméras employées. La calibration consiste à obtenir les paramètres intrinsèques et extrinsèques de la caméra (distance focale,

centre optique, coefficients de distorsion, vecteurs de translation et rotation) permettant ainsi d'obtenir la matrice caméra qui est la matrice de passage du repère globale à celui de la caméra (OpenCV). Par exemple, dans le repère caméra gauche $R_c = (C_1, \vec{x_c}, \vec{y_c}, \vec{z_c})$, la matrice caméra s'écrit donc (OpenCV) :

$$\begin{bmatrix} f_{x_1} & 0 & C_{x_1} \\ 0 & f_{y_1} & C_{y_1} \\ 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

La calibration permet alors de résoudre les problèmes de distorsion dans l'image en connaissant les coefficient de distorsion :

$$\begin{pmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{pmatrix} \tag{1.3}$$

L'une des méthodes pour réaliser la calibration est l'utilisation d'un damier noir et blanc (voir OpenCV). Elle est appliquée pour chaque caméra indépendamment, puis simultanément.

Sur la figure 1.13, le point P1 peut représenter la projection sur le plan image gauche de n'importe quel point se situant sur la droite (D) après le plan. La recherche de son correspondant P2 sur le plan image droite sera donc fait sur la ligne épipolaire E2 en considérant les deux plans image rectifiés. D'autres paramètres (couleur, texture, luminosité...) vont permettre de déterminer la position optimale de P2 sur E2 (Zhou *et al.* (2020)). Cependant trouver des correspondances est une tache difficile : les deux images présentes des différences causées par les variations de texture, d'occlusion (région présente sur une seule des deux images) ou photométriques (lumière, d'intensité, paramètres de caméra).
La carte de disparité est une information importante pour de nombreuse applications en vision par ordinateur. Sa qualité et les performances de la méthode utilisée (temps de traitement, ressources) sont des paramètres cruciaux pour une estimation précise de la profondeur.

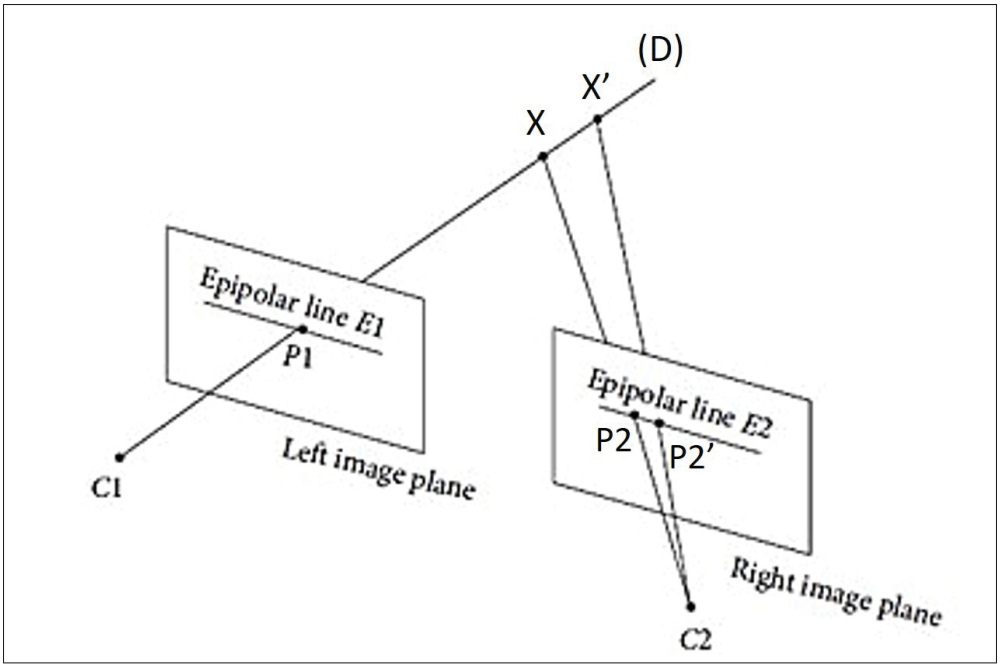

FIGURE 1.13 Illustration de la géométrie épipolaire*
Tirée de Zhou *et al.* (2020)

*Le point P1 peut être issu de n'importe quel point X de la scène sur la droite (D). Et il peut apparaître à n'importe quelle position P2 de la ligne epipolaire E2 de l'image droite.

1.2 Méthodes de Détection 3D par Stéréo-Image

Les méthodes en apprentissage profond de détection d'objet 3D (DO3D) par stéréo sont alimentées par deux images de la même scène et produisent en sortie l'image principale (image gauche en général) avec les classes et boites 3D des objets détectés. La classe d'objet *Voiture* est celle considérée pour la grande majorité des méthodes. L'utilisation d'image stéréo permet d'ajouter des caractéristiques de texture, couleur, et de forme, permettant d'extraire une représentation de la profondeur plus dense que les données lidar. Les informations de profondeur sont obtenues grâce au principe de *stereo-matching*. La figure 1.14 décompose les différentes étapes des méthodes de détection 3D par stéréo et présente les techniques utilisées dans la littérature à chaque étape. En général, les images stéréo sont d'abord traitées par un réseau d'extraction des caractéristiques. Puis la carte générée est transformée en une représentation 3D de la scène. Et à partir de ces informations 3D, la détection est réalisée. En supplément de

l'apprentissage supervisé de la détection 3D, il peut y avoir une supervision de la représentation 3D intermédiaire mais aussi l'ajout d'informations supplémentaire de la scène (nuage de points lidar, BEV, image à t+1...).

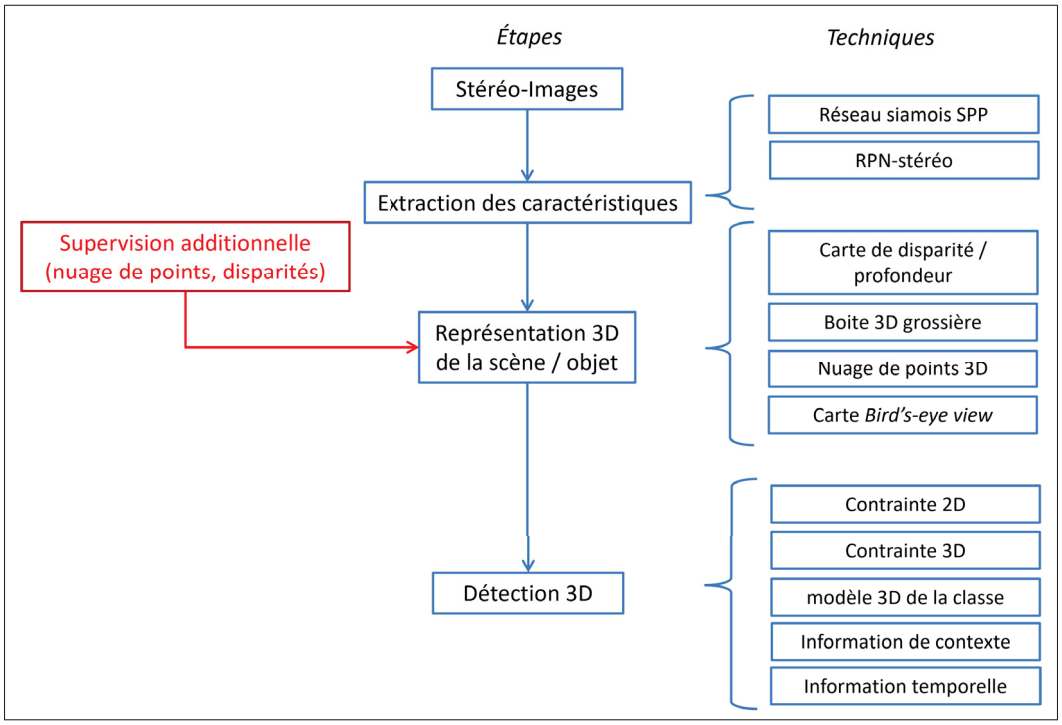

FIGURE 1.14 Principe des méthodes de DO3D*

*Il est indiqué les étapes et les techniques employées dans la littérature. Certaine approche applique une supervision supplémentaire à celle de détection.

Les méthodes sont classifiées dans quatre catégories principales : les méthodes basées sur de la détection 2D, les méthodes basées sur un réseau de proposition de régions (RPN), Les méthodes basées sur un réseau d'estimation des disparités et les méthodes hybrides. Les parties suivantes présentes les quatre familles DO3D par stéréo.

1.2.1 Méthodes basées sur de la détection 2D

De nombreux travaux ont été menés pour la détection 2D. Les avancées réalisées pour cette tâche ont permis aujourd'hui d'atteindre de très grandes performances contrairement aux méthodes de détection 3D. Ainsi, il semble intuitif d'étendre ces travaux pour la détection 3D. Les méthodes de cette catégorie utilisent un réseau état de l'art de détection 2D (et de segmentation dans certain cas) associé à un algorithme de *stéréo-matching* pour obtenir les informations de profondeur avec l'ajout de post-traitement pour affiner la prédiction. La figure 1.15 en présente le principe.

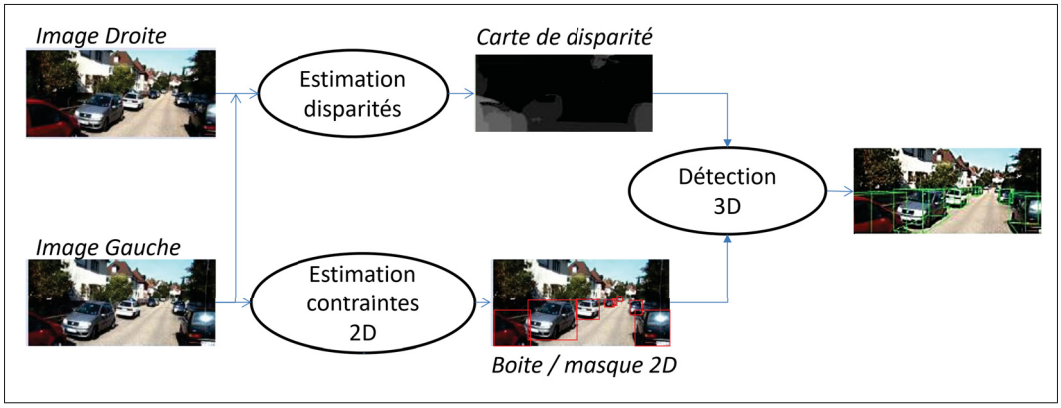

FIGURE 1.15 DO3D par stéréo basées sur la détection 2D*

*Les contraintes de profondeur issues de la carte de disparité avec les contraintes 2D de forme permettent d'effectuer la détection 3D.

H.Konigshof, Salscheider & al. (2019) proposent le RT3DStereo qui, à partir des images stéréo en entrée, réalise la segmentation et détection 2D sur l'image gauche. En parallèle, Un algorithme de *block-matching* (association de block similaire) en temps-réel est appliqué pour estimer les disparités. Un algorithme de *clustering* va ensuite regrouper les points de même classe sémantique et de disparité pour obtenir des propositions d'objets 3D. Ces propositions sont projetées dans un plan en vue d'oiseau BEV (*Bird-eye-view*), et leurs orientations et dimensions sont corrigées à l'aide d'un modèle préétabli de la classes-objet (créé à partir des valeurs moyennes et déviations standard des dimensions des objets de la classe). Un score de confiance permet de filtrer les détections incorrectes. Le RT3DStereo a la particularité de fonctionner en temps réel, ce qui est rare pour cette tâche. Cela est dû à la vitesse de l'algorithme de

stereo-matching. Cependant, il n'est pas entraînable *end-to-end* (le réseau de segmentation est entrainé indépendamment), et il possède une faible précision.
Li, Qin & al. (2018) exploitent de leur côté l'information temporelle pour affiner la détection 3D avec le réseau 3DOEM. A partir des images stéréo, le modèle prédit la boite 2D et l'orientation de l'objet pour estimer sa boite 3D grossière et son masque. Des algorithmes de *stereo-matching* et *temporal-matching* sont ensuite appliqués pour exploiter les informations de profondeur et temporelle et affiner la boite 3D précédemment estimée. En supplément, la trajectoire 3D des objets est prédite en rajoutant l'estimation du mouvement de la caméra. Cette méthode est l'une des seules en stéréo à bénéficier de l'information temporelle. Cependant elle n'est pas entraînable *end-to-end* et l'algorithme encodant l'information temporelle ne permet pas un fonctionnement en temps-réel. De plus, il demande beaucoup de ressource en calcul et mémoire de par sa complexité.

1.2.2 Méthodes basées sur un réseau de proposition de régions

D'autres approches s'inspirent des méthodes de détection et classification d'objets 2D en deux étapes utilisant un réseau de proposition de régions d'intérêt (RPN) tel que le Faster RCNN de Ren *et al.* (2015) (voir figure 1.16). Cette approche permet au réseau de se concentrer seulement sur les zones d'intérêt lors de la détection. On remarque que les méthodes utilisant un RPN sont plus précises mais possèdent un très faible rappel (des objets de la scène sont non-détectés) ce qui n'est pas permis pour certaine application en vision.
Dans le cas de la DO3D par stéréo, le RPN utilise les informations contenues dans l'image stéréoscopique afin de produire les propositions d'objet. Le principe de cette approche est présenté par la figure 1.17. Inspiré du Faster RCNN, Li, Chen & al. (2019) ont développé le Stereo RCNN, dont le RPN concatène les représentations des images gauche et droite issues d'un réseau d'extraction à deux branches, et retourne le couple de régions d'intérêt (RoIs), les classes et boites 2D grossières des objets présumés. Le couple de RoIs est concaténé et fourni à deux branches du réseau prédisant les boites 2D précises, l'orientation de l'objet, ses dimensions et les points clés (sommet) de la boite 3D. L'ensemble de ces données permet de reconstruire une boite

3D grossière par triangulation qui est affinée par alignement photométrique (minimisant l'erreur d'intensité des pixels). Ainsi, le réseau est supervisé au niveau du RPN sur la classe et boite 2D et au niveau de la prédiction des contraintes géométriques 3D, mais il n'est pas directement supervisé sur la position de l'objet dans l'espace. Les résultats de la méthode montre que le principe d'alignement photométrique n'est pas robuste aux occlusions, et que la contrainte de points-clé est difficile à généraliser à d'autres classes telles que *Piéton* ou *Cycliste*.

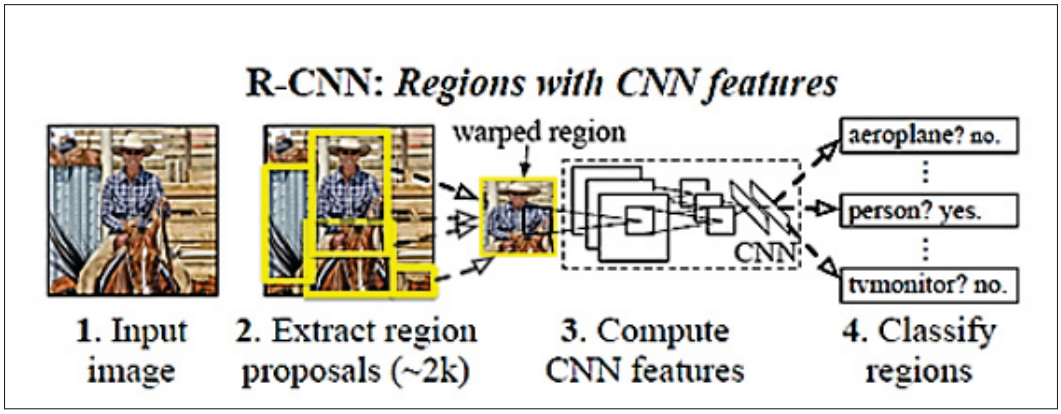

FIGURE 1.16 Illustration du réseau en 2 étapes R-CNN*
Tirée de Girshick *et al.* (2014)

*Combine un RPN qui prédit les régions pouvant contenir un objet à partir de l'image entrée, et un réseau qui réalise l'extraction des caractéristiques des régions, la détection et classification des objets.

Chen *et al.* (2018b) proposent le 3DOP, une méthode dont le RPN génère des RoIs sous forme de boites 3D positionnées dans l'espace, encodant la relation entre les images stéréo. Elles sont obtenues en minimisant une fonction énergie encodant plusieurs informations de profondeur : le nuage de points issu de la carte de disparité elle même obtenue par un algorithme de *block-matching*, la densité de ce nuage, l'espace libre, l'étendue du sol (permet de réduire l'espace de recherche) et les dimensions préétablies des classe-objets. Un algorithme de Non-Max-Suppression (NMS) est appliqué pour supprimer les boites indiquant une même région. Puis les propositions 3D sont fournies au réseau de détection qui les projette sur la carte de caractéristique de l'image principale. L'information du contexte est exploitée pour prédire la boite 3D, l'orientation de l'objet et sa classe. Cette méthode nécessite de générer beaucoup

de propositions 3D pour avoir un bon rappel. De plus, la fonction énergie est dépendante de la proposition et chaque proposition est dépendante de la classe. Il faut donc calculer la fonction énergie pour chaque proposition et le modèle doit être exécuté séparément pour chaque classe-objet. Cette méthode demande un temps de traitement et un coût en calcul importants. Pour répondre à ce problème, Pham & Jeon (2017) ont développé le DeepStereoOP, une version modifiée du 3DOP, dont les propositions 3D sont reclassées entre elles et indépendantes de la classe-objet.

D'une manière similaire, Qin, Wang & al. (2019) proposent le TLNet qui, à partir de propositions de boite 3D générées dans tout l'espace, va sélectionner les candidats de haute probabilité à l'aide d'une *objectness map* et les projeter sur les cartes de caractéristique des images stéréo pour obtenir les paires de RoIs. Les RoIs sont affinées par un score de cohérence puis fusionnées, afin d'estimer la boite 3D ainsi que la probabilité d'avoir un objet (*objectness*). Bien que les propositions 3D permettent de concentrer le modèle sur les RoIs, sa faible résolution ne lui permet pas d'exploiter entièrement la correspondance entre les pixels, rendant la représentation 3D de l'objet grossière.

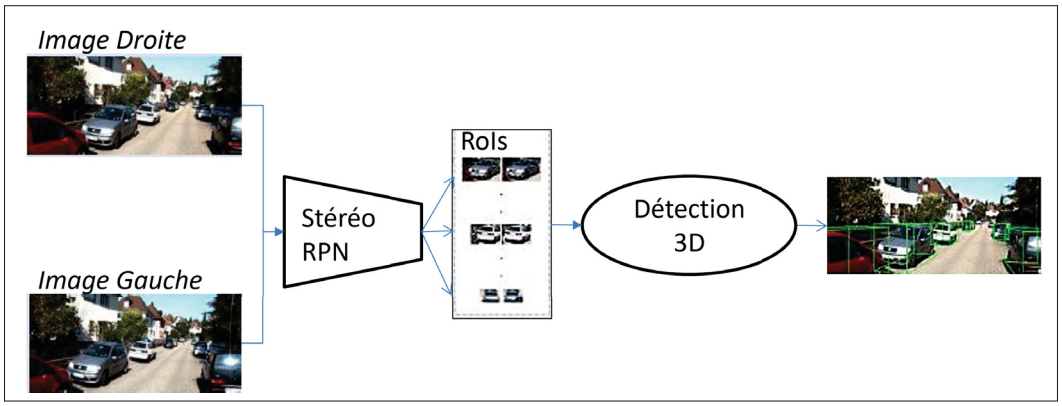

FIGURE 1.17 DO3D par stéréo basées sur un RPN*

*Le stéréo RPN propose des couples de RoIs riche en information 3D. L'estimation de contrainte complémentaire (classe, orientation, points clé...) vont aider à la détection 3D.

1.2.3 Méthodes basées sur un réseau d'estimation des disparités

Par la suite, les travaux ont cherché à répondre au problème en associant deux sous-tâches : l'estimation de profondeur ou de disparité, et la détection 3D. Un réseau de neurones est utilisé pour prédire soit la carte de disparité ou profondeur, soit la BEV. Puis un nuage de points 3D est généré et fourni au réseau de détection 3D. La détection est généralement réalisée par des méthodes de DO3D par lidar, car elles assurent les meilleures performances. L'approche générale est présentée par la figure 1.18.

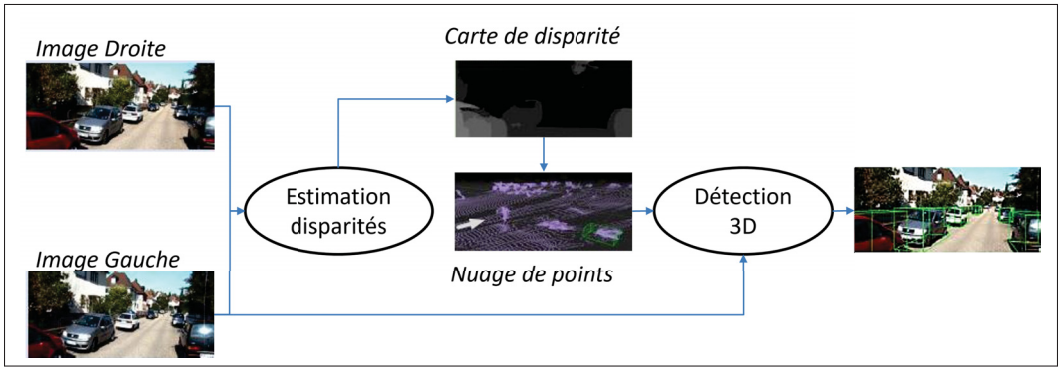

FIGURE 1.18 DO3D par stéréo basées sur l'estimation des disparités*

*L'information de profondeur est utilisée pour reproduire la scène 3D pour la détection.

Wang *et al.* (2019) suggèrent que les faibles performances des méthodes de DO3D par stéréo par rapport aux méthodes basées sur le lidar sont dues à la représentation 3D de la scène qui est pauvre en information sur la structure de l'espace occupé. Ils décident de proposer le Pseudo-Lidar, réseau de neurones divisé en deux parties : un réseau état de l'art, le PSMNet (Chang & Chen (2018)), estime la carte de disparité à partir des images stéréo dont les caractéristiques ont été extraites par un module SPP (*Spatial Pyramide Pooling*) (présenté à la figure 2.6). Puis cette carte est convertie en un nuage de points à l'aide d'une matrice de calibration. Les auteurs exploitent le fait que le nuage de points issu d'un réseau de disparité est d'une qualité proche à celui issu d'un lidar. Une méthode fusion (voir tableau 1.1) de détection 3D état de l'art (AVOD-FPN de Ku, Mozifian, Lee, Harakeh & Waslander (2018) ou SA-PV-RCNN de Bhattacharyya, Huang & Czarnecki (2021)), est ensuite appliquée sur le nuage de points.

Cette approche permet de combiner deux architectures état de l'art pouvant être pré-entraînées indépendamment mais pas d'une manière *end-to-end*. Le passage de la disparité au nuage de points n'est pas différentiable, mais cela fournit une information supplémentaire de densité qui permet une meilleure compréhension de la scène 3D par le réseau.

Cependant, l'erreur de disparité augmentant quadratiquement avec la distance, on observe une pauvre précision sur la détection d'objets lointain. C'est pourquoi You, Wang & al. (2020) ont développé le Pseudo-Lidar++, qui convertit le volume coût de disparité estimé en un volume coût de profondeur pour prédire directement la carte de profondeur. Cela permet de considérer l'échelle de profondeur des pixels lors de la prédiction. Un nuage de points dense est créé à partir de la carte de profondeur puis la détection 3D est réalisée. Les auteurs proposent aussi une configuration semi-supervisée du pseudo-lidar++ où le nuage de points est corrigé par sa vérité terrain clairsemée (issue d'un appareil lidar bas prix).

Chen, Liu & al. (2020) proposent de leur coté le DSGN qui associe en une étape l'estimation de la profondeur et la détection 3D. Cela permet un apprentissage *end-to-end*. Pour réaliser une représentation précise et efficace de la scène observée, les auteurs ont décidé d'encoder la scène 3D issue des images stéréo, dans un espace de caractéristiques 3D. Un réseau basé sur le PSMNet construit un *Plan-Sweep Volume* (PSV) pour estimer la carte de profondeur. Le PSV est transformé en un volume 3D fidèle à la géométrie de la scène, puis des convolutions 3D sont appliquées pour extraire de riches informations 3D et estimer la carte BEV. En chaque position de la carte BEV, des boites 3D prédéfinies de différentes tailles et orientations sont construites, puis les meilleures sont sélectionnées. Le réseau fini par prédire pour chaque boite 3D sa position et la classe de l'objet détecté. Leur méthode modélise une représentation 3D riche en information, permettant ainsi une meilleure compréhension de la scène par le modèle. Cependant, l'utilisation d'outil 3D demande beaucoup en ressource computationnelle (temps et mémoire).

1.2.4 Méthodes hybrides

Le Pseudo-Lidar et le Stéréo RCNN ont servi de base pour les travaux récent, qui vont chercher à gagner en précision et efficacité en combinant plusieurs approches. Sun, Chen & al. (2020) ont ainsi proposé le DispRCNN qui utilise le RPN du Stéréo RCNN pour obtenir une paire de RoIs et prédire la boite 2D et le masque de chaque objet. Un réseau inspiré du PSMNet va ensuite estimer la carte de disparité seulement pour les RoIs. La carte de disparité à l'échelle de l'objet est ensuite convertie en un nuage de points, puis une méthode de DO3D par lidar est appliquée. Les vérités terrains à l'échelle de l'objet sont obtenues en projetant un modèle 3D préétabli de la classe-objet. Par cette approche, on prend en compte la forme irrégulière de l'objet. De plus, estimer les disparités seulement pour les régions d'intérêt permet de gagner en temps, en ressource de calcul et en mémoire. Mais cela nécessite d'établir des modèles de forme pour chaque classe considérée.
De la même manière, Xu & al. (2020) ont développé le ZoomNet, entrainable *end-to-end*, utilisant aussi une version du RPN du Stéréo RCNN pour estimer boite 2D, dimensions et classe. Les RoIs en sortie du RPN sont toutes redimensionnées à la même résolution, puis fournies à une version du PSMNet qui estime les disparités, et un encodeur-décodeur qui estime le masque et la position 3D de chaque pixel au sein de l'objet. Le nuage de points 3D est alors généré à partir des données précédemment prédites, permettant d'estimer la pose de l'objet (orientation et translation) ainsi qu'un score mesurant la qualité de la boite 3D. L'étape de redimensionnement permet d'augmenter la précision dans l'estimation de profondeur pour les objets de loin en assurant une meilleure reconstruction du nuage de points. L'estimation de la position des pixels au sein de l'objet introduit par les auteurs rend le modèle plus robuste aux occlusions mais nécessite la création d'une base de données annoté.

Les tableaux 1.2, 1.3 et 1.4 regroupent les méthodes de DO3D par stéréo vues dans la littérature et résument leurs contributions pour chaque étape amenant à la détection 3D.
Le tableau 1.2 présente les 2 principales approches employées pour l'extraction des caractéristiques. On peut voir ainsi qu'une bonne partie des méthodes utilise une entrée siamoise pour extraire, en parallèle et d'une manière hiérarchique, les informations des images gauches et

droite, pour ensuite concaténer les deux cartes produites. Le tableau 1.3 regroupe les techniques employées pour produire une représentation 3D de la scène à partir des caractéristiques extraites. La majorité des méthodes va chercher à produire la carte de disparité issue des correspondances entre les image gauche et droite. Mais on remarque que parmi ces méthodes, nombreuses sont celles générant en plus le nuage de points de la scène. Elles vont chercher à exploiter les relations d'espace entre les points de la scène, que l'on ne trouve pas dans une image. Finalement, le tableau 1.4 présente les techniques réalisant la détection mais aussi celles la renforcant. Ainsi, pour estimer la boite 3D la plus précise possible, les méthodes de DO3D par stéréo vont d'abord s'appuyer sur des contraintes 2D (boite 2D, masque) mais aussi 3D (orientation, pose...). Cependant, comme estimer une boite 3D précise à partir d'image est une tache compliqué, les méthodes de DO3D par stéréo vont ajouter un modèle 3D (CAD) de la classe objet pour aligner la boite prédite ainsi que le nuage de points à l'échelle de l'objet.

TABLEAU 1.2 Techniques pour l'extraction des caractéristiques en DO3D par stéréo : un point indique que la technique est utilisée par la méthode de détection 3D

	Extraction caractéristiques		
Méthodes	**Architecture de base**	**Stéréo-RPN**	**SPP siamois**
RT3DStereo	ResNet		
3DOEM	Faster R-CNN		
Stereo RCNN	ResNet	•	
TLN	Faster R-CNN		
3DOP	Fast R-CNN		
DeepStereoOP	Fast R-CNN		
Pseudo-Lidar	PSMNet		•
Pseudo-Lidar++	PSMNet		•
DispRCNN	ResNet / PSMNet	•	•
ZoomNet	ResNet / PSMNet	•	•
DSGN	PSMNet		•

TABLEAU 1.3 Techniques pour encoder l'information 3D en DO3D par stéréo

	Représentation 3D			
Méthodes	**Disparité**	**Profondeur**	**Nuage de points**	**Boite 3D grossière**
RT3DStereo	•			
3DOEM	•			
Stereo RCNN				
TLN				•
3DOP	•		•	•
DeepStereoOP	•		•	•
Pseudo-Lidar	•		•	
Pseudo-Lidar++		•	•	
DispRCNN	•		•	
ZoomNet	•		•	
DSGN		•		

TABLEAU 1.4 Techniques pour réaliser et affiner la détection d'objet en DO3D par stéréo. Les contraintes 2D sont par exemple la boite 2D, le masque de l'objet. Les contraintes 3D peuvent être l'orientation, la pose, des *keypoints* de l'objet

	Détection 3D				
Méthodes	**Contrainte 2D**	**Contrainte 3D**	**Contrainte temporelle**	**Modèle 3D objet**	**DO3D par lidar**
RT3DStereo	•	•		•	
3DOEM	•	•	•		
Stereo RCNN	•	•			
TLN				•	
3DOP		•		•	
DeepStereoOP		•		•	
Pseudo-Lidar					•
Pseudo-Lidar++					•
DispRCNN	•			•	•
ZoomNet	•	•		•	
DSGN		•		•	

Les points précédemment observés vont pousser à nous concentrer par la suite sur les méthodes de la littérature pour l'estimation des disparités par image stéréo. Nous allons aussi nous intéresser a la technique SPP pour l'extraction des caractéristiques, et à la méthode de génération

du nuages de points à partir des disparités pour la détection. Mais avant cela, nous allons d'abord présenter la base de donnée référence KITTI, ainsi que les outils d'évaluation pour la détection.

1.2.5 Base de données

KITTI Détection d'objet. La base de données de DO3D KITTI (Geiger, Lenz & Urtasun (2012)) contient 7 481 pairs d'images de trafic routier pour l'apprentissage, et 7 518 pairs pour le test. Les scènes couvertes par la base sont des autoroutes, des zones urbaines complexes et des routes étroites de campagne. Le set d'apprentissage est divisé pour l'entraînement et la validation comme proposé par Chen *et al.* (2015) (soit 3 712 et 3 769 respectivement). Pour chaque image d'apprentissage, KITTI fournit le nuage de points lidar, les matrices de calibration des caméras et les labels des boites 3D pour différentes catégories d'objets routiers. Chaque catégorie est divisée en 3 niveaux de difficulté - facile, modéré et difficile - qui dépend de la hauteur de la boite 3D, son degré d'occlusion et de troncature. Les labels de difficulté "facile" correspondent en général aux objets à moins de 30 mètres de la caméra. Les méthodes en DO3D par stéréo vont surtout s'intéresser aux classes "voiture", "pieton", et "cycliste".
A noter, il existe aussi la classe "DontCare" qui correspond aux régions où les objets n'ont pas été labélisé et dont les résultats de détection ne seront pas considérés. Les images de la figure 1.10 sont issues de la base KITTI.

1.2.6 Outils d'évaluation

Pour évaluer et comparer les méthodes de DO3D par stéréo, les travaux de la littérature utilisent la vitesse de détection en image par seconde (FPS ou *Frame Per Second*) et la précision moyenne AP (*Average Precision* en %) avec des détections au-dessus d'un certain seuil de IoU (*Intersection over Union*).
La précision moyenne de détection AP évalue les résultats de classification et correspond à l'air

sous la courbe de la précision P en fonction du rappel R (1.4) :

$$\begin{aligned} P &= \frac{TP}{TP+FP} \\ R &= \frac{TP}{TP+FN} \\ AP &= \int_0^1 P(R)\,dR \;\simeq\; \frac{1}{m}\sum_{i=1}^{m} P(R_i) \end{aligned} \tag{1.4}$$

où TP (*true positive*) est le nombre de vrai objet détecté, FP (*false positive*) le nombre de faux objet détecté et FN (*false negative*) le nombre d'objet non-détecté.
Ainsi, P mesure la précision du modèle sur l'ensemble des objets qu'il a détectés tandis que R mesure la précision sur l'ensemble des objets annotés dans la scène. De plus, AP va évaluer la précision pour une classe spécifique, et on utilise mAP (*mean* AP) pour mesurer les performances sur toutes les classes d'objet.

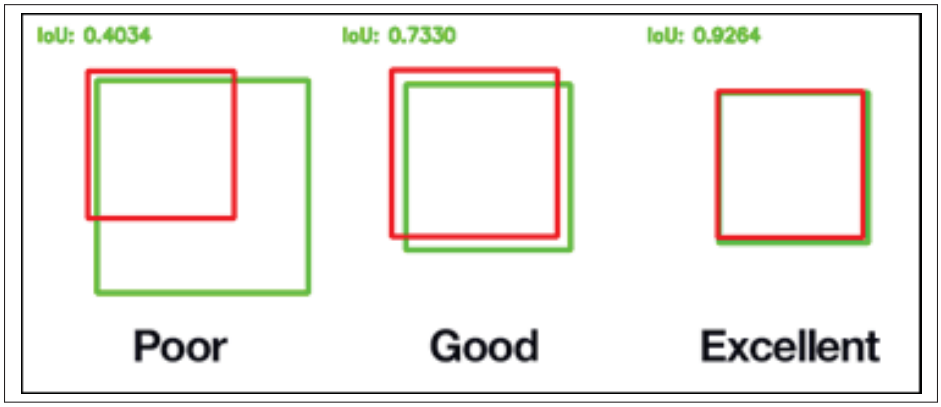

FIGURE 1.19 Plus la boite prédite (vert) se rapproche de la boite annotée (rouge), et plus la valeur IoU tend vers 1

Le ratio de chevauchement IoU va de son coté évaluer la qualité de la boite 3D prédite. Il est égale au rapport de l'intersection de la boite encadrante estimée b avec la boite réelle $\hat{b}$, sur leur union. Et il doit être supérieur à un seuil prédéfini ε :

$$IoU(b,\hat{b}) = \frac{b \bigcap \hat{b}}{b \bigcup \hat{b}} \geq \varepsilon \tag{1.5}$$

Plus la boite prédite est précise et plus son ratio IoU tend vers 1 (figure 1.19). Il existe différentes méthodes pour représenter la boite 3D : on peut considérer les huit sommets, ou quatre sommets avec 2 longueurs, ou bien estimer l'erreur de position du centre 3D et les dimensions de la boite. En général, une méthode de détection prédit pour chaque objet détecté j l'ensemble $\{(b_j, c_j, p_j)\}$ où b est sa boite encadrante, c sa classe et p sa probabilité (ou niveau de confiance). Le résultat de détection d'un objet est considéré comme TP (soit correct) si la catégorie prédite égale au label, et si le ratio IoU est supérieur au seuil prédéfini. Dans le cas contraire, la détection est considérée comme FP. Dans le cas de détection multiple d'un même objet, celle avec le niveau de confiance le plus grand est gardée (méthode NMS).
Les méthodes en DO3D par stéréo mesure AP pour la détection 3D (AP_{3D}) et pour la localisation par BEV (AP_{BEV}) en considérant un seuil ε de 0,7 pour la classe "voiture" et 0,5 pour les autres.

1.3 Méthodes d'estimation des Disparités par Stéréo-Image

1.3.1 Définition

L'estimation des disparités est un problème majeur en vision par ordinateur présent dans de nombreuses applications telles que la robotique, l'odométrie ou la conduite autonome. Elle permet de fournir une information dense de profondeur à partir d'images RGB. Les méthodes traditionnelles appliquent des algorithmes de *stereo-matching* dont les paramètres sont choisis par des experts. Elles se décomposent en plusieurs étapes : calcul du volume coût de *stereo-matching*, agrégation de ce volume, optimisation des disparités et post-traitement (Zhou *et al.* (2020)).
La première étape consiste à calculer des mesures de similarité entre des patchs de pixels de l'image gauche et droite, créant un volume coût. Les caractéristiques bas-niveau des patchs d'images vont servir à mesurer la dissimilarité autour d'un pixel (Zhou *et al.* (2020)). L'étape d'agrégation introduit des informations contextuelles d'appariement des pixels (similitude et dissimilitude) tandis que l'étape d'optimisation permet la régularisation du volume de disparités, afin de régresser sur une prédiction plus lisse. Des techniques de post-traitement (filtre, lissage, cohérence gauche-droite ...) vont finalement servir à affiner les cartes de disparités.

Les méthodes traditionnelles de *stereo-matching* peuvent être regroupées en deux grandes familles : les méthodes locales et les méthodes globales (Zhou *et al.* (2020)). Les méthodes locales calculent la disparité en un pixel donné en cherchant le plus proches voisins (dans un patch d'image) ayant le coût le plus faible. Ces méthodes sont rapides mais ont une très faible précision surtout dans les zones sans textures et répétitives. Les méthodes globales vont plutôt chercher à résoudre un problème d'optimisation sur la carte de disparité entière en minimisant une fonction énergie. Ces méthodes assurent des prédictions de bonne qualité mais sont gourmandes en temps. On observe que les performances des méthodes traditionnelles sont fortement limitées par leurs paramètres conçus à la main.

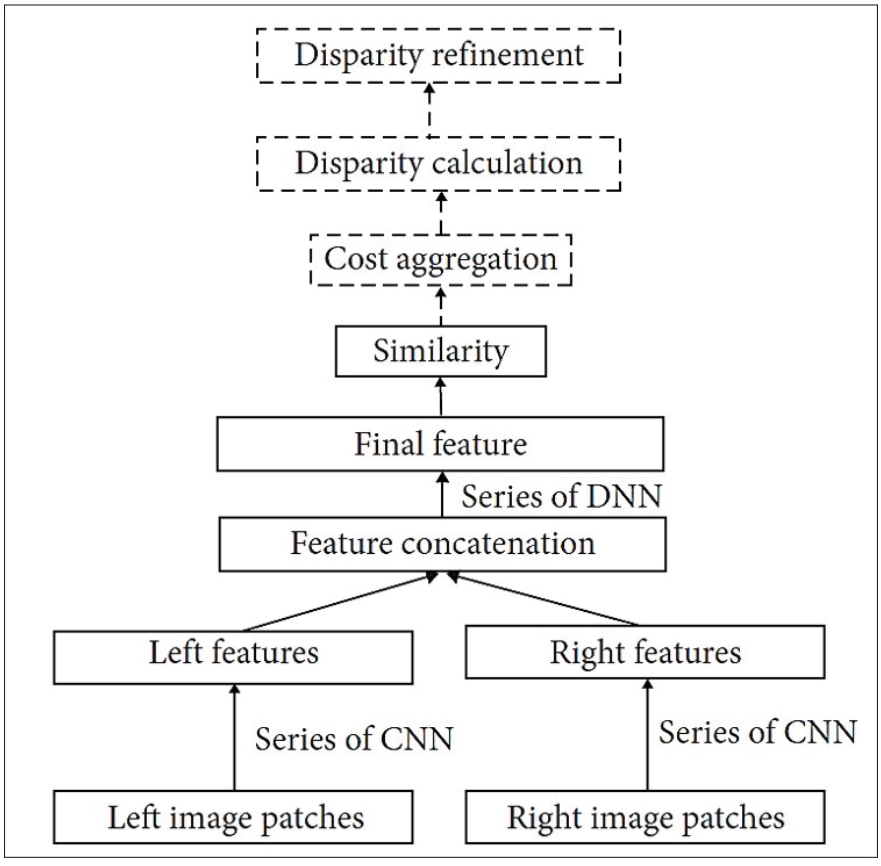

FIGURE 1.20 Structure d'un réseau Siamois pour l'estimation des disparités non *end-to-end* : Estimation des similarités entre deux patchs d'images
Tirée de Zhou *et al.* (2020)

L'émergence d'algorithmes en apprentissage profond a permis de lever ces limitations en proposant des méthodes répondant au problème d'une manière *end-to-end*. Les premières approches en apprentissage profond pour l'estimation des disparités n'étaient pas *end-to-end* mais cherchaient à remplacer une ou plusieurs étapes des méthodes traditionnelles par des réseaux de convolution (CNN). Les premiers à l'introduire sont Zbontar & LeCun (2015), qui ont substitué les étapes de pré-traitment et de mesure de similarité par un CNN siamois (voir figure 1.20). Bien qu'on observe un gain en performance par rapport aux méthodes traditionnelles, les approches *non-end-to-end* ont un champ réceptif limité et nécessite toujours un post-traitment.

Les méthodes *end-to-end* pour l'estimation des disparités peuvent être divisées en deux groupes (Laga (2019)). D'un coté les méthodes utilisant une architecture encodeur-décodeur avec seulement des couches de convolution 2D, et de l'autre les méthodes imitant le principe des approches traditionnelles tout en intégrant des convolutions 3D pour l'étape d'agrégation. Les couches de convolution 2D vont opérer sur des cartes de caractéristiques à 3 dimensions ($H \times W \times C$), tandis que les convolutions 3D traitent des volumes 4D (on ajoute ici la disparité comme quatrième dimension). De plus, certaines méthodes vont chercher à prédire la carte de disparité tandis que d'autres vont directement prédire la carte de profondeur :

- **estimation des disparités :** cette tâche est réalisée par des modèles prenant en entrée une image stéréo, en générale rectifiée. Le but est donc d'estimer à partir de la paire d'image la carte de disparité où chaque pixel est la valeur de disparité estimée entre le pixel dans l'image gauche à la même position, et le pixel droit correspondant. Les cartes de disparité sont par la suite converties en carte de profondeur ou nuage de points (voir figure 1.12) pour être exploitées ;
- **estimation de la profondeur :** cette tâche est généralement réalisée par un modèle monoculaire. L'objectif est d'estimer, à partir d'une image en entrée, la carte de profondeur où chaque valeur de pixel est sa distance absolue au système de caméra.

1.3.2 Méthodes 2D

Les méthodes 2D exploitent une structure encodeur-décodeur U-Net (Ronneberger, Fischer & Brox (2015)) pour effectuer une régression sur les disparités. Mayer & Ilg (2016) sont les premiers à proposer un modèle *end-to-end*, le DispNet, pour l'estimation des disparités. La structure siamoise de l'encodeur permet d'extraire les caractéristiques des images droite et gauche en parallèle, puis une couche de corrélation calcul le coût d'appariement par un produit scalaire. Le décodeur multi-échelle réalise l'agrégation et l'affinage en produisant des cartes de disparité sous différentes résolutions. La structure multi-échelle apporte du détail et des informations résiduelles pour la prédiction finale.

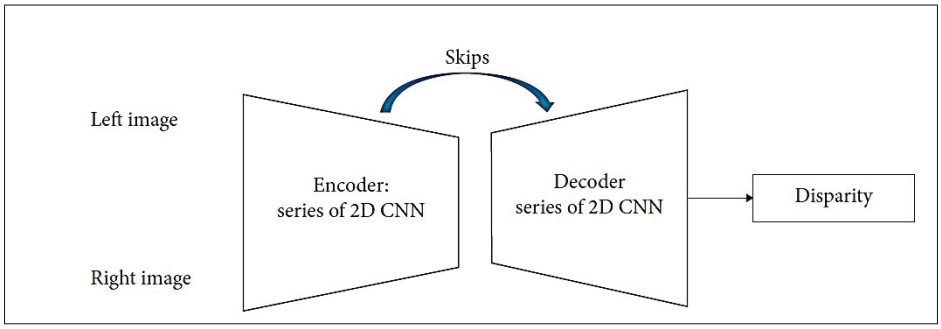

FIGURE 1.21 Structure d'un réseau encoder-decoder 2D pour l'estimation des disparités end-to-end
Tirée de Zhou *et al.* (2020)

Les approches suivantes vont s'inspirer du DispNet en proposant des configurations améliorant ses performances. Les méthodes CRL (Cascade Residual Network) de Pang, Sun, Ren, Yang & Yan (2017) et iResNet de Liang, Feng, Guo & al. (2018) proposent des versions du DispNet à deux étages assez similaires. Le premier étage réalise la prédiction des disparités tandis que le second produit des informations résiduelles pour le raffinage de la prédiction. Sur le même principe, Wang, Shi, Zheng, Zhao & Chu (2020) ont développé le FADNet qui ajoute des blocs résiduels et modifie la couche de corrélation du réseau principal de prédiction.
Tout récemment, trois méthodes exploitant des techniques 2D et assurant de très grande performance sur la base de données KITTI ont été proposées. Tankovich *et al.* (2021) proposent

le HITNet qui, à partir de caractéristiques multi-échelle issues d'un réseau en structure U-Net, initialise une carte de disparité multi-résolution qui est ensuite raffinée par des techniques de déformation d'image (*image warping*) et de propagation spatiale. Yee & Chakrabarti (2020) ont développé le Fast DS-CS, une méthode qui va d'abord construire le volume coût grâce à des techniques de *stereo-matching* traditionnelle, puis un encodeur-décodeur estime les disparités à partir d'une projection de ce volume. De leur coté, Xu & Zhang (2020) s'inspirent des méthodes 3D en remplaçant le rôle des convolutions 3D par des techniques 2D : ils proposent le réseau AANet qui produit plusieurs volumes coûts de corrélation à différentes échelles, puis effectue l'agrégation de ces volumes à travers l'échelle et entre les échelles. Cette double agrégation permet d'améliorer les performances face aux problèmes de discontinuité au niveau des bords et de régions sans textures.
Globalement, les méthodes 2D assurent une très bonne efficacité computationnelle mais ont en contrepartie une précision limitée sur les régions lointaines, sans textures et répétitives.

1.3.3 Méthodes 3D

Nombreuse sont les méthodes qui se sont intéressées à l'utilisation de convolution 3D pour estimer des cartes de disparités de hautes qualités. Kendall *et al.* (2017) ont ouvert la voie en proposant le GC-Net. Un volume coût d'appariement 4D ($H \times W \times C \times D$) est généré en concaténant les cartes de caractéristiques ($H \times W \times C$) des images stéréo selon l'espace de disparité D. Puis des convolutions 3D réalisent l'agrégation et le raffinement pour donner la prédiction. L'exploitation de la dimension de disparité et de convolution 3D permettent d'apprendre des informations spatiale et de disparité, mais aussi d'échelle évitant des erreurs et bruit.
Cette approche a inspiré Chang & Chen (2018), qui ont proposé le PSMNet. Leur modèle exploite un module SPP *(spatial pyramid pooling*) pour extraire les informations globales et locales, et un CNN 3D en sablier multi-échelle pour agréger et affiner les informations afin de prédire la carte de disparité. Le PSMNet va servir de modèle de base pour les méthodes suivantes. Cheng, Wang & Yang (2019) proposent le CSPN modifiant le 3D CNN du PSMNet

par un 3D SPN (*spatial propagation network*) assurant un gain en précision mais requérant un temps de prédiction important.

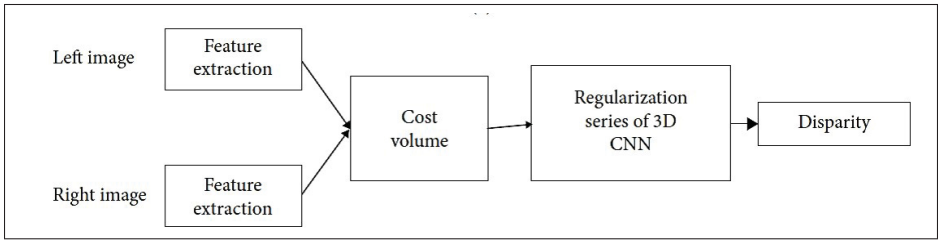

FIGURE 1.22 Structure d'un réseau avec convolutions 3D pour l'estimation des disparités end-to-end
Tirée de Zhou *et al.* (2020)

AcfNet de Zhang *et al.* (2020) améliore le PSMNet en réalisant une supervision du volume coût par un filtrage unimodale sur la distribution de chaque pixel. Cela permet ainsi d'améliorer le processus d'appariement dans les régions répétitives et sans texture. Le Deep Pruner proposé par Duggal, Wang, Ma, Hu & Urtasun (2019) ajoutent des modules de PatchMatch différentiable pour le calcul du volume coût, permettant de diminuer l'espace de recherche lors de l'appariement. D'autres méthodes comme le CSN (Gu *et al.* (2020)) traite le volume coût en cascade d'une manière *coarse-to-fine* (passer de grossier à fin) en créant plusieurs résolutions de ce volume puis en ajoutant un module d'agrégation et affinage.

Les méthodes 3D sont plus précises assurant des résultats état de l'art, mais des erreurs persistent dans les zones sans textures, au niveau des objets petits et lointains. De plus, leur complexité cubique et leur forte consommation en mémoire empêchent leurs déploiements dans des applications réelles.

1.3.4 Base de données

Bases de données synthétiques

SCENEFLOW. C'est une large base de données collectée par Mayer *et al.* (2016), contenant des images stéréo synthétiques avec leurs cartes de disparité densément annotées. Elle contient 35454 images pour l'apprentissage et 4370 images de test avec annotation, de résolution 960×540. Elle est composée de 3 sous-ensembles : Driving(+4000 images de scène urbaine du point de vue conducteur), Monkaa (+8000 images issues d'un film, contenant différents mouvement articulés) et FlyingThings3D (+20000 images d'objets du quotidien, volant suivant une trajectoire aléatoire). Les images de la figure 1.11 sont issues de FlyingThings3D.

vKITTI. Proposée par Cabon, Murray & Humenberger (2020), elle regroupe des données synthétiques très réalistes dont les scènes et points d'observation sont similaires à l'environnement KITTI (Menze *et al.* (2015)), en utilisant des modèles 3D et un simulateur de jeu vidéo. Elle possède 21,260 images stéréo avec les cartes de profondeur associées. Les images représentent différentes scènes urbaines du point de vue conducteur, avec des variations des conditions météorologique, de luminosité et d'orientation du points de vue. De plus, la base fournit des annotations de détection 2D et 3D, de segmentation et de suivi d'objets.

Bases de données réelles

KITTI. C'est une base de données réelle d'images stéréo avec cartes de disparité générées par lidar, représentant des scènes urbaines du trafic routier du point de vue conducteur. Il existe deux versions de la base : KITTI2012 (Geiger *et al.* (2012)) et KITTI2015 (Menze *et al.* (2015)). Leurs images sont de résolutions 1240×376 et KITTI2015 possède des cartes de disparité plus denses. KITTI2015 contient 200 images pour l'apprentissage (150 pour l'entraînement, 50 pour la validation), et 200 images pour le test sans annotation. KITTI2012 contient 194 images pour l'apprentissage (146 pour l'entraînement, 48 pour la validation), et 195 images pour le test sans annotation.

MIDDLEBURY (MB). Générée par Scharstein *et al.* (2014), elle rassemble 15 images stéréo (de résolution (1920×1080)) avec cartes de disparité pour l'apprentissage et 15 images pour le test sans annotation, contenant des scènes d'objets d'intérieur.

ETH3D. Elle est issue de Schöps *et al.* (2017) et fournit 27 (de résolution (752×480)) images stéréo avec cartes de disparité pour l'apprentissage et 20 images stéréo pour le test de scènes d'intérieure et d'extérieure.

1.3.5 Outils d'évaluation

Les performances des méthodes de disparité sont évaluées par les mesures :

- EPE (End-Point Error) : mesure l'erreur de disparité moyenne par pixel entre la prédiction y_D et la vérité terrain $\hat{y_D}$;

$$EPE(y_D, \hat{y_D}) = \| y_D - \hat{y_D} \| \tag{1.6}$$

- D1 : proposée par Menze *et al.* (2015), elle correspond au pourcentage de pixels pour lequel l'erreur de disparité est plus grand que 3 pixels et que ε_d% de la vérité terrain en ce pixel.

$$D1(y_D, \hat{y_D}) = \| y_D - \hat{y_D} \| > max(3px; \varepsilon_d \hat{y_D}) \tag{1.7}$$

Plus ces mesures sont basses, meilleurs sont les résultats. A noter, on utilise en général D1-all, qui permet de mesurer l'erreur pour tous les pixels de la vérité terrain (contrairement à D1-noc mesurant l'erreur seulement pour les régions non occlues). EPE est employée pour les bases synthétiques tandis que D1 est employée pour les bases réelles avec : KITTI($\varepsilon_d = 5\%$) ; ETH3D($\varepsilon_d = 1\%$) ; MB($\varepsilon_d = 2\%$).

1.4 Adaptation de domaine non-supervisée pour la détection et la profondeur

1.4.1 Définition

La construction d'un modèle en apprentissage profond nécessite l'utilisation d'une base de données pour son apprentissage. En général, des bases de données publiques annotées sont disponibles pour répondre à divers problèmes en vision par ordinateur. Certaines de ces bases sont devenues des références pour comparer les performances entre modèles, du fait de leur grande taille (allant jusqu'à plus de 200k images labélisées pour la base de données COCO de Lin *et al.* (2014)) et de leur forte diversité (pouvant avoir plus de 80 catégories d'objets ; différentes échelles, orientations, couleurs d'un même objet). Les méthodes états de l'art vont être supervisées sur ces grandes bases de données annotées. Cependant, les bases de données publiques ne peuvent répondre à toutes les applications et couvrir toutes les situations (Zhao *et al.* (2022)).

L'une des façons pour répondre à ce problème est de générer une base de données privée beaucoup plus représentative de la situation réelle. L'annotation d'images étant un travail fastidieux dans plusieurs applications réelles, les bases privées possèdent généralement une quantité limitée d'images annotées. Le modèle est alors entraîné et pré-évalué sur une large base publique, puis la base privée va servir à tester le modèle et l'affiner quand assez de données labélisées sont disponibles. Cependant, on observe une chute des performances sur les données terrain : le modèle ne peut généraliser correctement. Cela est dû à la différence de distribution des données entre les deux bases, empêchant le modèle de transférer les connaissances apprise sur le domaine source (la base de données publique) vers le domaine cible (la base de données privée) (Tzeng, Hoffman, Saenko & Darrell (2017)).

Le principe d'adaptation de domaine a émergé pour résoudre le problème de changement de domaine. C'est un type de *transfert learning* où l'on considère un domaine source et un domaine cible ayant des distributions différentes (Wilson & Cook (2020)). Dans ce mémoire, nous allons nous intéresser à l'adaptation de domaine non-supervisée dont le but est de transférer les informations apprises par le modèle sur un domaine source labélisé, vers un domaine cible

non-labélisé en rendant les représentations similaires entre les domaines. Cela permet au modèle d'apprendre des caractéristiques riches à partir des données source et cible, d'éviter un risque de sur-apprentissage sur le domaine cible, mais aussi de rendre le modèle plus robuste (Li, Li, Luo, Wang & sun (2020)). Il existe aussi l'adaptation de domaine supervisée pour laquelle les labels cibles et sources sont disponibles, et l'adaptation de domaine semi-supervisée pour laquelle quelques labels cibles sont disponibles (Wilson & Cook (2020)).

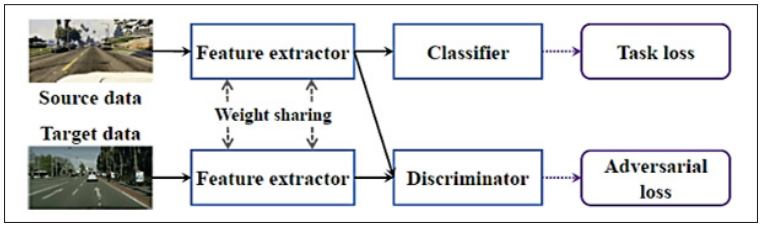

FIGURE 1.23 Illustration des méthodes *Adversarial Learning* pour la tache de classification
Tirée de Zhao *et al.* (2022)

Plusieurs techniques sont employées pour réaliser l'adaptation de domaine :

1. **Discrepancy.** l'écart entre les domaines est réduit afin de rendre les deux distributions similaires. La différence entre les domaines est mesurée en une couche du réseau par des méthodes comme MMD (Maximum Mean Discrepancy), CORAL (Correlation Alignment), ou EM (Entropy Minimisation) (Madadi, Seydi, Nasrollahi, Hossieni & Moeslund (2020)). Le modèle peut être ensuite affiné sur les données cible sans labels ou avec labels bruités ;
2. **Adversarial Learning.** le changement de domaine est minimisé en provoquant la confusion des domaines source et cible par une fonction cout adverse (voir figure 1.23). Goodfellow *et al.* (2014) ont proposé la fonction coût GAN pour l'apprentissage adverse (*adversarial learning*). Cette technique utilise un générateur (pouvant être le réseau d'extraction) fournissant une représentation pertinente de la donnée et d'un discriminateur de domaine classifiant la donnée selon son domaine d'appartenance. Le but du générateur est de fournir une représentation confuse de sorte que le discriminateur ne soit plus en mesure de différencier les domaines ;

3. **Image Reconstruction.** plutôt que de traiter les deux domaines, cette approche cherche à reconstruire le domaine source ou cible en utilisant un réseau de translation (voir figure 1.24). Les méthodes vont en général employer un GAN pour mapper un domaine vers un autre, tels que le GAN conditionnel (Isola, Zhu, Zhou & Efros (2017)) dans le cas d'images source et cible appariées, ou bien le CycleGAN (Zhu *et al.* (2017)) pour des images source et cible non appariées ;
4. **Hybrid.** combine les méthodes précédentes pour avoir de meilleures performances.

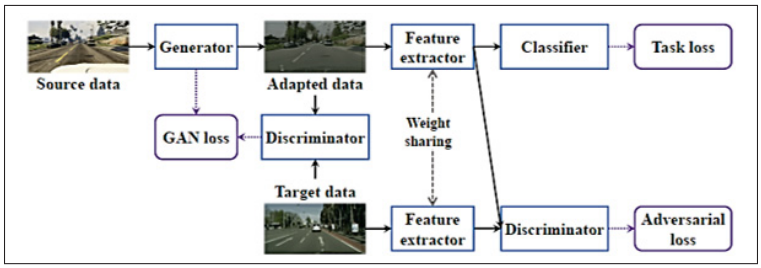

FIGURE 1.24 Illustration des méthodes GAN en *Image Reconstruction* pour la tache de classification
Tirée de Zhao *et al.* (2022)

Par la suite, nous allons passer en revu la littérature en adaptation de domaine non-supervisée pour les applications de détection d'objet et d'estimation de profondeur.

1.4.2 Détection d'objet

Les méthodes états de l'art en détection considèrent que les données source et cible ont la même distribution. Ce qui leurs permet d'exploiter de larges bases de données annotées. Mais dans le cas de changement de domaine, cette hypothèse entraîne de faible performance sur les données cible. L'annotation de boites 2D (ou 3D) pour chaque classe considérée pouvant être coûteuse, la détection d'objet par adaptation de domaine est apparue comme une solution pour améliorer les performances et la généralisation du modèle. Ces méthodes s'appuient en général sur des détecteurs existants assurant déjà d'excellentes performances (Li *et al.* (2020)).
Khodabandeh, Vahdat, Ranjbar & Macready (2019) proposent d'appliquer une approche de

discrepancy (divergence) sur le détecteur FasterRCNN (Ren *et al.* (2015)). Ils vont tout d'abord pré-entraîner le détecteur sur les données source pour ensuite le tester sur les données cible non-labélisées. Cela crée des données avec labels bruités qui sont corrigés par un réseau de classification apportant en informations sur les données cible. Ces nouvelles données avec les données source sont utilisées pour affiner le réseau et ainsi améliorer sa robustesse. Arruda *et al.* (2019) utilisent aussi le détecteur FasterRCNN mais avec une approche de reconstruction, pour détecter des objets 2D sur des images de nuit. Le CycleGAN proposé par Zhu *et al.* (2017) est employé comme réseau de translation pour reconstruire les images annotées de jour en image de nuit. Cette nouvelle base de données cible va servir à entraîner un détecteur d'images de nuit.
De leurs coté, Chen, Li, Sakaridis, Dai & Van Gool (2018c) appliquent une approche d'apprentissage adverse (*adversarial learning*) avec le détecteur FasterRCNN. Ils emploient une méthode minimisant la H-Divergence, qui mesure la capacité d'une classe H à discriminer entre les distributions source et cible. Deux discriminateurs de domaine sont utilisés : un au niveau de la représentation de l'image (dimension et style de l'image, luminosité. . .) et un au niveau de la représentation des régions d'intérêt détectées par le RPN (apparence, taille. . .). Shen, Maheshwari, Yao & Savvides (2019) s'inspirent de leurs travaux en ajoutant un ensemble de discriminateur de domaine en interactions à différents niveaux du réseau pour aligner les domaines.
D'autres auteurs vont chercher à renforcer le processus d'adaptation de domaine en proposant des méthodes d'adaptation hybride. Inoue, Yamasaki & Aizawa (2018) vont combiner les approches de reconstruction d'image (*image reconstruction*) et de *discrepancy* avec le détecteur SSD de Liu *et al.* (2016). Le CycleGAN translate les données source dans le domaine cible, ce qui permet par la suite d'entraîner le SSD sur ces « fausses » images cible. Le détecteur est ensuite appliqué sur les vraies images cible pour générer des pseudo labels qui seront affinés, pour améliorer la robustesse du réseau. D'une manière similaire, Rodriguez & Mikolajczyk (2019) appliquent les même approches sur le détecteur SSD, mais vont à la fois translater les images sources et les images cibles dans un style d'image aléatoire du domaine cible. Ils ajoutent une fonction coût de consistance intermédiaire pour assurer la conservation de l'information entre les caractéristiques extraites dans les images translatées et initiales.

1.4.3 Estimation des disparités ou de la profondeur

L'estimation des disparités et de la profondeur sont des applications dont l'obtention de données réelles annotées coûte cher, est fastidieuse et peut mener à des imprécisions ou du bruit. Ainsi, le peu de bases de données disponibles sont mal balancées, et représentent une même distribution de données, avec des conditions lumineuses, météorologiques et un cadre variant très peu. Des modèles entraînés sur ces bases de données deviennent alors biaisés et ne peuvent être déployés sur tout type de terrain sans prendre des précautions préalables. Pour contourner ce problème, des données synthétiques sont utilisées lors de l'apprentissage. Elles ont la particularité d'être facilement annotables avec précision, et d'être disponibles en grande quantité. Cependant, les modèles généralisent difficilement sur des données de scènes réelles à cause d'un changement de domaine trop important. D'un point de vue probabiliste, un réseau entraîné sur des données issues d'une source ayant une certaine distribution, ne peut performer très bien que sur des données test de même distribution. Pour permettre cela, l'approche générale est de transférer les représentations apprises du synthétique vers le réel en affinant le réseau sur des données mixes. Cet affinage nécessite d'adapter les millions de paramètres du réseau sur le nouveau domaine ce qui demande une large quantité de données cible. Des méthodes de la littérature se sont donc intéressées à appliquer le principe d'adaptation de domaine non-supervisée et semi-supervisée à l'estimation de profondeur ou des disparités (Laga (2019)).

Les méthodes de la littérature pour l'estimation de la profondeur non-supervisée sont en grande partie monoculaire. Les premières méthodes se sont d'abord intéressées à répondre au manque d'annotations des bases de données monoculaire en utilisant le principe de *discrepancy*. Garg, Kumar, Carneiro & Reid (2016) proposent préalablement de reconstruire l'image droite à partir de l'image gauche, pour ensuite appliquer un algorithme de *stereo-matching* qui va produire la carte de disparité puis de profondeur, qui sera utilisé comme vérité terrain pour entraîner un modèle monoculaire estimant la profondeur. Cela permet ainsi d'augmenter la quantité d'annotations disponibles des bases de données ne possédant pas d'images stéréo, grâce aux pseudo-labels. Godard, Aodha & Brostow (2017) améliorent ce principe de reconstruction en proposant une méthode entièrement différentielle (*end-to-end*), augmentant l'efficacité et

robustesse du modèle (voir figure 1.25). L'idée de leur réseau monoculaire est d'estimer les cartes de disparités pour l'image droite et l'image gauche. Une fonction coût va alors s'assurer de la cohérence entre la carte de disparité gauche et la carte de disparité droite projetée selon l'image gauche (*image warping*). Ces deux méthodes nécessitent une calibration précise de la caméra stéréo, pour que ses paramètres intrinsèque et extrinsèque soient les plus pertinent possible. Cette contrainte est non-négligeable car elle rend difficile le croisement de base de données différentes.

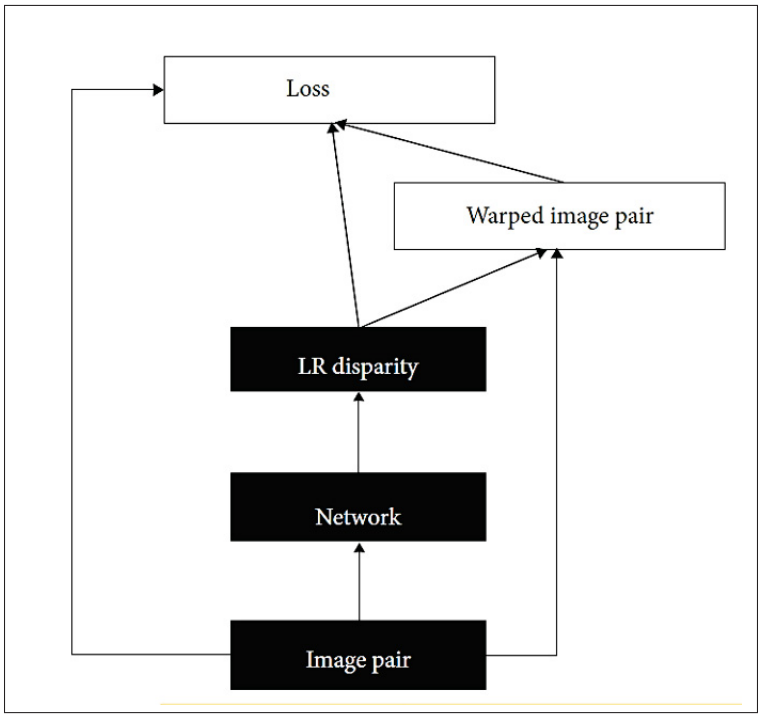

FIGURE 1.25 Méthodes traditionnelles pour l'estimation des disparités par adaptation de domaine
Tirée de Zhou *et al.* (2020)

D'autres méthodes s'intéressant à l'estimation du flux optique ont cherché à incorporer au problème de disparité monoculaire non-supervisé des contraintes d'odométrie (orientation, position), afin d'enrichir l'apprentissage du modèle (Zhou *et al.* (2020)). Zhou, Brown, Snavely & Lowe (2017) associe l'estimation de profondeur avec l'estimation du mouvement de la caméra (*ego-motion*). Vijayanarasimhan, Ricco, Schmid, Sukthankar & Fragkiadaki (2017)

propose un réseau de neurones multitâche pour l'estimation du mouvement dans une vidéo, pouvant à la fois apprendre la profondeur, la segmentation et le mouvement de la caméra et d'objets rigides. D'une approche similaire, Yin & Shi (2018) vont développer une méthode traitant en même temps la profondeur, le flux optique et l'odométrie.

Les méthodes plus récentes ont cherché à réduire le changement de domaine par l'apprentissage adverse et une fonction GAN (Goodfellow *et al.* (2014)). Elles considèrent les données synthétiques comme source et les données réelles comme cible. Kundu *et al.* (2018) proposent d'entraîner leur modèle à l'aide d'un discriminateur classifiant les représentations intermédiaires des domaines et un discriminateur en sortie du réseau discriminant la prédiction de la vérité terrain (remplace la fonction coût). Ils cherchent ainsi à rendre leur modèle invariant au domaine afin qu'il traite une image synthétique et réelle de la même manière. Zheng, Cham & Cai (2018) développent une stratégie d'apprentissage utilisant aussi un discriminateur intermédiaire, mais décident d'ajouter un réseau de translation d'image et une fonction coût de reconstruction. Ils emploient un GAN pour translater les images synthétiques et réelles vers le domaine réelle, et une fonction de reconstruction s'assure que l'image réelle translatée reste identique à sa version initiale. Ainsi leur modèle produit des cartes de profondeur en étant directement entraîné sur le domaine cible. Atapour-Abarghouei & Breckon (2018) vont au contraire utiliser un cycleGAN (Zhu *et al.* (2017)) pour translater les images cible vers le domaine source. Le modèle estimant la profondeur est donc seulement entrainé sur le domaine source. Si on veut utiliser cette méthode directement dans le domaine réel, cela implique l'utilisation en série du réseau de translation réel vers synthétique puis du réseau de disparité.

Au moment de la réalisation de ma cherche, très peu de méthode de disparité en apprentissage profond emploient l'adaptation de domaine sur un réseau de neurones basé sur la stéréo. Appliquer les méthodes monoculaires non-supervisées sur un problème de stéréoscopie n'est pas trivial. En effet, si on ne cherche pas à préserver les relations géométriques épipolaires, on obtient des performances inférieures aux méthodes de *stereo-matching* traditionnelles. Ainsi, Tonioni, Tosi, Poggi, Mattoccia & Stefano (2019) ont proposé le MADNet une méthode 2D de disparité basée sur la stéréo. Elle est adaptable au domaine en temps réel, par un apprentissage continuel sur les données en entrée. L'adaptation en temps réel est réalisée par *self-supervision*

(auto-apprentissage) grâce à sa structure modulaire, sa prédiction de disparités à plusieurs résolution, et une fonction coût de reconstruction multi-échelle, inspirée de Godard *et al.* (2017).

1.5 Limitation des méthodes actuelles de DO3D par stéréo

Les défis des approches de détection d'objet actuelles sont d'assurer une grande précision et efficacité. Une détection de grande précision se traduit par une reconnaissance et détection d'objets précise et rigoureuse à partir d'une image issue d'un environnement non-contrôlé. Cela signifie aussi que la méthode doit être robuste face à la large possibilité de classe et d'aspect d'un même objet. Un problème majeur dans la précision d'un détecteur est que lorsqu'un objet est proche de la caméra, celui-ci est facilement distinguable, mais lorsqu'il est loin de la caméra et donc apparaît petit sur l'image, la précision chute fortement.
Une détection efficace se traduit par un faible taux de faux positif, un fonctionnement en temps réel, un faible coût computationnel et une consommtion de mémoire optimale.

Les méthodes actuelles de détection 3D par image stéréo possèdent encore certaines limitations les empêchant d'avoir des performances similaires aux méthodes de DO3D par lidar. Cet écart de performance se trouve dans :

1. **La précision du modèle.** Les méthodes de DO3D par stéréo atteignent une précision de détection encore éloignée des méthodes par lidar (figure 1.26). Cet écart est notamment dû aux erreurs faites par les méthodes d'estimation de disparité ou de profondeur qui affectent fortement la détection. L'erreur de disparité augmentant quadratiquement avec la distance, la détection d'objet lointain atteint de faibles performances face aux méthodes par lidar. On remarque que des méthodes comme Pseudo-lidar++ (PL++) (You *et al.* (2020)) ou bien DispRCNN (Sun *et al.* (2020)) proposent des versions de leur modèle utilisant les données lidar comme entrée supplémentaire ou dans la supervision, ce qui permet d'augmenter la précision de la méthode ;

2. **La rapidité du modèle.** Le temps d'exécution des méthodes reste encore loin d'un fonctionnement en temps réel si on cherche un bon compromis entre précision et rapidité. Contrairement aux méthodes de DO3D par lidar, elles doivent traiter 2 images en parallèle mais aussi chercher la correspondance entre les pixels pour estimer les disparités. Des méthodes comme ZoomNet (Xu & al. (2020)) ou bien DispRCNN (Sun *et al.* (2020)) propose d'estimer les disparités seulement au niveau des RoIs ce qui permet de gagner en temps et calcul. D'autres méthodes vont plutôt chercher à employer un réseau de disparité 2D ou des algorithmes conçus à la main pour gagner en rapidité. Le RT3DStereo (H.Konigshof *et al.* (2019)) est la seule méthode s'approchant du temps réel en détection 3D mais ses pauvres performances montrent la difficulté de bien balancer précision et rapidité. L'utilisation d'un détecteur 2D permet plus facilement d'atteindre des vitesses de prédiction en temps réel dû aux très bonnes performances obtenues par ces détecteurs. On remarque que les méthodes état de l'art utilisant les données lidar sont plus rapide et très proche d'un fonctionnement temps réel ;

3. **La robustesse du modèle.** Pouvoir extraire et encoder, à partir d'images stéréo RGB, les caractéristiques 3D importantes tout en négligeant celle non-essentielles est nécessaire pour une détection robuste. Ces caractéristiques permettent au réseau de représenter le plus fidèlement possible la scène 3D et ainsi la comprendre. Mais les images ne permettent pas de donner explicitement l'information de profondeur, et la robustesse du réseau est affectée par les occlusions, les régions de l'image sans textures et répétitives (ciel, route) ainsi que par les variations des conditions météorologiques (jour, nuit, pluie). Afin de rendre le réseau plus robuste, le Pseudo-lidar (PL) (Wang *et al.* (2019)) est le premier à proposer de convertir la carte de disparité prédite en nuage de points. Il s'inspire des méthodes de DO3D par lidar en voulant exploiter les informations d'espace libre et de densité du nuage de points. D'autres méthodes, comme le 3DOP (Chen *et al.* (2018b)), décident d'utiliser un RPN encodant les informations du nuage de points pour construire des boites 3D préétablies dans les régions à haute probabilité d'objet. Mais il est nécessaire de créer un grand nombre de proposition 3D pour avoir une bonne précision et rappel. La méthode stéréo la plus robuste est la DSGN

(Chen *et al.* (2020)), qui encode la structure 3D de la scène dans un Plan-Sweep-Volume (PSV) qui est transformé en un volume 3D, en restant fidèle à la géométrie de la scène. Cependant l'utilisation de techniques 3D demande beaucoup de ressource computationnelle. De plus la majorité des méthodes s'intéressent à une seule classe (voiture), ce qui affecte la robustesse de leur modèle en diminuant leur capacité de discrimination des classes d'objet ;

4. **Les bases de données utilisées.** La diversité et la taille de la base de données influence très fortement les performances du modèle. Les méthodes stéréo de la littérature utilisent toutes la benchmark KITTI (Menze *et al.* (2015) et Geiger *et al.* (2012)) pour l'apprentissage et le test. Cette base à la particularité de contenir des images stéréo bien annotées pour de nombreuses tâches (détection, disparité, tracking. . .). Cependant dans les images proposées, il y a peu de variation des conditions lumineuses, météorologiques et de point de vue, les classes sont mal balancées (75% voiture, 4% cycliste and 15% piéton d'après Arnold *et al.* (2019)). De plus, la faible quantité de données disponibles pour la tâche de détection 3D et d'estimation de disparité (14 000 pour la détection et 400 pour l'estimation de disparité) empêche le réseau de généraliser ce qu'il a appris à d'autre scénario. En effet, les modèles sont testés sur la base test KITTI qui a été prise dans des conditions similaires que la base apprentissage. Cependant, si le réseau doit être testé sur une base privée et installé sur le terrain, on risque d'observer une forte chute dans les performances ;

5. **La généralisation du modèle.** L'adaptation de domaine permet deux choses : de faire face aux manques de données annotées sur la base cible et d'améliorer les performances du modèle lors d'un changement de domaine. Cependant, on peut voir tout d'abord que les plus grandes avancées en adaptation de domaine sont limitées à la tâche de classification. Il n'y pas de méthode notable, à ce jour, appliquant l'adaptation de domaine en détection 3D par stéréo. Quelques auteurs (Zheng *et al.* (2018) et Atapour-Abarghouei & Breckon (2018) et Kundu *et al.* (2018)) l'ont appliqué pour l'estimation de profondeur à partir d'une image, mais les gains en performances observées ne sont pas comparable aux méthodes stéréo.

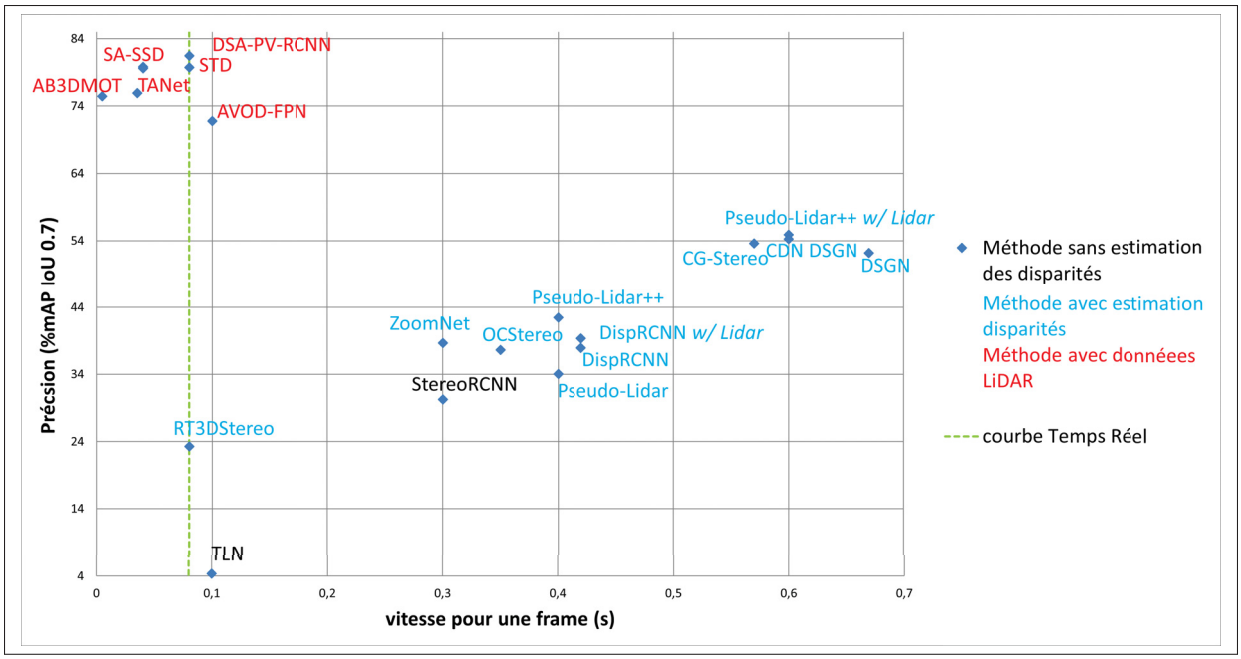

FIGURE 1.26 Précision moyenne de détection 3D en fonction du temps de prédiction, pour la classe voiture de la base KITTI 2012. Les méthodes avec l'indication *w/Lidar* réalisent en plus une supervision du nuage de points

C'est à partir des limitations observées que nous avons choisi de proposer une nouvelle méthode de détection 3D qui va permettre de répondre aux contraintes de vitesse, précision et robustesse. Cette méthode va associer un réseau de disparité 2D avec un réseau de détection 3D basée sur des données lidar. Pour cela, nous allons d'abord sélectionner un réseau de disparité 2D assurant le meilleur compromis entre vitesse, précision et calcul, puis nous allons chercher à le rendre robuste face au changement de domaine du synthétique vers le réel. Ce nouveau réseau de disparité sera finalement utilisé pour réaliser la détection 3D.

CHAPITRE 2

ESTIMATION DES DISPARITÉS

2.1 Choix du modèle de base

2.1.1 Choix de la méthode de détection 3D par stéréo

4 grandes familles de DO3D par image-stéréo ont été présentées dans la revue de littérature. Parmi ces approches, celles assurant les meilleures performances (voir figure 1.26) emploient un réseau de disparité afin de transformer la carte produite en un nuage de points 3D, puis la détection 3D est réalisée par l'une des deux méthodes suivantes :

1. **Réseau de détection 2D.** Cette approche estime les boites 3D grâce à un détecteur 2D associée aux contraintes géométriques de la classe-objet. Cela permet de profiter de la rapidité et précision d'un détecteur 2D état de l'art. Elle a la particularité de pouvoir faire fonctionner le réseau de disparité et de détection 2D en parallèle, et demande peu de ressources computationelles comparée à un détecteur 3D. Les performances de cette approche vont surtout dépendre du réseau de disparité choisi et des contraintes géométriques définies pour le filtrage du nuage de point ;

2. **Réseau de détection 3D.** On utilise directement un réseau développé pour la détection 3D à partir d'un nuage de points. Cette approche profite des caractéristiques d'un détecteur 3D, qui permettent d'exploiter au mieux les informations 3D de la scène. Les détecteurs 3D état de l'art (DSA-PV-RCNN de Bhattacharyya *et al.* (2021), SA-SSD de He *et al.* (2020) ou AVOD-FPN de Ku *et al.* (2018)) basés sur un nuage de points assurent déjà de très bons résultats. Ainsi, les performances globales de l'approche vont encore une fois dépendre du réseau de disparité choisie car c'est lui qui construit le nuage de points.

Si on cherche à gagner en rapidité, on emploiera un réseau de disparité 2D. Tandis que si on cherche à être le plus précis possible sans se soucier du coût de calcul, on utilisera une méthode de disparité 3D.

Dans le cadre de la recherche, il a été décidé de choisir cette famille de méthodes en DO3D par stéréo car leurs capacités à encoder l'information de profondeur et d'espace par la disparité, et les informations de forme de l'objet par la détection permettent d'atteindre une bonne précision comparée aux autres méthodes. Elles peuvent recourir à des approches état de l'art de détection 3D ou 2D pouvant prédire en temps réel des boites encadrantes précises. Les modèles peuvent être entraînés à partir de zéro en simultané (*end-to-end*) ou indépendamment, ce qui permet de pré-entraîner chacun des modèles à sa tâche. De plus, cette approche s'intègre facilement dans l'architecture du projet Dista, en remplaçant l'algorithme de *stereo-matching*.
Comme vu précédemment, les méthodes de détection 3D par stéréo basées sur un réseau de disparité ont une bonne partie de leurs erreurs de prédiction provenant de l'estimation de la carte de disparité. Cela est dû à la difficulté d'estimer avec précision la profondeur pour les objets petits, obstrués et lointains (Xu & al. (2020)). C'est pourquoi les travaux suivants vont se concentrer sur l'optimisation de l'architecture de la méthode estimant les disparités pour améliorer ses performances. On va déterminer par la suite le réseau de disparité le plus adapté au problème.

2.1.2 Choix de la méthode d'estimation des disparités

Les méthodes pour l'estimation des disparités basées sur des images-stéréo vont réaliser la correspondance entre les pixels soit à l'aide de convolutions 2D soit de convolutions 3D. Le premier cas permet d'atteindre des prédictions en temps réel et sollicite une quantité en ressource de calcul raisonnable. Mais l'utilisation de convolution 2D seule, pour encoder la profondeur, ne permet pas d'obtenir une bonne prédiction. Les convolutions 3D permettent d'encoder avec une meilleure précision les informations de la scène en considérant la dimension de disparité. Cependant, ces techniques 3D demandent beaucoup en ressource de calcul ce qui diminue ainsi

fortement le temps de prédiction. Dans les parties suivantes, nous allons d'abord présenter les critères du cahier charges puis nous allons comparer les méthodes de disparité selon ces critères.

2.1.2.1 Critères du cahier des charges

Afin de concevoir le système de caméra pour la mesure de distance lors d'un dépassement, un cahier des charges avait été dressé. Ainsi, à partir du cahier des charges, nous avons défini les critères suivants permettant de choisir le réseau de disparité le plus adapté à notre problème :

- vitesse de prédiction $\geq$ 16 f/s (temps réel) ;
- empreinte mémoire (inférence) $\leq$ 1,5 Go ;
- précision de la prédiction (supervisée) : (D1) $<$ 3% ou (EPE) $<$ 1 ;
- difficulté de reproduction de la méthode : faible.

Le critère de vitesse doit prendre en compte le fonctionnement en temps réel du système de mesure, et la vitesse du véhicule lors du dépassement. Une prédiction en temps réel signifie qu'il y a une fluidité (pas de latence) dans l'enchaînement des frames et des boites prédites, ce qui permet d'estimer une distance moyenne tout le long du dépassement. On considère une vitesse en temps réelle quand la prédiction est inférieure ou égale à 60ms par frame (soit 16fps). De plus, la vitesse d'un véhicule en agglomération est de l'ordre de 50km/h soit environ 14m/s, tandis que la vitesse moyenne d'un cycliste est de 15km/h soit 4m/s. Ce qui correspond à une vitesse relative de 10m/s. Si on suppose que la taille d'un vélo se situe entre 1-2m, alors le véhicule prend minimum 0,1 seconde pour dépasser un cycliste. Sachant que, selon Xu & al. (2020) et les données de la base de détection KITTI, un détecteur 3D peut détecter des objets au moins à partir de 40m du système de caméra, alors un fonctionnement en temps réel est largement suffisant pour capturer un dépassement sur 40m.

L'empreinte mémoire du réseau est limitée car on doit pouvoir implémenter le modèle sur un système embarqué. Plus particulièrement, celui-ci doit pouvoir fonctionner sur une Jetson Nano Nvidia qui possède une carte GPU de 4 Go, étant la carte choisie dans le cadre du projet Dista. Et en plus du réseau de disparité, il faut prendre en compte le réseau de détection 3D qui fonctionnera aussi sur la carte GPU.

L'erreur de prédiction du réseau doit être la plus faible possible. Les meilleurs modèles de la littérature (stéréo ou non) possèdent généralement une erreur (D1) au alentour des 2% sur la base de test KITTI2015, ayant une distribution similaire à celle d'apprentissage (figure 2.1). On considère donc que (D1) doit être inférieure à 3%.
La difficulté de reproduction est un outil de comparaison à la fois subjectif et objectif. Il correspond au degré de difficulté de prise en main du modèle : une compréhension rapide des techniques utilisées est importante pour pouvoir par la suite effectuer une amélioration de l'architecture et optimiser ses performances. Mais cela comprend aussi la disponibilité du code pour pouvoir reproduire les résultats obtenus dans le papier, et les librairies utilisées.

2.1.2.2 Base de données et outil d'évaluation

Les résultats des méthodes de disparité à partir d'images stéréo sont obtenus sur la fraction test de la base de données KITTI2015 (Menze *et al.* (2015)). L'outil d'évaluation est la mesure D1-all qui mesure le pourcentage de valeurs de disparité aberrantes (D1) moyenné sur tous les pixels de la vérité terrain.

2.1.2.3 Comparaison des méthodes selon les critères

Rapidité. Parmi l'ensemble des méthodes vu précédemment dans la littérature, on en retrouve 7 avec des performances en temps réel (figure 2.1). Parmi ces 7 méthodes, seulement une est construite à partir de convolutions 3D : le DeepPruner-Fast qui est une version du DeepPruner (Duggal *et al.* (2019)) plus légères permettant ainsi d'atteindre une prédiction en temps réelles mais entrainant une perte en précision (+0,44% d'erreur). Les méthodes 2D les plus rapides sont les modèles récents HitNet (Tankovich *et al.* (2021)) et Fast DS-CS (Yee & Chakrabarti (2020)) suivis par le MADNet (Tonioni *et al.* (2019)). A noter, les vitesses du Fast DS-CS et HitNet sont obtenues en utilisant des méthodes de calculs GPU optimisées.
La rapidité de prédiction étant le critère primordial pour le choix de la méthode finale, seules les méthodes ayant une prédiction en temps réel seront sélectionnées pour la comparaison.

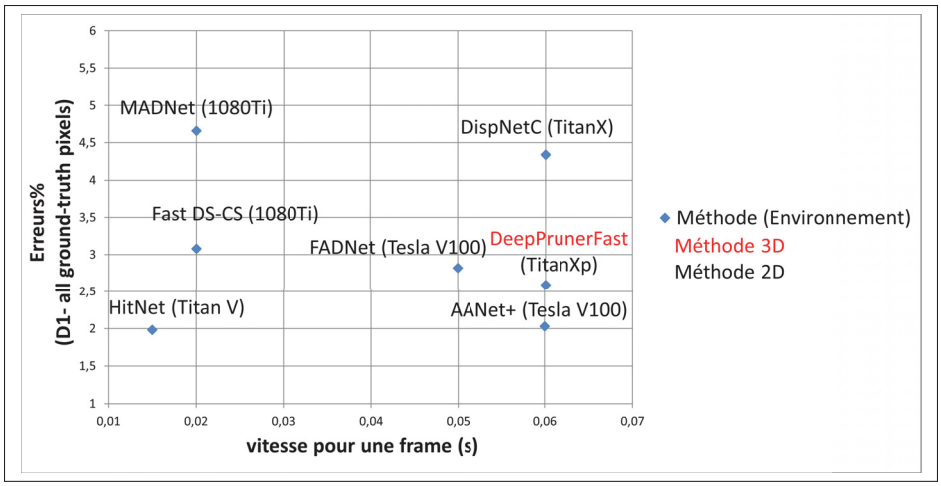

FIGURE 2.1 Comparaison des méthodes de disparité en temps réel sur KITTI 2015 (Menze *et al.* (2015)) selon l'erreur D1-all

Environnement GPU. Les modèles ont tous été évalués sur une même base de données (base de test de la KITTI2015) mais l'environnement GPU (*Graphics Processing Unit*) d'exécution du code varie d'une architecture à l'autre. On retrouve ainsi l'utilisation de cartes comme NVIDIA 1080Ti, Titan V, Tesla V100 ou bien Titan X. Les performances d'une carte graphique vont avoir tendance à influencer la rapidité de prédiction du modèle. Il est donc important de le prendre en compte pour comparer les modèles. Une carte GPU possède une horloge interne qui va influencer ses performances. La vitesse de l'horloge caractérise la vitesse à laquelle la carte va lire et écrire des données de sa mémoire (*Memory Clock*), mais aussi la vitesse de calcul pour le traitement des données (*Core Clock*).

Dans le tableau 2.1, on a comparé les 7 méthodes retenues selon la fréquence d'horloge de la carte GPU employée, et le nombre d'opérations sur des floats que peut effectuer le processeur chaque seconde (TFLOPs). On peut voir grâce au nombre de TFLOPs que les cartes Titan V et Tesla V100 ont les vitesses de calcul les plus élevées avec 14,9 TFLOPs et 14 TFLOPS respectivement. Elles sont suivies par la Titan Xp et 1080Ti. Ainsi les résultats des méthodes exécutées sur ces cartes GPU peuvent être, dans une certaine limite, comparés ensemble. Seul le réseau DispNetC a été testé sur une carte fortement inférieure (6,69 TFLOPs). Il est ainsi possible d'obtenir des résultats plus intéressant pour ce modèle, sur une carte graphique plus puissante.

Tableau 2.1 Performances des cartes GPU employées[a]

Carte GPU	Méthode associée	Fréquence d'horloge (Mhz)	TFLOPs (FP32)
Titan V	HitNet	850	**14,9**
Tesla V100	FADNet ; AANet	876	**14**
Titan Xp	DeepPruner-Fast	1426	12,15
1080Ti	Fast DS-CS ; MADNet	1376	11,34
GTX Titan X	DispNetC	1753	6,69

[a]Les résultats sont des approximations et ne représentent pas les performances réelles (données issues de https://www.techpowerup.com/ et https://www.nvidia.com/)

Précision. La figure 2.1 permet d'observer la précision en fonction du temps de prédiction pour les méthodes en temps réel. HitNet assure la meilleure précision parmi l'ensemble des méthodes en temps réel avec un D1-all à 1,98%, suivie par AANet+ (une version optimisée du AANet de Xu & Zhang (2020)) où son erreur D1-all est de 2,03%. Un écart assez important est créé avec le DeepPruner-fast (D1-all = $2,59\%$) et le reste des méthodes. L'erreur de prédiction va jusqu'à 4,66% pour le MADNet. Les performances obtenues par le HitNet et AANet+ sur Kitti2015 sont autour des 2% d'erreur (D1-all) et ainsi comparable aux méthodes 3D.

Tableau 2.2 Empreintes mémoire des méthodes de disparité Real-Time[a]

Méthode	Consommation mémoire (Go)	Nombre de paramètres ($\times 10^6$)
DeepPruner - Fast	0,81	n.c.*
DispNetC	1,62	38
HitNet	n.c.	0,45
AANet	1,63	3,68
FADNet	3,87	77

[a]Les données sont issues des papiers d'origines. (*non connu)

Empreinte Mémoire. Un réseau de neurone profond va être composé de million de paramètres, appelés poids, qui ont été optimisés durant l'apprentissage et vont permettre de répondre à la tache durant la prédiction. Cependant ces poids demandent un espace de stockage qui peut être très important. En plus de cela, il faut prendre en compte la mémoire consommée par les calculs lors de la prédiction. Dans le tableau 2.2 on a regroupé pour certaines méthodes la consommation mémoire et le nombre de paramètres. Le nombre de paramètres est une indication très utile

donnant des informations sur la quantité de calcul à réaliser avant la prédiction, et la quantité de mémoire requise. De plus un modèle avec peu de paramètres sera moins susceptible de faire du sur-apprentissage et assurera une meilleur généralisation.
On peut voir ainsi que le DeepPruner-Fast, bien qu'étant une méthode de disparité 3D, consomme très peu en mémoire et moins que certaines méthodes 2D. Les convolutions 3D demandent beaucoup en ressource computationelle, mais l'emploie du module PatchMatch (Duggal *et al.* (2019)) permet de réduire considérablement l'empreinte mémoire. Les auteurs du HitNet ne précisent pas la consommation de mémoire lors d'une prédiction mais seulement son nombre de paramètres. Cette méthode prédit des cartes de plusieurs résolutions qui permettent d'affiner la carte principale à l'aide de modules de propagation spatiale. Comme elle nécessite peu de paramètres, on peut supposer que son empreinte mémoire est faible. Le FADNet de Wang *et al.* (2020) est la seule méthode du tableau dont on est sûr que son empreinte mémoire ne respecte pas le cahier des charges et contient le plus de paramètres. Les approches Fast DS-CS et MADNet ne donnent pas d'indication sur leurs consommations de mémoire, cependant on peut supposer que leurs empreintes est faible car le Fast DS-CS n'utilise que des convolutions 2D avec une méthode de Stereo-Matching traditionnelle, et le MADNet a été développé pour fonctionner en temps réel sur un système embarqué.

Reproductivité. On retrouve dans le tableau 2.3 différentes informations sur les méthodes préselectionnées, qui vont permettre d'évaluer la capacité de reproduction d'une méthode. Un modèle proposant une architecture légère et modifiant le moins possible la structure de base va permettre une meilleure compréhension de son fonctionnement pour pouvoir appliquer des améliorations par la suite, mais aussi une meilleure intégration à la stratégie d'apprentissage par adaptation de domaine. Ainsi des méthodes comme DeepPruner (utilisation de convolution 3D) ; AANet (ajout de techniques complexes appliquant des opérations dans l'espace des caractéristiques) ; FADNet (modification du DispNetC avec l'ajout d'un réseau résiduel) ne permettent pas une faible difficulté de reproduction.
La disponibilité du code et les outils informatiques utilisés pour développer le modèle jouent aussi un rôle important. Étant plus familier avec le langage Python et la librairie Pytorch en

apprentissage profond, l'utilisation de ces outils assurent une faible difficulté de reproduction. Bien que les méthodes HitNet et Fast DS-CS assurent de très bonnes performances et semblent les plus intéressantes, celles-ci utilisent des calculs GPU optimisées ce qui ne permet pas une reproduction de leurs résultats dans une configuration normale. De plus, les auteurs du HitNet n'ont pas rendu leur code public, et le Fast DS-CS a été développé en utilisant la librairie Tensorflow. Le DispNetC à la particularité d'être la première approche de structure U-Net estimant les disparités. Nombreuses sont les méthodes ayant utilisées sa structure comme base (FADNet, Fast DS-CS, AANet). Elle est donc plus propice à l'ajout d'amélioration, et à ce qu'on l'emploie comme réseau de base à l'adaptation de domaine.

Tableau 2.3 Paramètres influant la reproductivité pour les méthodes de disparités

Méthode	**Conv.**	**Baseline**	**Code disponible**	**Librairie DL**	**Techniques utilisées**
DeepPruner - Fast	3D	Pyramid-Net (Chang & Chen (2018))	Oui	Pytorch	PatchMatch
DispNetC	2D	U-Net (Dosovitskiy, Fischer, Ilg & al. (2015))	Oui	Pytorch	Couche de corrélation
HitNet	2D	U-Net (Mayer & Ilg (2016))	Non	/	Propagation spatial ; calculs GPU optimisés
AANet	2D	Pyramid-Net (Chang & Chen (2018))	Oui	Pytorch	Couches de corrélation ; Agrégation intra-échelle ; Agrégation extra-échelle
FADNet	2D	U-Net (Mayer & Ilg (2016))	Oui	Pytorch	Couche de corrélation ; Branche résiduelle
Fast DS-CS	2D	U-Net (Mayer & Ilg (2016))	Oui	Tensorflow	calculs GPU optimisés ; Stereo-Matching
MADNet	2D	Pyramid-Net (Ranjan & Black. (2017))	Oui	Tensorflow	Auto-adaptation

2.1.2.4 Conclusion et choix du modèle de base

Le DispNetC de Mayer & Ilg (2016) est le réseau d'estimation des disparités choisi pour la suite du problème. Il propose une architecture U-Net légère et efficace, qui sera optimisée par la suite afin d'améliorer ses performances. Cette méthode assure un temps de prédiction en temps réel (de 16 f/s), et une précision de 4,34% sur la KITTI2015. On cherchera donc d'abord à améliorer sa précision tout en modifiant le moins possible sa vitesse et son empreinte mémoire. Le choix de ce réseau de disparité va permettre par la suite de l'associer soit à un réseau de détection 3D état de l'art.

2.2 Amélioration du modèle de base

La partie précédente nous a permis de choisir, à partir des contraintes du cahier des charges, le DispNetC pour l'estimation des disparités. Nous allons maintenant présenter l'architecture de ce modèle et les différentes améliorations retenues pour optimiser les performances initiales du réseau.

2.2.1 Approche initiale : DispNetC

Le DispNetC (Mayer & Ilg (2016)) est un réseau de structure U-Net (Ronneberger *et al.* (2015)) conçu pour l'estimation des disparités, dont l'architecture est inspirée par le FlowNet de Dosovitskiy *et al.* (2015). *À* partir d'images stéréo, il va produire la carte de disparité associée. L'encodeur du modèle va servir à extraire les caractéristiques importantes pour la tâche et produire le volume coût grâce à une couche de corrélation explicite 1D. Le décodeur va utiliser les représentations apprises pour estimer des cartes de disparité à multi-échelles grâce à des connexions longue portée avec l'encodeur. Les figures 2.2 et 2.3 présentent l'architecture du modèle.

2.2.1.1 Détails de l'encodeur

L'encodeur (figure 2.2) est la partie contractant les informations de disparité issues des images stéréo dans un volume coût. Il applique des convolutions 2D avec un *stride* de 2 pour certaines couches, ce qui aboutit à une réduction finale par un facteur 64 de la résolution de l'image.

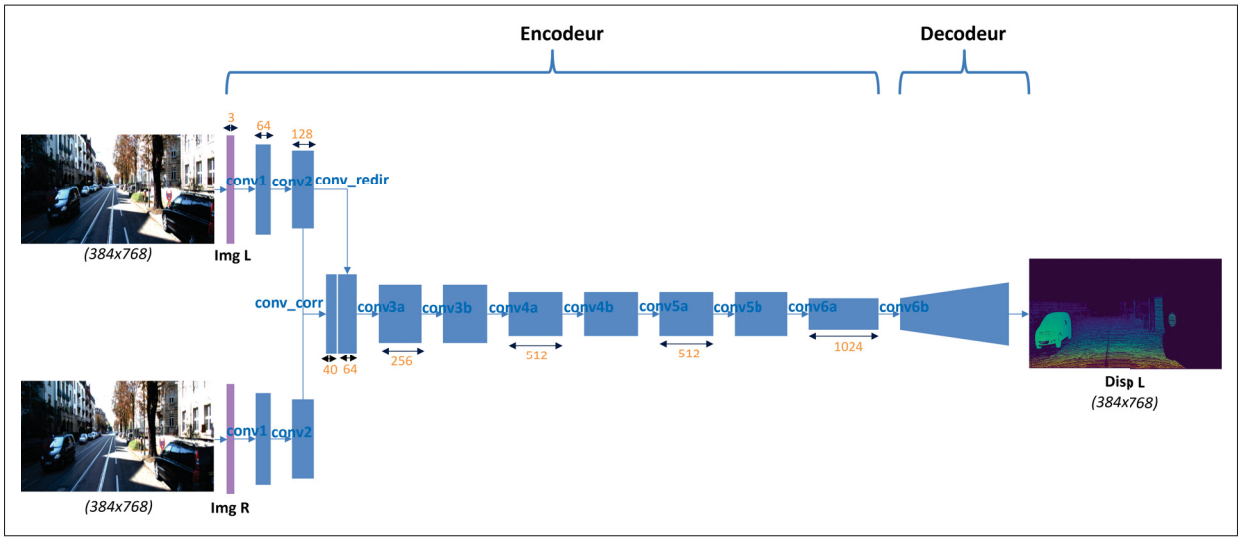

FIGURE 2.2 Schéma encodeur DispNetC*

*Les blocs violets représentent les images en entrée pré-traitées. Les blocs bleus sont les cartes de caractéristiques en sortie des couches de convolution 2D. En orange est indiqué le nombre de channel. Les poids des branches siamoises sont partagés lors de l'apprentissage. Le DispNetC reçoit en entrée les images stéréo (ImgL, ImgR) qui sont traitées en parallèle puis concaténées avant d'être fournies à la couche de corrélation *conv_corr* (avec D=40). Le volume coût produit est ensuite réduit par des couches de convolution 2D, puis est transmis au décodeur pour la prédiction de la carte de disparité (DispL).

Entrée siamoise. L'entrée de l'encodeur a une forme siamoise jusqu'à la couche *conv2*, permettant de traiter en parallèle les images stéréo préalablement rectifiées de résolution (768×384). Puis les cartes de caractéristiques gauche et droite sont concaténées pour être traitées par une couche de corrélation.

Couche de corrélation. La couche de corrélation *conv_corr* permet de remplacer le processus de *stereo-matching* et par conséquent entraîne le réseau à trouver les correspondances entres les

images stéréo. *À* partir de la carte gauche F_l et de la carte droite F_r de même taille ($H \times W \times C$), on calcul un volume de corrélation F_{corr} de même résolution avec D channels qui correspond au nombre de valeurs de disparités possible entre les deux images. Ainsi pour un pixel à la position (x, y) de la carte de caractéristique gauche, le modèle cherche le pixel de la carte droite correspondant en calculant différents coûts de corrélation pour chaque channel $d \in [0; D[$ de la carte F_{corr} à la même position (x, y). La valeur de ce coût est le produit scalaire de $F_l(x, y)$ avec $F_r(x - d, y)$ (Mayer & Ilg (2016)). Comme on recherche les correspondances à partir d'un pixel gauche dans la carte droite, on réalise les déplacements vers la gauche soit de F_l selon $-x$ avec un *stride* de 1 (d'où $x - d$). L'équation 2.1 représente le fonctionnement de la couche de corrélation 1D pour un pixel à la position (x, y) de la channel i de F_{corr}.

$$F^i_{corr}(x, y) = \langle F_l(x, y), F_r(x - i, y) \rangle \tag{2.1}$$

Cette couche n'a pas de paramètres entraînables car elle effectue la convolution entre données et n'utilise pas de filtre (Dosovitskiy *et al.* (2015)). Le nombre de multiplication de l'équation 2.1 pouvant être très élevé, un maximum de déplacement D est imposé.Mayer & Ilg (2016) considère un $D = 40$ pixels ce qui correspond à un déplacement de 160 pixels dans l'image d'entrée. Ce choix correspond au déplacement maximal observable dans les vérités terrain de la base réelle KITTI, qui est la base de données test.

De plus, contrairement au FlowNet qui utilise une corrélation 2D pour l'estimation du flux optique, la couche de corrélation est 1D c'est-à-dire qu'elle cherche les correspondances seulement selon l'axe x. Une corrélation 1D demande un coût de calcul plus faible, et permet de chercher des correspondances plus finement pour de plus grand déplacement horizontal. En effet, dans un problème d'estimation des disparités, on recherche par convention les correspondances entre les images stéréo selon les lignes épipolaires.

Compression du volume coût. Le volume produit est ensuite traité successivement par des couches de convolution 2D (*conv3a* à *conv6b*) en alternant entre un *stride* de 1 et 2. Ce qui permet de réduire la résolution du volume mais va doubler son nombre de channels, permettant

ainsi au réseau d'apprendre de grand déplacement. La dernière couche produit une carte de caractéristique à 1024 channels.

2.2.1.2 Détails du décodeur

Le décodeur (figure 2.3) va progressivement expandre le volume des caractéristiques issu de l'encodeur à l'aide de couches de déconvolution et convolution 2D, mais aussi en réutilisant des informations de l'encodeur par des *skip connections* (connexions longue portée).

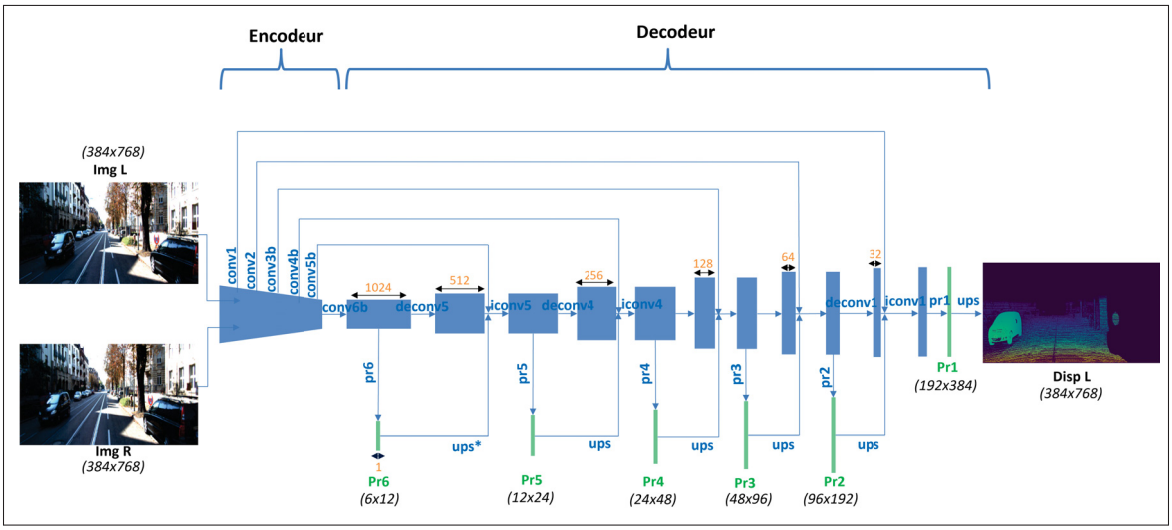

FIGURE 2.3 Schéma décodeur DispNetC*

*Soit $i \in [1;6]$. Le décodeur prédit à l'aide des couches *pri*, plusieurs résolutions *Pri* de la carte de disparité. La couche *iconvi* utilise la concaténation de la prédiction précédente *pr(i+1)* avec la carte de caractéristique *deconvi* (déconvolution) et les informations du décodeur *convi* de même échelle (par la *skip connection*), pour produire une représentation des disparités plus détaillée en sortie de la couche *pri*. La prédiction finale est *Pr1* qui est agrandie par la fonction *ups* pour produire DispL, la carte de disparité selon l'image gauche, de résolution (384×768).
***ups** : upsample (fonction d'agrandissement bilinéaire)

Expansion du volume. Il est nécessaire de réduire progressivement la résolution des cartes de caractéristique pour agréger les informations contenues dans différentes régions de l'image, mais aussi pour réduire le nombre de paramètres du réseau et rendre son apprentissage faisable. Cependant, le modèle doit pouvoir prédire à la fin des cartes de disparité dense, ce qui

oblige d'affiner les représentations approximatives des disparités des cartes de faible résolution (Dosovitskiy *et al.* (2015)). Pour cela, un affinage progressif des caractéristiques est effectué (voir figure 2.4) : la carte de caractéristique va être traitée par une couche de déconvolution *deconvi* augmentant sa résolution, puis concaténée avec la carte de résolution correspondante dans l'encodeur *convi* par des *skip connections*, et avec la carte de disparité prédite *pr(i+1)* à la résolution précédente (si disponible) qui a été agrandie (*upsample*), créant ainsi le volume *iconvi*. On affine à chaque étape du décodeur la représentation grâce aux informations de haut niveau issues des cartes précédemment prédites, et grâce aux informations locale de faible niveau présent dans l'encodeur.

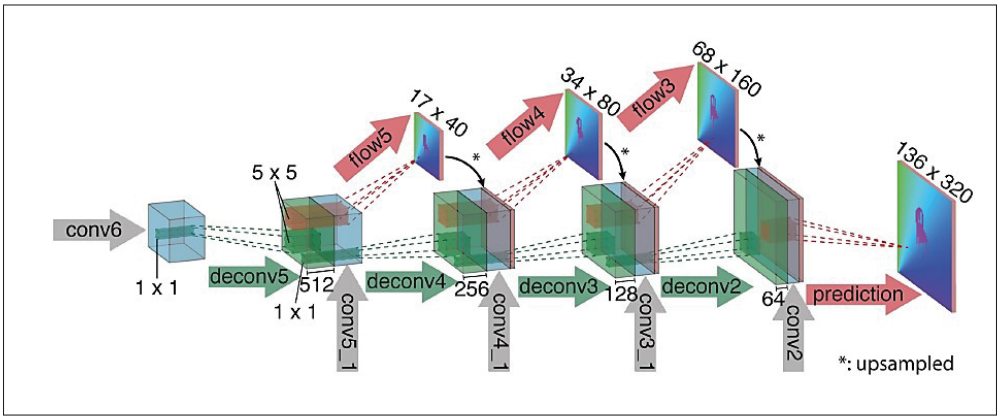

FIGURE 2.4 Principe de l'affinage progressif de la représentation des caractéristiques de basse résolution vers les hautes résolutions
Tirée de Dosovitskiy *et al.* (2015)

Prédiction multi-échelles. Le décodeur réalise la prédiction de 6 cartes de disparité *Pri* de différentes résolutions. Chaque carte de disparité est générée par une couche de convolution 2D *pri* appliquée au volume *iconvi* issu de l'agrégation d'une déconvolution, de la carte de disparité prédite juste avant et des informations extraites par le décodeur. A chaque étape de prédiction, la résolution de la carte de disparité est doublée (résolution allant de 12×6 à 384×192) jusqu'à la prédiction de la carte finale DispL de taille 768×384. L'utilisation d'une prédiction multi-résolutions permet de renforcer la prédiction principale par des informations issus des couches précédentes mais aussi d'effectuer un affinage des représentations brutes des disparités

dans les cartes de basse résolution vers une représentation plus détaillée dans les cartes de haute résolution (Dosovitskiy *et al.* (2015)).

Fonction coût globale. Une fonction coût multi-échelles L_{disp} est utilisée avec une stratégie de pondération pour mettre à jour les paramètres du réseau vers leurs valeurs optimales. Pour chaque résolution i, l'erreur de disparité L1 est calculée par la différence au niveau des pixels entre la carte prédite y_{D_i} et la vérité terrain $\hat{y_D}$. Les vérités terrains étant de résolution 768×384, les cartes de résolutions inférieures sont agrandies par une fonction bilinéaire pour calculer l'erreur. De plus, on ajoute une fonction $L_{smoothdisp}$ permettant de lisser les prédictions au niveau des frontières entre les objets dans la scène, évitant ainsi de grand écart de disparité entre deux pixels voisins appartenant à des objets différents.
La fonction coût global s'écrit donc :

$$L_{disp}(f_T) = \sum_{i=1}^{6} loss_i = \sum_{i=1}^{6} \omega_i [L_1(y_{D_i}, \hat{y_D}) + c \times L_{smoothdisp}(y_{D_i})] \tag{2.2}$$

avec $c = 0,1$ choisi empiriquement, où

$$\begin{aligned} L_1(y_{D_i}, \hat{y_D}) &= \| \, y_{D_i} - \hat{y_D} \, \|_1 \\ L_{smoothdisp} &= | \, \partial_x y_{D_i} \, | + | \, \partial_y y_{D_i} \, | \end{aligned} \tag{2.3}$$

La fonction 2.2 étant composée de la somme des erreurs de disparité à chaque résolution i, on considère une pondération ω_i pour chaque résolution, qui évolue en fonction de l'apprentissage. Cette stratégie permet d'apprendre les disparités d'une manière grossier à fin (*coarse-to-fine*), mais aussi confère une meilleur généralisation (Wang *et al.* (2020)).

2.2.2 Amélioration du DispNetC

Afin d'améliorer les performances globales du DispNetC, nous allons chercher à affiner la représentation des caractéristiques dans le réseau, mais aussi à les enrichir en informations de contexte global. Nous allons aussi nous intéresser à augmenter l'efficacité du réseau par un apprentissage de caractéristiques robustes et essentielles à la tâche d'estimation des disparités. Dans la suite, nous allons présenter une nouvelle architecture du DispNetC assurant une meilleure efficacité lors de l'apprentissage, plus robuste, et atteignant une meilleure précision par rapport au modèle initial. Son architecture sera modifiée en nombre de couche et par l'ajout de modules adaptés à sa structure U-Net, utilisés dans les méthodes de segmentation (Ronneberger *et al.* (2015); Jha *et al.* (2019); Gu *et al.* (2021); Azad, Asadi-Aghbolaghi, Fathy & Escalera (2019)). Nous allons introduire des blocs résiduels et des blocs Squeeze&Excite à travers le réseau pour optimiser son apprentissage et sa robustesse. Deux types de couches de Spatial Pyramid Pooling (SPP) seront étudiées dans l'encodeur pour améliorer la qualité des représentations. De plus, on cherchera à affiner la prédiction multi-échelles du décodeur en ajoutant des blocs d'attention spatial, de channel et des résolutions, qui vont mettre en avant les informations nécessaires à l'estimation des disparités.

2.2.2.1 Amélioration globale

Afin d'améliorer l'apprentissage du modèle, les blocs de convolutions 2D sont substitués par des blocs résiduels. Ce type de bloc développé par He, Zhang, Ren & Sun (2016), était initialement utilisé pour la tache de classification. Il propage l'information à travers les couches grâce à la *skip connection*, ce qui permet d'entraîner efficacement des modèles profonds en demandant un coût de calcul moins important et évitant par conséquent le problème de disparition du gradient. Mais il assure aussi l'apprentissage de caractéristiques robustes pour la tâche. On peut donc insérer des couches supplémentaires au modèle sans rencontrer des difficultés à l'entraîner. Ainsi, nous allons rajouter un ensemble de couche (*deconv0, iconv0, pr0*) pour affiner la prédiction finale, mais aussi des couches de Squeeze&Excite (Hu, Shen, Albanie, Sun & Wu (2020)). Le but du bloc Squeeze&Excite (figure 2.5) est d'augmenter la sensibilité du réseau envers les

caractéristiques utiles pour qu'il supprime ceux non nécessaire au problème. En activant des dépendances entre les channels, les représentations produites par le réseau sont affinées. De plus, Il est associé au bloc résiduel pour améliorer les capacités de généralisation du modèle (voir Jha *et al.* (2019)).

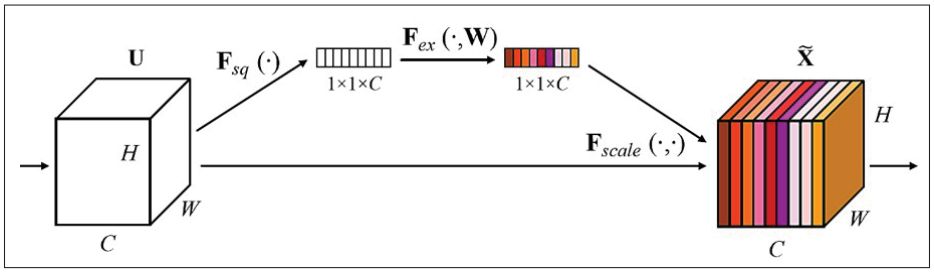

FIGURE 2.5 Schéma bloc Squeeze&Excite*
Tirée de Hu *et al.* (2020)

*La première étape est le *squeeze* qui va regrouper les informations globales du volume U (de taille $H \times W \times C$) par une couche *average pooling*. La deuxième étape est *excitation* capturant les dépendances à travers les channels pour activer les plus importantes à la tache et supprimer celles non nécessaires.

2.2.2.2 Amélioration encodeur

L'encodeur permet d'extraire les informations contenues dans les images pour en apprendre une représentation utile répondant à la tâche. Cependant, pour produire des cartes détaillées, il est nécessaire de connaître la relation d'un pixel dans le contexte global de la scène. Dans la tâche d'estimation des disparités, avoir l'information de contexte des caractéristiques est important. Deux pixels voisins peuvent appartenir à deux régions d'objet différentes et avoir des valeurs de disparité très éloignées. Ou bien au sein même d'un objet, selon son orientation, on peut avoir des pixels ayant des valeurs de disparités très éloignées. C'est pourquoi l'agrégation des informations de contexte global et des caractéristiques local permet d'enrichir les représentations produites par le modèle mais aussi d'améliorer le processus de *stereo-matching* du réseau, en le poussant à chercher les correspondances dans les régions de contexte similaire.

Afin d'exploiter les informations de contexte global, le module de *Spatial Pyramid Pooling* (SPP) (Zhao, Shi, Qi, Wang & Jia (2017)) proposé initialement dans le domaine de segmentation

d'image, sera employé. Il permet d'élargir le champ réceptif et de capturer les informations au niveau de toute l'image et ainsi d'améliorer la prédiction. Cela permet de faire face aux erreurs dues aux variations d'échelles d'un même objet, en traitant un objet loin et un proche d'une manière similaire. Mais aussi de résoudre le problème de discontinuité aux frontières en prenant en compte les pixels voisins.

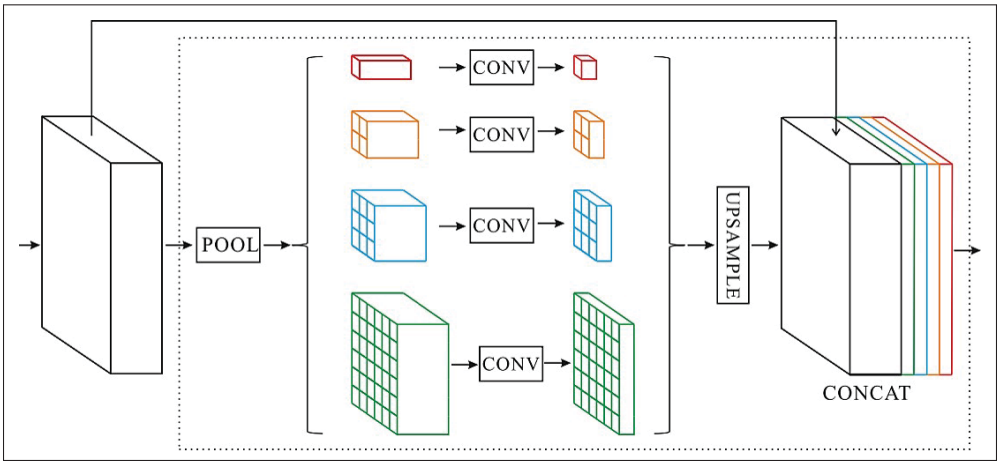

FIGURE 2.6 Schéma du bloc SPP*
Tirée du PSPNet (Zhao *et al.* (2017))

*Le module SPP applique différentes résolutions de *average pooling*, qui sont agrandies à la résolution d'entrée puis agrégées avant d'être traitées par une couche de convolution.

Le module SPP (figure 2.6) va incorporer des informations de contexte hiérarchique dans le réseau : il agrège les caractéristiques de plusieurs résolutions en une seule carte, riche en informations de contexte global et des sous-régions. L'implémentation du SPP est celle utilisée par le PSMNet (Chang & Chen (2018)) soit quatre blocs *average pooling* à différentes échelles : 64×64, 32×32, 16×16, 8×8. Chaque couche de *pooling* est suivie de convolution 1×1 pour ensuite agrandir les représentations par une interpolation bilinéaire à la dimension de l'entrée. Le bloc SPP produit en sortie la concaténation des différents niveaux de caractéristiques.
Nous allons aussi nous intéresser à l'ajout d'un bloc ASPP (Atrous Spatial Pyramid Pooling) (Chen, Papandreou, Kokkinos, Murphy & Yuille (2018a)) (figure 2.7) utilisé dans la tache de segmentation par Jha *et al.* (2019). Il capture aussi les informations de contexte à différentes résolutions mais en employant des atrous convolutions ayant différents taux de dilatation, à la

place des blocs *average pooling*. Les atrous convolutions permettent de contrôler le champ de vue et ainsi capturer précisément les informations à plusieurs échelles. Pour Jha *et al.* (2019), le module ASPP joue le rôle du pont reliant l'encodeur avec le décodeur.

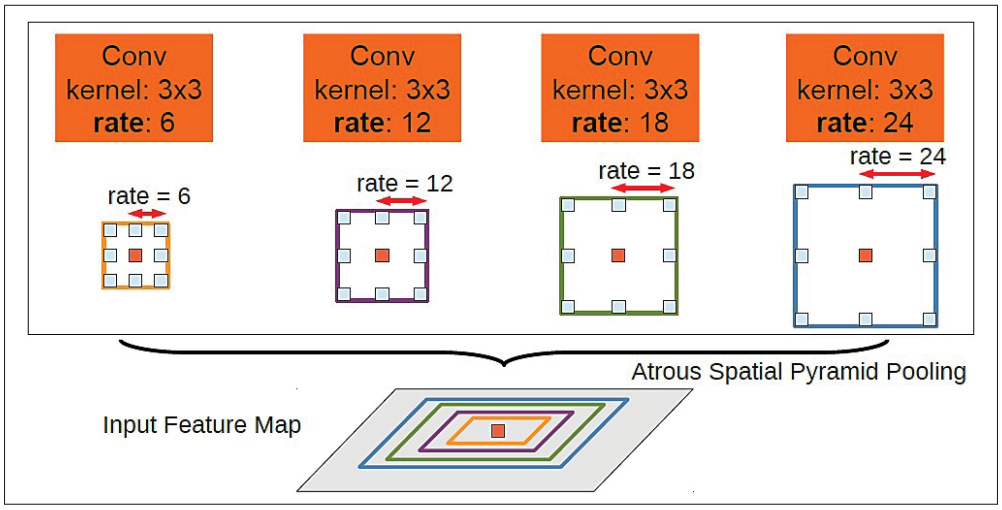

FIGURE 2.7 Schéma du bloc ASPP*
Tirée du DeepLab (Chen *et al.* (2018a))

*Le module ASPP extrait les informations multi-résolutions en appliquant des atrous convolution (*Conv* de kernel 3×3) avec différents taux de dilatation (*rate*). Le champ de vue de chaque filtre est indiqué en un carrée dans sa couleur.

2.2.2.3 Amélioration décodeur

Le décodeur réalise les prédictions multi-échelles à partir des informations apprises dans l'encodeur. Les *skip connection* permettent d'apprendre des relations entre les caractéristiques de bas niveau et celles de haut niveau. Afin d'améliorer l'efficacité du modèle et la qualité des caractéristiques utilisées pour les prédictions, plusieurs types de blocs d'attention sont employés. Ils concentrent l'apprentissage sur les informations essentielles en leur donnant plus d'importance. Jha *et al.* (2019) introduisent un module d'attention spatial (figure 2.8) dans le décodeur de sorte que le modèle renforce les régions issues de la *skip connection* essentielles à la prédiction.

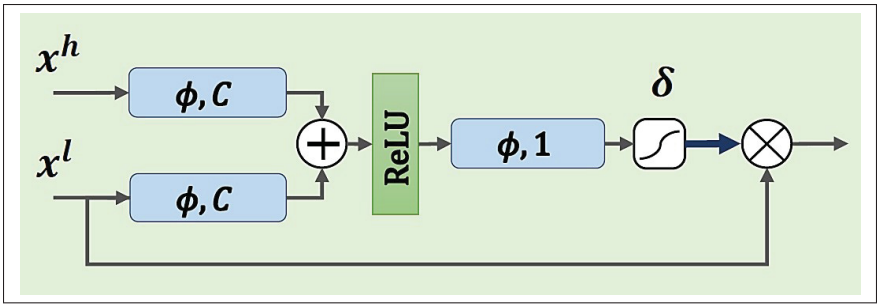

FIGURE 2.8 Bloc d'Attention spatial*
Tirée de Gu *et al.* (2021)

* x^h est issue de la *skip connection* et x^l du décodeur. Les régions utiles de x^l sont renforcées, et les régions non pertinentes sont supprimées.

Gu *et al.* (2021) proposent des modules d'attention pour les channels et pour les prédictions multi-résolutions (figure 2.9). L'attention au niveau des channels se présente comme un module Squeeze&Excite. Il est placé dans les blocs *iconvi*, juste après la concaténation, pour calibrer les informations bas niveau avec celles de haut niveau, donnant plus d'importance aux channels essentielles pour la prédiction. On va aussi étudier l'attention sur les prédictions multi-résolutions. Ce module met en avant les caractéristiques de la résolution la plus pertinente. Il va traiter une concaténation des différentes cartes de disparité mises à la dernière résolution, et produire en sortie d'une convolution 1×1 la prédiction finale. Pour le DispNetC, sa prédiction finale est remplacée par la sortie du bloc d'attention sur les résolutions, tandis que pour la version améliorée du DispNetC, la sortie du bloc d'attention est ajoutée après la dernière prédiction. De plus, des couches BatchNorm sont ajoutées après les déconvolutions, ce qui va accélérer la convergence du modèle (Azad *et al.* (2019)).

L'architecture finale de la version améliorée du DispNetC est présentée par les figures 2.10 pour l'encodeur et 2.11 pour le décodeur.

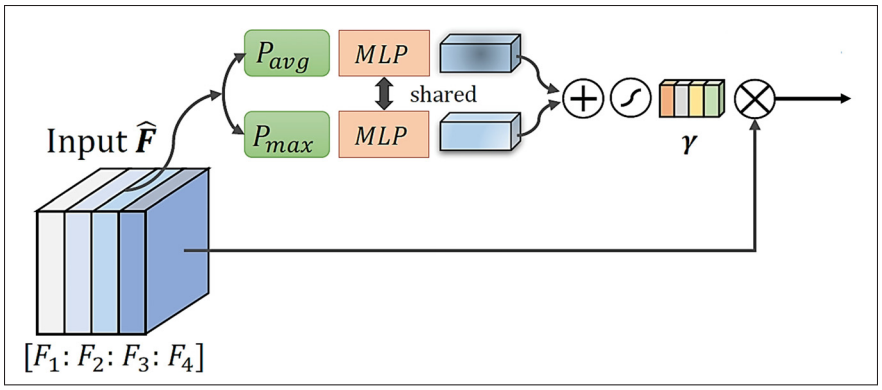

FIGURE 2.9 Bloc d'Attention sur la résolution*
Tirée de Gu *et al.* (2021)

*Les cartes de différentes résolutions sont concaténées et les couches de *maxpooling* et *average pooling* met en avant les résolutions les plus pertinentes pour la prédiction finale

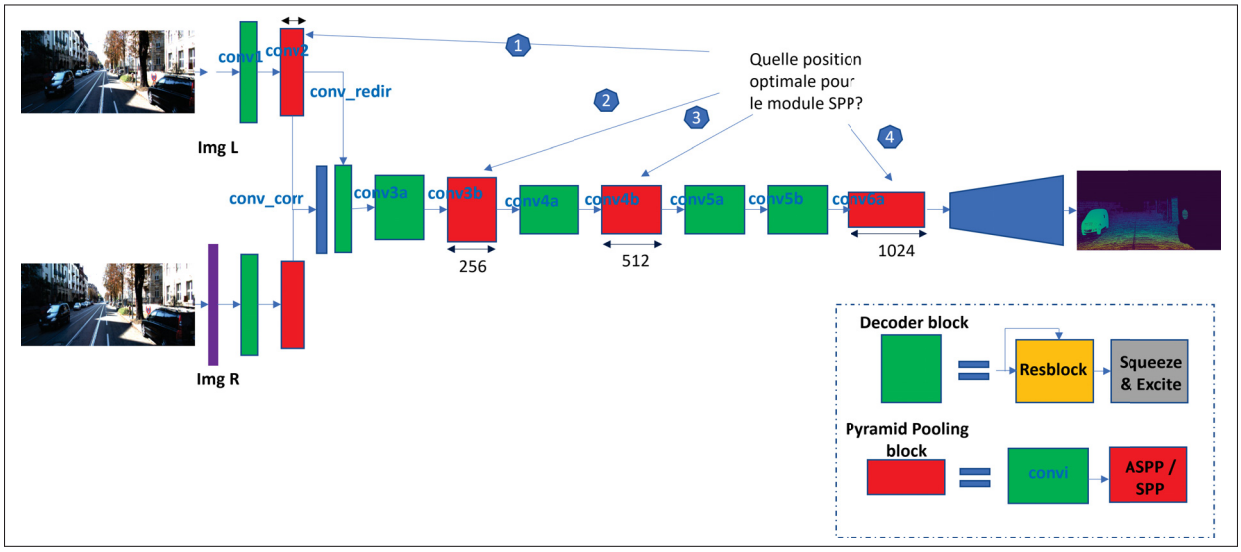

FIGURE 2.10 Schéma Amélioration DispNetC : détail de l'encodeur*

*Différentes positions des modules SPP et ASPP sont étudier pour déterminer celle optimale.

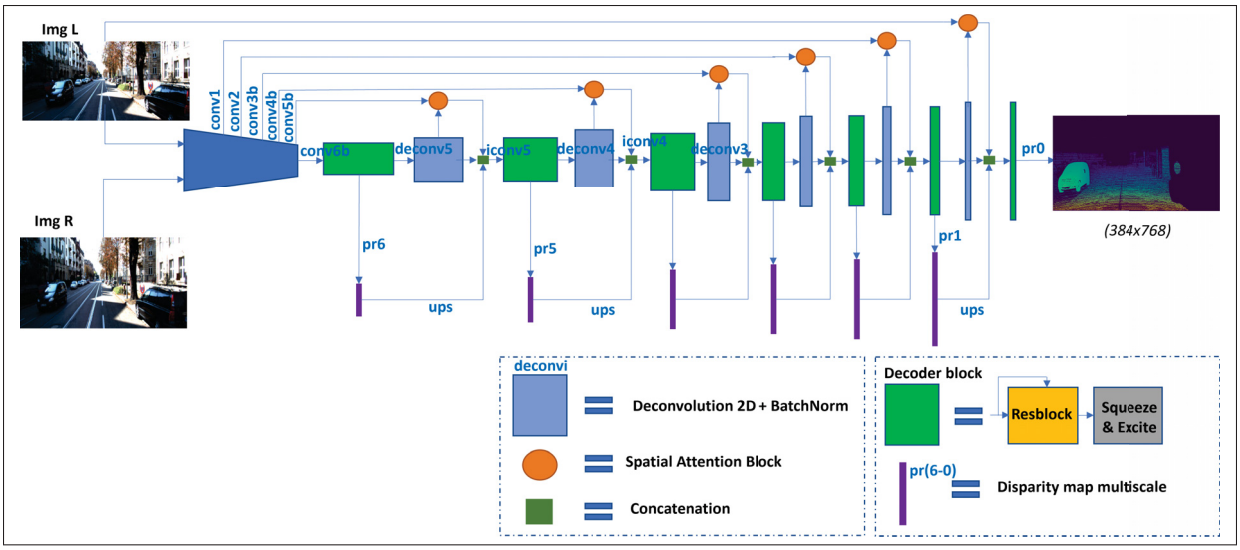

FIGURE 2.11 Schéma Amélioration DispNetC : détail du décodeur*

*Le bloc d'attention sur la résolution n'est pas représenté.

2.3 Résultats des expériences

2.3.1 Bases de données

SCENEFLOW DATASET. En s'inspirant de la méthode initiale de Mayer & Ilg (2016), FlyingThings3D (Mayer *et al.* (2016)) (+20000 images stéréo synthétiques d'objets du quotidien, volant suivant une trajectoire aléatoire) est utilisée comme base pour le préapprentissage du modèle à l'estimation des disparités.

KITTI. KITTI2015 (Menze *et al.* (2015)) contient 200 images stéréo réelles pour l'apprentissage (150 pour l'entraînement, 50 pour la validation) utilisée pour affiner le modèle sur la base KITTI2015. KITTI2012 (Geiger *et al.* (2012)) contient 194 images stéréo réelles pour l'apprentissage (146 pour l'entraînement, 48 pour la validation) utilisée pour affiner le modèle sur cette base.

2.3.2 Détails de l'implémentation

Le réseau est entraîné à partir de zéro sur la partie FlyinThings3D de Sceneflow (Mayer *et al.* (2016)), puis facultativement affiné sur les bases de données KITTI (Geiger *et al.* (2012); Menze *et al.* (2015)). L'implémentation de la méthode et l'apprentissage du modèle sont faits avec Pytorch. ADAM (Kingma & Ba (2015)) est employé pour l'optimisation avec comme paramètre ($\beta_1 = 0,95\,; \beta_2 = 0,999$).
Les implémentations des modules d'Attention sont celles employées par Gu *et al.* (2021). Les architectures des modules ASPP, Resblock et Squeeze&Excite sont celles utilisées par Jha *et al.* (2019). L'implémentation du SPP est quant à elle issue du PSMNet de Chang & Chen (2018).

Apprentissage. Le taux d'apprentissage est fixé à 0,0001 avec un batch de quatre durant 20 époques, puis est divisé par deux toutes les 10 époques. Comme le réseau de base produit en tout six cartes de disparité de résolutions différentes, on applique une méthode de planification des poids pour la fonction coût multi-résolution 2.2, ce qui permet d'éviter de mixer les gradients lors de l'optimisation de chaque résolution. Ainsi au début de l'apprentissage du DispNetC, on assigne un poids de $\omega_6 = 1$ pour la fonction coût de plus faible résolution $loss_6$ et de $\{\omega_i\} = 0$ pour les autres. Au cours de l'apprentissage, on augmente progressivement le poids des fonctions de haute résolution et désactive les fonctions de faible résolution, jusqu'à atteindre un poids de $\omega_1 = 1$ pour la $loss_1$ et de $\{\omega_i\} = 0$ pour les autres. Cela permet au réseau d'apprendre en premier lieu les informations globales de la scène pour ensuite se concentrer sur les détails plus fins, d'une manière *coarse-to-fine* (Mayer & Ilg (2016)). Lors de l'inférence, on utilise seulement la dernière carte produite soit $\omega_1 = 1$ et $\{\omega_i\} = 0$ pour $i \in [2;6]$.
Pour les versions améliorées du DispNetC, on considère une à deux fonctions coût supplémentaires en fin de réseau (7 prédictions au totale ou 8 prédictions en ajoutant le bloc d'attention sur les résolution). Lors de l'inférence, on aura $\omega_0 = 1$ et $\{\omega_i\} = 0$ pour $i \in [1;6]$ ou $i \in [1;7]$.

Outils d'évaluation. Nous allons évaluer les performances des méthodes proposées sur les disparités par les mesures EPE (mesure l'erreur de disparité moyenne par pixel), et D1 (pourcentage de pixels de disparité erroné). On utilise D1-all calculant l'erreur pour tous les

pixels de la vérité terrain. Plus ces mesures sont basses, meilleurs sont les résultats. La mesure D1-all est utilisée pour les bases KITTI tandis que la mesure EPE est utilisée pour SceneFlow.

Configurations. On va comparer les méthodes en réalisant un préapprentissage sur la base Sceneflow, puis en affinant si besoin le modèle sur la base KITTI2012 ou KITTI2015.
On présente 2 nouvelles versions du DispNetC : le DispNetCv2 (version du DispNetC avec les améliorations globales mais de même profondeur) et DispNetCv3 (version du DispNetC plus profonde avec les améliorations globales). Chaque approche est représentée par le ou les modules employés (SPP, ASPP, ATT_Scale pour attention sur la résolution, ATT_spatial pour attention spatiale) et leurs positions *j* dans le réseau (voir figure 2.10 et 2.11).

Augmentation de données. On réalise une normalisation des couleurs sur l'ensemble des données source et cible. Durant l'apprentissage, les images sont aléatoirement recadrées aux dimensions 768×384, et subissent des modifications aléatoires de couleurs, contraste et de luminosité selon la fonction ColorJitter de Pytorch.

2.3.3 Résultats

SceneFlow. Les résultats des améliorations et des méthodes de la littérature sur la base SceneFlow sont présentés dans le tableau 2.4. Le modèle $DispNetCv3\_SPP4 + \_ATT\_scale$ est l'amélioration possédant l'erreur la plus faible sur la base SceneFlow avec $EPE = 0,80$ réduisant ainsi d'environ 35% l'erreur du modèle de base. Il est suivi par le $DispNetCv3\_SPP4$ ($EPE = 0,84$) puis du $DispNetCv3\_SPP3$ ($EPE = 0,86$).
On peut voir que les améliorations globales permettent de réduire l'erreur d'au moins 14% (DispNetCv2) par rapport à la version initiale. Chaque module SPP et ASPP assure un gain en précision, mais la version SPP=4 est la plus performante. La techniques d'attention d'échelle permettent aussi de produire des cartes de disparité plus précise en réduisant l'erreur d'environ 7%. Cependant, l'attention spatiale tend à faire légèrement diminuer la précision ($DispNetCv3\_ATT\_spatial$ avec $EPE = 1,05$ tandis que $DispNetCv3$ avec $EPE = 1,01$).

Tableau 2.4 Résultats des méthodes de disparité sur SceneFlow. L'erreur EPE en pixel est indiquée dans le tableau

Méthode	**EPE (px)**
DispNetC	1,24
PSMNet	**1,09**
DispNetCv2	1,07
DispNetCv2_SPP1	1,01
DispNetCv3	1,01
DispNetCv3_SPP1	0,95
DispNetCv3_SPP2	0,93
DispNetCv3_SPP3	**0,86**
DispNetCv3_SPP4	**0,84**
DispNetCv3_ASPP1	0,87
DispNetCv3_ASPP4	0,88
DispNetCv3_ATT_spatial	1,05
DispNetCv3_SPP4+ATT_spatial	0,94
DispNetCv3_ATT_scale	0,93
DispNetCv3_SPP4+_ATT_scale	**0,80**
DispNetCv3_SPP4+ATT_spatial+scale	0,89

KITTI 2015 et KITTI 2012. Les résultats sur les bases KITTI 2012 et KITTI 2015 des améliorations et des méthodes de la littérature sont présentés dans le tableau 2.5. L'ajout des couches *deconv0, iconv0, pr0* dans le DispNetCv3 permet de réduire l'erreur à moins de 4%(D1) sur KITTI2015 comparé à la méthode DispNetCv2. Le modèle $DispNetCv3\_SPP4 + \_ATT\_scale$ possède l'erreur la plus faible sur les bases KITTI et permet de descendre D1 en-dessous des 3% sur la KITTI2015, et de s'approcher des 2% d'erreur sur KITTI 2012. Il réduit ainsi d'environ 33% et 48% l'erreur du modèle de base sur KITTI 2015 et KITTI 2012 respectivement. Il est suivi des versions $DispNetCv3\_SPP4$ et $DispNetCv3\_ASPP4$.
Comme sur SceneFLow, les différents modules vont permettre de diminuer l'erreur de prédiction. Les architectures SPP=4 et ASPP=4 se démarquent fortement des autres configurations en avoisinant les 2% d'erreurs. Cependant, l'attention spatiale tend à limiter la précision du modèle comme le montre le résultat du $DispNetCv3\_SPP4 + ATT\_spatial$.

Généralisation. Évaluer la robustesse des modèles face au changement de domaine est important dans le cadre de notre problème. En effet, nous allons chercher par la suite une méthode d'adaptation de domaine à appliquer pour l'estimation des disparités. Ainsi, il faut choisir au préalable une méthode pouvant généraliser les caractéristiques apprises sur différents domaines.

Tableau 2.5 Résultats des méthodes de disparité sur les bases KITTI. L'erreur D1-all(%) est indiquée dans le tableau[a]

Méthode	**KITTI 2015**	**KITTI 2012**
DispNetC-K	10,97	9,45
DispNetC	4,41	4,04
PSMNet	**1,83**	**1,11**
DispNetCv2_SPP1	4,03	2,65
DispNetCv3	3,81	2,48
DispNetCv3_SPP1	3,64	2,40
DispNetCv3_SPP2	3,53	2,34
DispNetCv3_SPP3	3,03	2,15
DispNetCv3_SPP4	**2,98**	**2,14**
DispNetCv3_ASPP1	3,11	2,20
DispNetCv3_ASPP4	**3,01**	**2,11**
DispNetCv3_ATT_spatial	3,97	2,73
DispNetCv3_SPP4+ATT_spatial	3,79	2,41
DispNetCv3_ATT_scale	3,74	2,42
DispNetCv3_SPP4+_ATT_scale	**2,92**	**2,09**
DispNetCv3_SPP4+ATT_spatial+scale	3,71	2,34

[a] Sauf DispNetC-K qui est seulement entraîné sur les bases KITTI, les méthodes ont été pré-entrainées sur FlyingThings, puis affinées sur les bases KITTI avant d'être évalué.

Le tableau 2.6 présente les résultats D1-all sur les bases KITTI, des méthodes entraînées seulement sur SceneFlow. On peut voir que le modèle $DispNetCv3\_SPP4 + \_ATT\_scale$ assure les meilleurs résultats de généralisation sur les KITTI.

La version initiale ne présente pas de bonne performance en généralisation (l'erreur est multipliée par quatre par rapport à l'apprentissage supervisé). Les techniques proposées permettent d'améliorer la robustesse du réseau comme le montre les résultats sur KITTI. Les améliorations globales diminue l'erreur d'environs 24%. Les modules d'agrégation de contexe SPP et ASPP vont ajouter en précision en allant jusqu'à $D1 = 11,31\%$ pour SPP=4. Le module d'attention spatiale augmente l'erreur de prédiction.

ResBlock et Squeeze&Excite. Les améliorations globales Resblock et Squeeze&Excite dans le DispNetCv2 et le DispNetCv3 permettent un apprentissage plus rapide et robuste des disparités, ce qui assure une prédiction plus détaillées et donc d'atteindre une meilleure précision. Le bloc Squeeze&Excite active les caractéristiques essentielles et supprime celles pouvant ajouter de l'erreur. Les blocs résiduels ont permis d'agrandir la profondeur du réseau DispNetC tout en ayant une meilleure efficacité lors de l'apprentissage (réduction de 40% de la durée d'apprentissage). On atteint ainsi de meilleures performances avec le DispNetCv3, pour un temps de prédiction légèrement supérieur. De plus, les améliorations globales augmentent la capacité de généralisation du modèle, comme le montre les résultats de DispNetCv2 et DispNetCv3.

SPP et ASPP. Les modules SPP, ASPP augmentent considérablement la précision du réseau en lui fournissant les informations nécessaires à la tache ainsi que les relations hiérarchiques entre elles, comme le montre les résultats sur Sceneflow et KITTI (voir figure 2.12). Les informations locales et de contexte sont très importantes pour traiter la tache de disparité. En enrichissant la représentation par ces modules, le gain en précision devient non négligeable.
On observe des améliorations dans les prédictions, au niveau des frontières d'objets mais aussi pour les objets lointain. Les performances de la configurations SPP=4 est la meilleure car elle permet d'agréger sous plusieurs résolutions toutes les informations extraites dans l'encodeur (couleur, forme, profondeur), avant de les transmettre au décodeur pour la prédiction.
De plus, les résultats de généralisation montrent que SPP et ASPP permettent d'apprendre des représentations invariantes au domaine, et donc plus robuste.

Spatial et Scale Attention. Le bloc d'attention sur les résolutions a montré de très bonnes performances à travers les expériences menées sur Sceneflow et KITTI. Il permet de faire contribuer toutes les cartes de résolutions à la prédiction finale. Il garde dans chaque carte de résolution les caractéristiques les plus importantes et invariantes au domaine, pour produire à la fin une carte de disparité plus précise (voir figure 2.13).
Le module d'Attention spatial a pour but de guider les informations utiles bas niveaux (issues de la *skip connection* avec l'encodeur), vers les prédictions multiscale. Cependant, les expériences ont montré que l'attention spatiale avait tendance à faire diminuer les performances du modèle (voir

figure 2.12). Le bloc d'attention spatiale apprend un filtre permettant d'activer les informations spatiales utiles à la prédiction. Ce filtre dépend des informations de la *skip connection* et de la couche le précédant. On peut voir alors que les caractéristiques spatiales issues de l'encodeur vont perturber la prédiction *multiscale*. A l'inverse des modules proposés où l'on emploie une méthode de *pooling* pour alléger les contraintes, le module d'attention spatial va regrouper les informations importantes et non importantes pour générer le filtre spatial. De plus, contrairement aux méthodes de segmentation, les informations bas niveaux contenues dans l'image ne sont pas suffisantes pour prédire une carte de disparité. Il est nécessaire de connaître la relation de profondeur entre les deux images encodées dans les channels. Ainsi, l'attention spatiale va pousser le réseau à traiter les channels de la même manière, ce qui crée de l'erreur dans la prédiction.

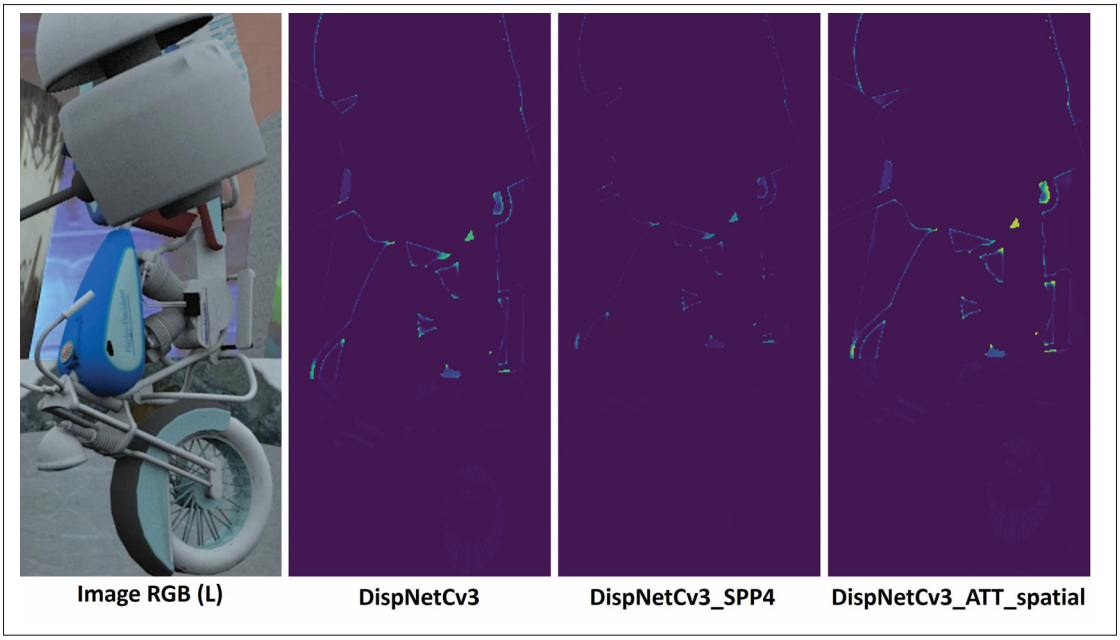

FIGURE 2.12 Cartes d'erreur de disparité pour différentes configurations sur SceneFlow *

*Les cartes indiquent l'erreur absolue relative avec la carte *ground-truth*. Plus le pixel est jaune et plus l'erreur est importante. On peut voir que les améliorations SPP ajoutent de la précision au niveau des frontières et des petites régions avec de grande valeur de disparité. Le module d'attention spatiale va au contraire ajouter de l'erreur dans ces régions.

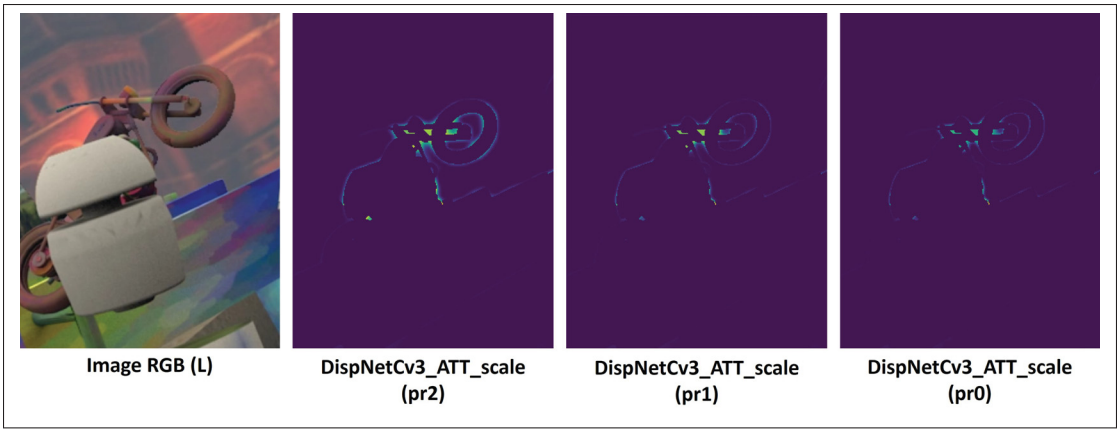

FIGURE 2.13 Cartes d'erreur de disparité produites par le *DispNetCv3_ATT_scale* à différentes échelles de prédiction sur SceneFlow *

*Les trois cartes ont été générées par des couches de résolutions différentes (pr2, pr1, pr0). Elles indiquent l'erreur absolue relative avec la carte *ground-truth*. Plus le pixel est jaune-rouge et plus l'erreur est importante. On peut voir que les prédictions multiscale contribuent à la prédiction finale, en minimisant progressivement des régions d'erreurs.

Mémoire et temps de prédiction. Le tableau 2.7 regroupe les informations d'empreinte mémoire et de vitesse des modèles lors d'une prédiction, obtenues sur la carte GPU Geforce GTX 1080Ti 11Gb. Premièrement, on peut voir que l'utilisation d'une carte graphique avec de meilleures performances que celle utilisée dans le papier de Mayer & Ilg (2016) (11,34 TFLOPs contre 6,69 TFLOPs selon le tableau 2.1) permet de passer d'une vitesse de prédiction de 20 f/s à 52,6 f/s pour le modèle de base. Ce gain en temps est non-négligeable car il permet d'appliquer des améliorations au DispNetC tout en ayant une prédiction en temps réel. Seulement le DispNetCv3_SPP1 et les amélioration utilisant le bloc d'attention spatial ont un temps de prédiction inférieur à 16 f/s. Le module SPP=1 est placé sur l'entrée siamoise ce qui va doubler le temps pris par le module dans le modèle.

De plus, la contrainte du cahier des charges de coût en calcul lors de la prédiction est respectée pour presque chaque modèle. La carte graphique utilisée permet de diminuer l'empreinte mémoire du modèle de base par rapport au papier d'origine, ce qui assure aux autres versions d'avoir une consommation en dessous des 1,5 Go.

Tableau 2.6 Résultats des des améliorations du DispNetC pour la généralisation[a]. L'erreur D1-all(%) est indiquée dans le tableau

Méthode	**KITTI 2015**	**KITTI 2012**
DispNetC	16,13	13,91
DispNetCv2	12,23	9,88
DispNetCv2_SPP1	**11,54**	**9,37**
DispNetCv3	12,18	10,41
DispNetCv3_SPP1	11,74	9,71
DispNetCv3_SPP2	13,45	11,35
DispNetCv3_SPP3	11,73	9,49
DispNetCv3_SPP4	**11,31**	**9,25**
DispNetCv3_ASPP1	12,07	9,76
DispNetCv3_ASPP4	11,68	9,48
DispNetCv3_ATT_spatial	17,60	14,34
DispNetCv3_SPP4+ATT_spatial	21,82	16,28
DispNetCv3_ATT_scale	12,06	10,32
DispNetCv3_SPP4+_ATT_scale	**11,17**	**9,21**
DispNetCv3_SPP4+ATT_spatial+scale	21,65	16,13

[a]Les méthodes ont été pré-entraînées sur FlyingThings puis testées sur les bases KITTI.

Tableau 2.7 Comparaison du coût de calcul et de la vitesse lors de la prédiction avec les méthodes de la littérature, sur une carte GPU Geforce GTX 1080Ti 11Gb

Méthode	**Temps de prédiction (f/s)**	**Mémoire (Go)**
PSMNet	2,4	4,5
DispNetC	52,6	1,0
DispNetCv2	21,7	1,2
DispNetCv3	18,2	1,3
DispNetCv3_SPP1	15,2	1,4
DispNetCv3_SPP4	17,5	1,3
DispNetCv3_ASPP4	17,2	1,4
DispNetCv3_ATT_spatial	16	1,5
DispNetCv3_SPP4+ATT_spatial	13,9	1,5
DispNetCv3_ATT_scale	18,1	1,3
DispNetCv3_SPP4+ATT_scale	17,4	1,3
DispNetCv3_SPP4+ATT_spatial+scale	14	1,5

2.4 Conclusion : choix de l'architecture optimale

Nous avons pu voir à travers les expériences l'efficacité des modules de Spatial Pyramid Pooling, Resblock et Squeeze&Excite. Ils permettent d'améliorer la précision et robustesse du modèle tout en assurant une empreinte mémoire et vitesse de prédiction respectant le cahier des charges. De plus, les résultats ont montré que le bloc d'attention sur les résolutions est très efficace pour sélectionner les caractéristiques importantes sur chaque résolution, mais que le bloc d'attention spatial est moins adapté au problème d'estimation des disparités. Le modèle qui sera sélectionné pour les prochaines expériences est le $DispNetCv3\_SPP4 + ATT\_scale$, qu'on renommera DispNetCv4, qui assure les meilleures performances. Qualitativement, les figures 2.14 et 2.15 montrent les résultats qualitatives du modèle de base par rapport à la version améliorée. On peut voir que les prédictions des améliorations sont plus détaillées au niveau des formes des objets, des frontières, et des objets lointains.

Dans la suite, nous allons employer une approche d'adaptation de domaine par apprentissage adverse et par translation d'image permettant d'améliorer la généralisation du domaine sur les bases réelles KITTI.

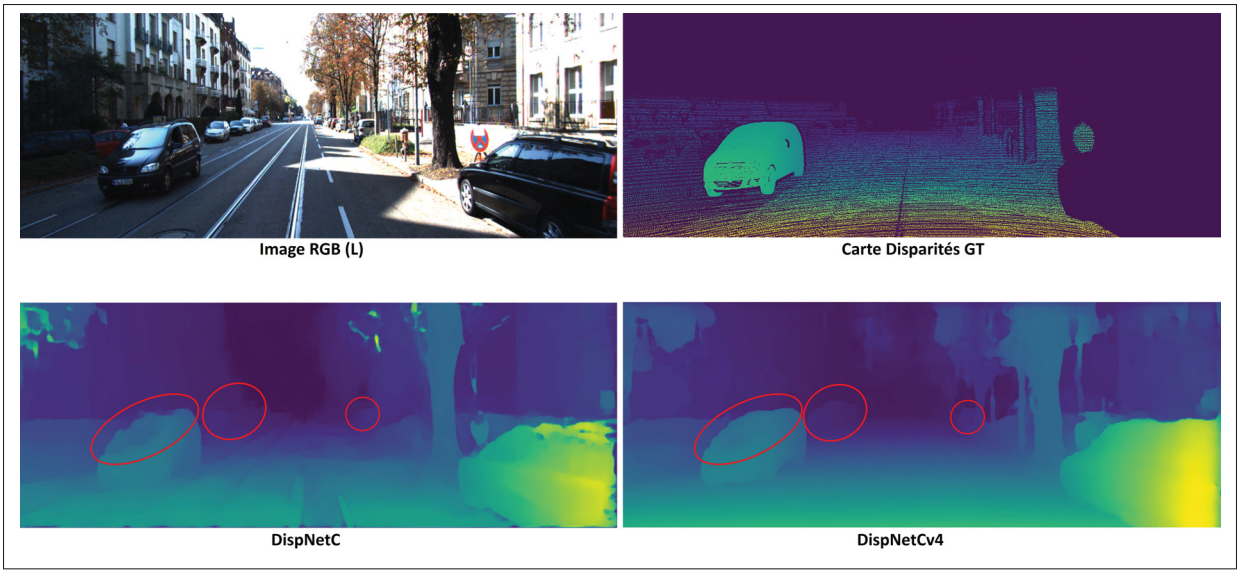

FIGURE 2.14 Résultats des cartes de disparités prédites sur la base KITTI 2015*

*Il est entouré en rouge les différences de prédiction entre le DispNetC et le DispNetCv4. On peut voir que les améliorations affinent les prédictions. (*GT : ground-truth*)

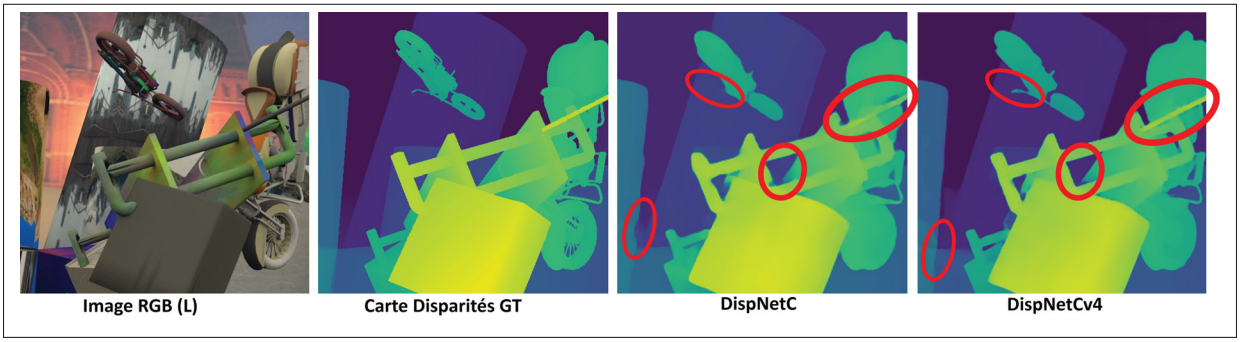

FIGURE 2.15 Résultats des cartes de disparités prédites sur la base SceneFlow*

*Il est entouré en rouge les différences de prédiction entre le DispNetC et le DispNetCv4.

CHAPITRE 3

ADAPTATION DE DOMAINE POUR L'ESTIMATION DES DISPARITÉS

3.1 Mise en contexte : translation d'image et adaptation de domaine

L'adaptation de domaine non-supervisée a pour objectif d'entraîner un modèle à être robuste sur le domaine cible à l'aide de données d'apprentissage riche en label et de données cible sans label, afin de diminuer les chutes de performances provoqués par le changement de domaine. Cette approche est d'autant plus utile dans le cas où il est indispensable d'avoir un domaine source ayant une distribution différente du domaine cible, et quand un simple apprentissage sur le domaine cible n'est pas suffisamment ou impossible.

Dans la tâche d'estimation de disparité par images stéréo, très peu de bases de données réelles annotées sont disponibles. Et celles qui sont accessibles possèdent un nombre limité d'images (KITTI2015 possède seulement 200 images pour l'apprentissage) ce qui ne permet pas d'entraîner de manière efficace et robuste des modèles profonds avec des millions de paramètres. Cette faible quantité de données est entre autre due au processus d'annotation qui est long et fastidieux. Adapter le réseau au domaine semble donc une alternative intéressante pour les points soulevés précédemment : face au manque de données réelles annotées, il est nécessaire d'entraîner le modèle sur des données synthétiques. Une approche triviale serait de pré-entraîner le modèle sur le domaine source et puis affiner ses paramètres sur les images réelles disponibles. Cependant une telle approche ne permet pas au réseau d'apprendre une relation entre deux domaines très éloignés. Et le peu de données cibles disponibles ne couvre pas l'ensemble de données appartenant à l'espace du domaine cible. Les méthodes de la littérature vont donc plutôt appliquer les principes de reconstruction d'image et/ou d'apprentissage adverse, afin de réduire l'écart entre les distributions source et cible, voir les confondre.

3.1.1 L'apprentissage adverse pour la translation d'image

L'apprentissage adverse fait référence à toutes méthodes utilisant un processus d'opposition durant l'apprentissage. Avant l'apparition des méthodes adverse, ce terme désignait les méthodes d'apprentissage améliorant la robustesse d'un modèle par l'utilisation d'exemple contradictoire : au sein du set d'apprentissage, on ajoute à certaines images une petite perturbation (vecteur bruit, patch noir, . . .) modifiant très peu l'image mais amenant à une mauvaise classification (Wilson & Cook (2020)).
En suivant ce principe d'apprentissage adverse, Goodfellow *et al.* (2014) ont proposé le GAN (*Generative Adversarial Network*). Leur approche était appliquée traditionnellement pour la génération d'image synthétique. S'en suit alors la publication d'une centaine de papier sur le sujet pour différentes applications, dont l'adaptation de domaine. La méthode GAN traditionnelle optimise en alternance deux modèles, dans l'objectif de produire des fausses images similaires à celles de la base d'apprentissage. Le principe est le suivant (voir figure 3.1) : Un générateur G reçoit en entrée un vecteur bruit aléatoire z tiré d'une distribution normal ou uniforme. Son but est alors de produire une image indiscernable d'une vraie image d'apprentissage. Un discriminateur D va recevoir soit une vraie image d'apprentissage soit une fausse générée par G. Son but à lui est de déterminer la probabilité que l'image en entrée est vraie. Le discriminateur est entraîné d'une manière supervisée où le label est le domaine d'origine de la donnée. Durant l'apprentissage, les deux réseaux jouent à un jeu de mini-max, où le générateur cherche à tromper le discriminateur tandis que ce dernier essaye de discerner l'image vraie de la fausse. Cela se traduit par l'introduction de la fonction coût du discriminateur dans celle du générateur, en inversant les labels (similaire à la couche d'inversement de gradient).

Avec le succès du GAN, les chercheurs ont développé des méthodes employant le GAN pour la translation d'image vers image, dont le but est de mapper un domaine à l'autre. Le mappage est typiquement créé à l'échelle du pixel par un générateur qui translate une image source en entrée vers une image semblable à la distribution du domaine cible. Les données source translatées avec leurs labels vont permettre d'entraîner un modèle à une tache sur le domaine cible. Puis les vraies données cible non labelisées vont servir à évaluer le modèle.

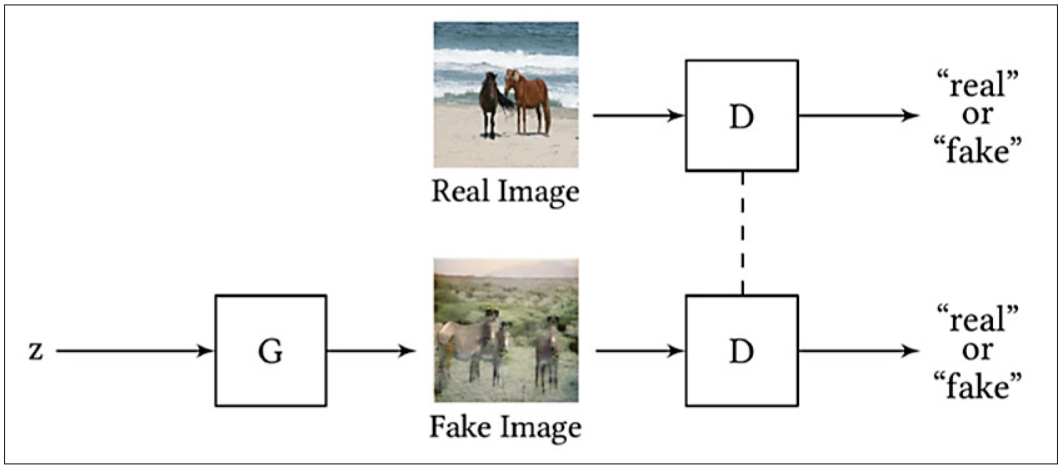

FIGURE 3.1 Principe du GAN pour la génération d'image synthétique à partir d'un générateur G et d'un discriminateur D*
Tirée de Wilson & Cook (2020)

*La ligne en pointillé indique que les poids sont partagés.

Par définition, le GAN est non-conditionnel car considère seulement un vecteur bruit en entrée. Mais dans le cas où il faut mapper un domaine vers un autre, il faut considérer une image en entrée et la distribution du domaine dont elle appartient. Isola *et al.* (2017) ont proposé le GAN conditionnel (cGAN), qui considère en entrée tout type d'information (image, classe, labels. . .) pour translater une image d'un domaine source vers un domaine cible. Cependant, cette approche est seulement utilisée dans une translation supervisée, où l'on considère des paires d'images source et cible. Ainsi, Le générateur transforme une image source en une image cible, et le discriminateur différencie l'image générée de l'image réelle à l'aide de l'image source en entrée (voir figure 3.2).

Dans notre cas d'étude, les bases de données source et cible ne sont pas appariées : les méthodes exploitent des images synthétiques issues de la base SceneFlow, dont le contenu est très éloigné des images de la base KITTI. Ainsi, pour pouvoir appliquer une méthode de translation d'image en apprentissage adverse pour la tache d'estimation des disparités, il est nécessaire de se placer dans un cas de translation d'image non supervisée. On considère alors des méthodes où les images source et cible sont non-appariées (voir figure 3.3).

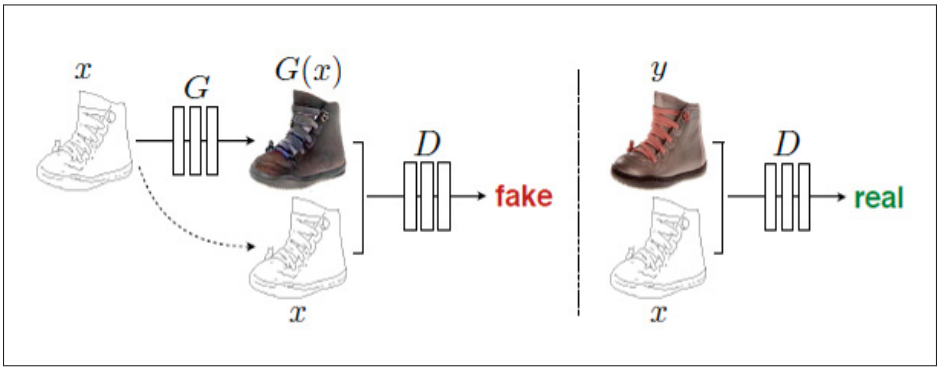

FIGURE 3.2 Principe du GAN conditionnel pour la translation d'image de croquis vers photo réaliste
Tirée de Isola *et al.* (2017)

Différentes méthodes de la littérature se sont intéressées à ce problème : obtenir des pairs d'images de deux domaines est difficile et coûteux. Le SimGAN est proposé par Shrivastava *et al.* (2017) afin de translater les images synthétiques dans le domaine réel pour l'estimation de la direction du regard et de la position de la main. En plus de l'apprentissage adverse, une fonction coût de self-régularisation est employée pour la translation non-supervisée. Cependant, cette approche est limitée à des petits changement de domaine. Le cycleGAN développé par Zhu *et al.* (2017) décide de réaliser la translation d'images non-appariées à l'aide de l'apprentissage adverse associé à une fonction coût de cohérence cyclique. Cette fonction permet d'éviter d'avoir des données source et cible appariées : après avoir translaté l'image d'un domaine à l'autre, elle est translatée dans le sens inverse afin de reconstruire l'image initiale et la différence entre l'image originale et sa reconstruction est minimisée. De leur coté, Zheng *et al.* (2018) propose le t2Net utilisant une fonction coût de reconstruction sur les images cible en plus de la fonction adverse, pour prédire des cartes de profondeur à partir d'image monoculaire. Dans ce mémoire, nous allons exploiter le CycleGAN qui est une méthode très répandue pour la translation d'images non-appariées.

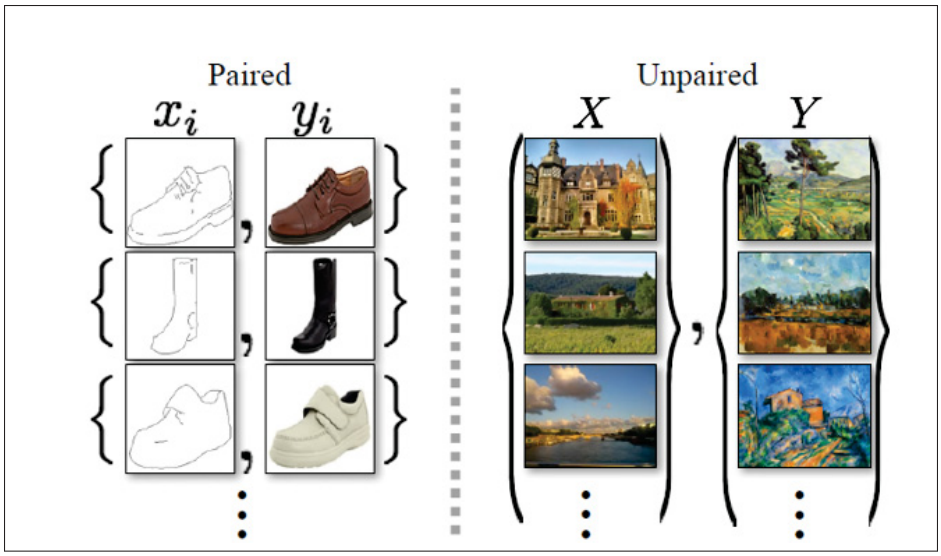

FIGURE 3.3 Illustration d'images appariées (gauche) et non appariées (droite)*
Tiré de Zhu *et al.* (2017)

*Les données sont appariées s'il existe une correspondance entre l'image source $x_i \in X$ et la donnée cible $y_i \in Y$ pour $i \in [1, N]$.

3.1.2 Objectifs

Notre but est d'entraîner un réseau estimant des cartes de disparité à partir d'images-stéréo synthétiques labélisées, tout en s'assurant qu'il puisse maintenir ses performances sur des images du domaine réel non annotées. C'est pourquoi dans la suite, la méthode CycleGAN de Zhu *et al.* (2017) sera appliquée pour répondre au problème de changement de domaine. Après l'apprentissage, le réseau de translation sera utilisé pour translater les images synthétiques dans le domaine réel (contrairement à Atapour-Abarghouei & Breckon (2018)), pour ensuite alimenter le modèle principal. En plus du réseau de translation, nous allons nous intéresser à adapter au domaine le réseau de disparité au niveau des caractéristiques intermédiaire. Inspiré des travaux de Zheng *et al.* (2018) et Kundu *et al.* (2018), différents discriminateurs de caractéristique seront évalués, pour rendre les représentations interne apprises par le réseau invariantes au domaine. La méthode d'apprentissage suivante est réalisée (voir figure 3.4) :

Un entraînement du réseau de disparité directement sur les images synthétiques n'étant pas suffisant pour assurer la robustesse du modèle sur le domaine réel, un réseau de translation

entraîné par une fonction coût GAN via un discriminateur va reconstruire dans le domaine réel, les images du domaine synthétique afin qu'elles paraissent plus réalistes. Ces images translatées vont permettre d'entraîner le réseau de disparité à faire des prédictions dans le domaine réel d'une manière supervisée. De plus, on va chercher aussi à ce que les représentations internes des caractéristiques du réseau de disparités partagent une distribution similaire pour les images réelles et translatées. Pour cela, un discriminateur des caractéristiques sera placé à une couche optimale interne au réseau afin de rendre les représentations apprises invariantes au changement de domaine.

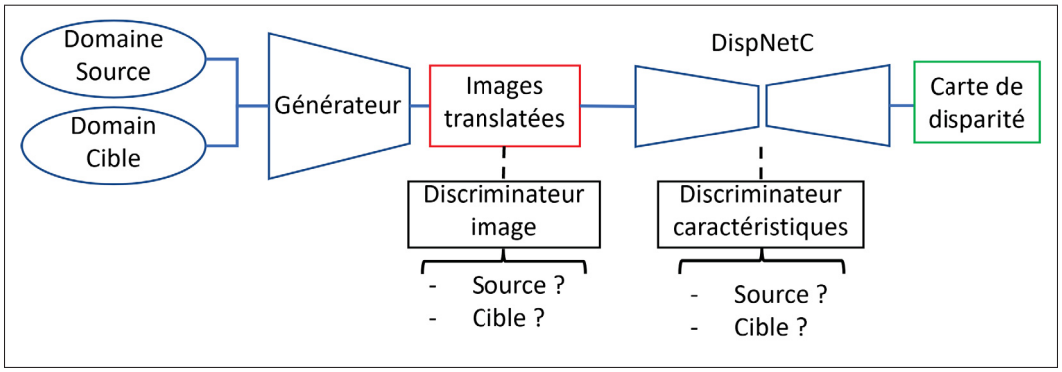

FIGURE 3.4 Illustration de la méthode d'estimation des disparités par adaptation de domaine*

*Les discriminateurs et le générateur sont seulement utilisés durant l'apprentissage.
Le générateur collecte une base d'images synthétiques translatées, puis cette base est utilisée pour entraîner le réseau de disparité. Ce dernier est en même temps adapté au domaine par le discriminateur intermédiaire.

3.2 Définition

Soit f_T le réseau estimant les cartes de disparité à partir d'images stéréo. Soit x = $\{x_L, x_R\}$ une paire d'image stéréo exemple. On a alors $y_D = f_T(x)$ la carte de disparité prédite par le réseau. En adaptation non-supervisée, on suppose X_s le domaine source de distribution $p_s(x, y_D)$ et X_r le domaine cible de distribution $p_r(x, y_D)$ avec $p_s \neq p_r$. On a $\{x_s, y_{D_s}\} \in X_s$ correspondant aux données sources et leurs labels respectivement ; et $\{x_r\} \in Xr$ les données cibles dont les labels sont inconnus. Dans notre cas, X_s est le domaine synthétique (indice 's' pour synthétique)

et (x_s, y_{D_s}) est l'ensemble des données synthétiques d'apprentissage composé d'images stéréo synthétiques $x_s = \{x_{s_L}; x_{s_R}\}$ (où x_{s_L} est l'image gauche et x_{s_R} l'image droite), et des cartes de disparité y_{D_s}. X_r est le domaine réel (indice 'r' pour réel) qui est composé de l'ensemble des images stéréo réelles $x_r = \{x_{r_L}; x_{r_R}\}$.

Définissons maintenant un réseau de translation $G_{s\to r}$ qui permet de transformer des données synthétiques x_s dans le domaine réel X_r. Ainsi, on a alors $G_{s\to r}(x_s) = \tilde{\mathrm{x}}_r \in Xr$. L'objectif de l'adaptation est d'apprendre un réseau f_T sur le domaine réel X_r tel qu'à partir d'une image $x_r \in X_r$, il puisse prédire correctement la carte de disparité y_{D_r} où $y_{D_r} = f_T(x_r)$. Comme les labels cibles ne sont pas disponibles, on va chercher à apprendre f_T sur le domaine source par adaptation pour que le modèle puisse être appliqué sur le domaine réel.

3.3 Méthodes

Une image réelle et une image synthétique sont différenciables avant tout grâce aux informations de couleur et texture (à l'échelle du pixel) plutôt qu'aux informations de forme et géométrie contenues dans l'image. Le rôle du réseau de translation est d'assurer une liaison entre les deux distributions, permettant ainsi de combler l'écart entre les domaines. Il doit produire à partir d'une image synthétique une image la plus réaliste possible, en conservant le contenu de la scène observée. Le principe de reconstruction associé à celui d'apprentissage adverse ont pour but d'entraîner un générateur $G_{s\to r}$ à reconstruire des images x_s du domaine synthétique X_s dans le domaine réel X_r. L'image générée $\tilde{\mathrm{x}}_r$ doit être indistinguable d'une vraie image réelle x_r. Pour cela, un discriminateur D_r est entraîné à différencier entre $\tilde{\mathrm{x}}_r$ et x_r en indiquant la vraie image réelle de celle translatée. Cette méthode de confusion de domaine suit le principe du GAN de Goodfellow *et al.* (2014).

On a alors deux modèles à entraîner en alternance :

- un générateur $G_{s\to r}$, qui essaye de reproduire la distribution du domaine cible à partir des données source. Il est entraîné à produire des données similaires au domaine cible qui vont tromper le discriminateur par la maximisation de son erreur ;
- un discriminateur de domaine D_r, qui classifie si la donnée a été produite par le générateur ou si elle est vraie.

Nous allons présenter par la suite la méthode utilisée pour la translation d'images non appariées.

3.3.1 Approche CycleGAN

L'approche CycleGAN de Zhu *et al.* (2017) a pour but d'effectuer la translation entre deux domaines différents dont les images sont non appariées. Soit les deux domaines X_s et X_r, de distribution p_s et p_r respectivement. Les fonctions de passage d'un domaine à l'autre sont représentées par deux générateurs séparés, $G_{s\to r}$ et $G_{r\to s}$ et de deux discriminateurs D_s (discrimine entre $x_s \in X_s$ et $G_{r\to s}(x_r)$) et D_r (discrimine entre $x_r \in X_r$ et $G_{s\to r}(x_s)$).
La fonction objective est composée d'une fonction adverse pour aligner la distribution des images générées avec celle du domaine cible, d'une fonction de cohérence cyclique évitant que les deux générateurs se contredisent et d'une fonction de reconstruction.

Fonction adverse

Chaque couple de générateur/discriminateur possède une fonction adverse qui assure que la distribution du domaine en entrée est transférée vers l'autre :

- Pour D_r et $G_{s\to r}$

$$\begin{aligned} L_{GAN}(G_{s\to r}, D_r) = min_{G_{s\to r}} max_{D_r} \mathbb{E}_{x_r\sim p_r}[\log D_r(x_r)] \\ + \mathbb{E}_{x_s\sim p_s}[\log(1 - D_r(G_{s\to r}(x_s)))] \end{aligned} \tag{3.1}$$

  où $G_{s\to r}$ va produire des images $\tilde{\mathrm{x}}_r$ similaire aux images du domaine cible X_r, et D_r tente de distinguer une vraie image cible x_r d'une générée. Ainsi, le générateur cherche à minimiser la fonction coût alors que le discriminateur essaye de la maximiser.

- D'une manière similaire on a pour D_s et $G_{r\to s}$

$$\begin{aligned} L_{GAN}(G_{r\to s}, D_s) = min_{G_{r\to s}} max_{D_s} \mathbb{E}_{x_s\sim p_s}[\log D_s(x_s)] \\ + \mathbb{E}_{x_r\sim p_r}[\log(1 - D_s(G_{r\to s}(x_r)))] \end{aligned} \tag{3.2}$$

Fonction de cohérence cyclique

Cependant il existe une infinité de fonction sur X_s mappant la même distribution sur X_r, ce qui n'assure pas une sortie unique $\tilde{x}_r$. De plus, il est difficile d'optimiser la fonction adverse seul : les procédures standards mènent souvent au problème de mode collapse, où toutes les images en entrée mappent vers la même image en sortie, échouant l'optimisation (Zhu *et al.* (2017)).
De manière à contraindre la fonction adverse pour que le générateur produise des images réelles conservant le contenu sémantique de leur versions originales, une fonction de cohérence cyclique est ajoutée en plus de la translation inverse $G_{r\to s}$. La fonction de cohérence pousse le générateur $G_{r\to s}$ à être capable de ramener une image $\tilde{x}_r$, qui a été translatée dans le domaine cible par $G_{s\to r}$, dans son état initiale du domaine X_s, en conservant les informations de structures. C'est-à-dire, après un cycle complet on doit avoir :

$G_{r\to s}(\tilde{x}_r) \approx x_s$ où $\tilde{x}_r = G_{s\to r}(x_s)$ soit $G_{r\to s}(G_{s\to r}(x_s)) \approx x_s$,

et vice versa.

Ainsi la fonction coût s'écrit :

$$\begin{aligned} L_{cyc}(G_{s\to r}, G_{r\to s}) = \mathbb{E}_{x_s\sim p_s}[\parallel G_{r\to s}(G_{s\to r}(x_s)) - x_s \parallel_1] \\ + \mathbb{E}_{x_r\sim p_r}[\parallel G_{s\to r}(G_{r\to s}(x_r)) - x_r \parallel_1] \end{aligned} \tag{3.3}$$

Fonction de reconstruction

On ajoute de plus une fonction de reconstruction pour chaque générateur pour s'assurer que lorsqu'une image du domaine de sortie est fournie en entrée, celle-ci reste inchangée et que le générateur applique juste les transformations utiles pour le changement de domaine :

$G_{s\to r}(x_r) \approx x_r$ et $G_{r\to s}(x_s) \approx x_s$

On a alors :

$$L_{rec}(G_{s\to r}, G_{r\to s}) = \mathbb{E}_{x_s\sim p_s}[\parallel G_{r\to s}(x_s) - x_s \parallel_1] + \mathbb{E}_{x_r\sim p_r}[\parallel G_{s\to r}(x_r) - x_r \parallel_1] \tag{3.4}$$

Fonction coût globale du réseau de translation (T-Net)

Ainsi la fonction objective du réseau de translation s'écrit :

$$L_{T-Net} = L_{GAN}(G_{s\to r}, D_r) + L_{GAN}(G_{r\to s}, D_s) + \lambda_{cyc}L_{cyc}(G_{s\to r}, G_{r\to s}) + \lambda_{rec}L_{rec}(G_{s\to r}, G_{r\to s}) \tag{3.5}$$

avec $\lambda_{cyc}, \lambda_{rec}$ choisit empiriquement.

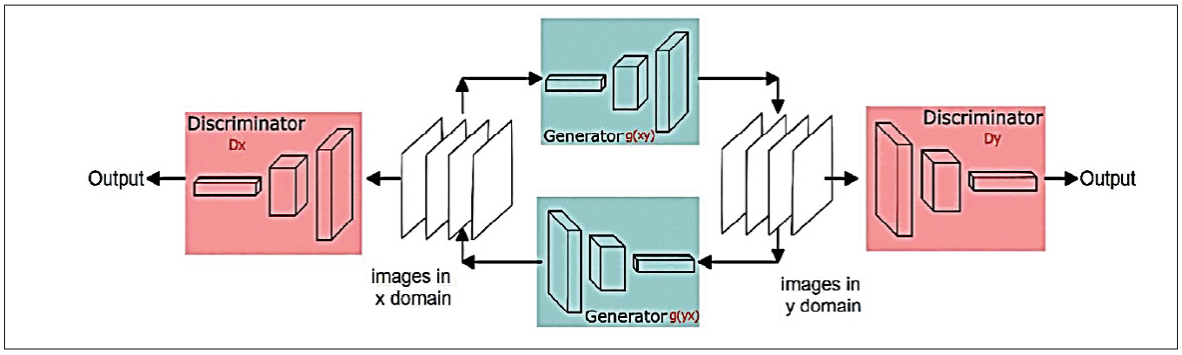

FIGURE 3.5 Méthode CycleGAN pour la translation d'images non-appariées
Tirée de Madadi *et al.* (2020)

3.3.2 Discriminateur des caractéristiques

L'alignement des domaines peut aussi être effectué au niveau des caractéristiques apprises par le réseau. Une représentation des caractéristiques est invariante au domaine si elle suit la même distribution pour des données entrée source et cible. Le but est donc d'entraîner le modèle à apprendre une représentation des caractéristiques invariante au domaine sources et cibles, afin qu'il puisse généraliser les informations de disparités apprises grâce aux caractéristiques issues des données sources labélisées, vers les données cibles (Wilson & Cook (2020)).

Fonction coût du discriminateur (D_f)

On va chercher à renforcer l'alignement des domaines à travers un discriminateur D_f des caractéristiques intermédiaires de f_T. Son rôle va être de confondre la représentation intermédiaire des images synthétiques avec celle des images réelles. Ainsi, cela va permettre au modèle de prédire des cartes de disparité dans les domaines source et cible à partir d'une représentation des caractéristiques indépendante du domaine d'origine.

Le réseau de disparité et le discriminateur de caractéristiques sont entraînés par le principe GAN, où f_T joue le rôle du générateur tout en étant entraîné à sa tache. Le discriminateur est placé à une couche intermédiaire L_i du réseau. Il doit prédire le domaine dont est issue la représentation intermédiaire qu'il reçoit en entrée, tandis que le réseau de disparité cherche à tromper le discriminateur tout en produisant des cartes de disparité :

$$\begin{aligned} L_{GAN_f}(f_T, D_f) = min_{f_T} max_{D_f} \mathbb{E}_{x_r \sim p_r}[\log D_f(f_T^{L_i}(x_r))] \\ + \mathbb{E}_{x_s \sim p_s}[\log(1 - D_f(f_T^{L_i}(x_s)))], \end{aligned} \tag{3.6}$$

où $f_T^{L_i}(x_r)$ est la carte de caractéristique de x_r à la couche intermédiaire L_i du modèle f_T, $f_T^{L_i}(x_s)$ est celle de x_s.

Le discriminateur de caractéristique est placé dans l'encodeur du modèle et après la couche de corrélation pour prendre en compte les informations de corrélation entre les images gauches et droites. Kundu *et al.* (2018) montre que les représentations des caractéristiques synthétique et réelle sont plus discernables au niveau des couches profondes. Ainsi, seulement la dernière couche de l'encodeur sera mise a jour par D_f, soit $i = conv6b$.

De plus, en suivant l'approche appliquée par les mêmes auteurs, une fonction coût de reconstruction des caractéristiques (*feature consistency*) est ajoutée sur le domaine cible pour régulariser le processus d'adaptation adverse. En effet, la fonction coût adverse peut amener le modèle à prédire des cartes de disparité incohérente par rapport aux images cible en entrée (problème de *mode collapse*), du fait de l'absence de labels. Ainsi, pour s'assurer que la structure spatiale et le contenu ne sont pas altérés par le processus d'adaptation, on ajoute la multi-couche C_r permettant

de reconstruire la carte de caractéristique $f_T^{L_{i-1}}(x_r)$ en entrée de L_i durant l'apprentissage adverse. La fonction coût de reconstruction se formule comme suit (3.7) :

$$L_{recf}(C_r) = \mathbb{E}_{x_r \sim p_r}[\| f_T^{L_{i-1}}(x_r) - C_r(f_T^{L_i}(x_r)) \|_1] \tag{3.7}$$

où C_r est la multi-couche de reconstruction, $f_T^{L_i}$ la fonction de l'encodeur à la couche adapté L_i, et $f_T^{L_{i-1}}$ la fonction de l'encodeur avant la couche L_i.

La figure 3.6 illustre le principe avec $M_r = f_T^{L_i}$, $L_r = f_T^{L_{i-1}}$ et $i = conv6b$.

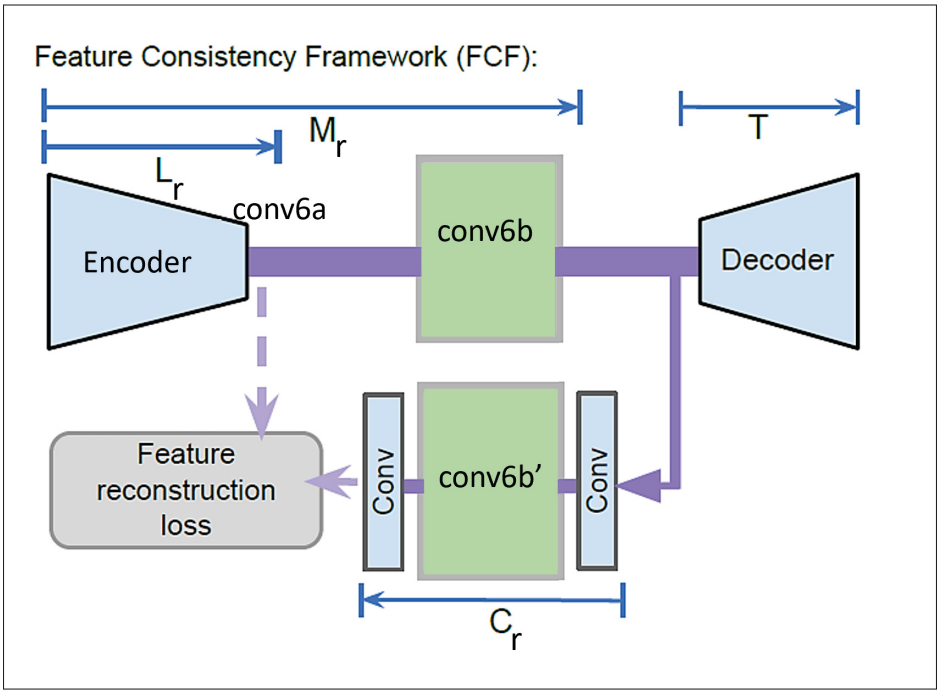

FIGURE 3.6 Régularisation du processus d'adaptation pour la préservation de la consistance des caractéristiques*
Adaptée de Kundu *et al.* (2018)

*Seul le bloc vert *conv6b* de M_r est mis à jour par adaptation.
La branche C_r va reconstruire conv6a à partir de conv6b par une fonction de reconstruction pour forcer le processus d'adaptation à garder les informations cohérentes issues des couches précédentes.

A noter, quand le discriminateur D_f est associé au réseau de translation, il discrimine entre les images synthétiques translatées et les images réelles.

3.3.3 Estimation des disparités

On cherche à entraîner le réseau de disparité f_T grâce aux images source, afin qu'il puisse performer dans le domaine réel tout en s'assurant que ses prédictions soient le moins affectées par le changement de domaine. Pour cela, on va d'abord fournir au réseau de disparité l'image synthétique x_s puis sa version translatée dans le domaine réel par le générateur $\tilde{x}_r$. La carte de disparité annotée correspondante est utilisée pour optimiser les paramètres du modèle. La fonction coût du réseau de disparité va mesurer la différence euclidienne L_1 (à l'échelle pixellique) entre la prédiction du réseau et la vérité terrain $\hat{y}_{D_s}$:

$$L_{disp}(f_T) = \| f_T(x) - \hat{y}_{D_s} \|_1 \tag{3.8}$$

où $x = \{x_s, \tilde{x}_r\}$

Les images réelles non translatées vont aussi être fournies au réseau de disparité afin de régulariser ses prédictions sur le domaine réel. Un terme de lissage locale est ajouté à la fonction coût du modèle pour corriger les discontinuités au niveau des frontières entre les objets dans la carte de disparité estimée $f_T(x_r)$ comme propose Zheng *et al.* (2018) :

$$L_{smooth}(f_T) = | \partial_x f_T(x_r) | e^{-|\partial_x x_r|} + | \partial_y f_T(x_r) | e^{-|\partial_y x_r|} \tag{3.9}$$

Nous allons aussi étudier l'ajout d'un discriminateur de caractéristiques au niveau de la prédiction, pour forcer le réseau de disparité à produire des cartes qui ne dépendent pas du domaine d'origine.

3.3.4 Fonction objective globale

En considérant toutes les fonctions précédentes, on obtient l'objectif global suivant :

$$L_{global} = L_{T-Net}(G_{s\to r}, D_r, G_{r\to s}, D_s) + \lambda_f L_{GAN_f}(f_T, D_f) + \lambda_{recf} L_{recf}(f_T, C_r) \\ +\lambda_d L_{disp}(f_T) + \lambda_s L_{smooth}(f_T), \tag{3.10}$$

Où L_{T-Net} représente la fonction coût entraînant un générateur à translater le plus fidèlement possible des images synthétiques dans le domaine réel.
A noter que L_{GAN_f} cherche à confondre les représentations intermédiaires de f_T tandis que L_{disp} et L_{smooth} entraînent le réseau à l'estimation des disparités. $\lambda_f, \lambda_{recf}, \lambda_d, \lambda_s$ sont choisis empiriquement.

3.4 Résultats des expériences

Dans cette section, nous allons présenter les architectures des modèles évalués, les détails de l'implémentation et les différentes expériences réalisées sur les bases de données SceneFlow et KITTI. Nous allons comparer qualitativement et quantitativement les résultats entres les différentes configurations de la méthode, mais aussi avec des méthodes de la littérature. On considère durant l'apprentissage que les images sources et cibles sont non-appariées.

3.4.1 Bases de données

Bases de données synthétiques
SCENEFLOW DATASET. En s'inspirant de l'approche appliquée par Mayer & Ilg (2016), FlyingThings3D est utilisée comme base source pour entraîner les modèles de translation et disparité.

vKITTI. vKITTI est employé comme base de données source additionnelle pour entraîner le réseau de translation.

Bases de données réelles

KITTI. KITTI2015 (200 images pour l'entraînement) et KITTI2012 (194 images pour l'entraînement) sont utilisées comme bases cibles durant l'apprentissage adverse. On suppose donc par la suite que leurs vérités terrain ne sont pas disponibles.

ETH3D et MB. L'ensemble d'apprentissage de chaque base (27 et 15 respectivement) est utilisé pour évaluer la généralisation du modèle dans le domaine cible.

Comme certaines cartes ont de grande valeur de disparité, on limite le maximum de disparité à 192 lors de l'apprentissage sur les bases synthétiques et à 160 pour affiner sur KITTI.

3.4.2 Architecture du réseau

L'implémentation du L_{T-Net} (CycleGAN) est celle proposée par Zhu *et al.* (2017) [1]. Le générateur est un ResNet inspiré de Johnson, Alahi & Fei-Fei (2016) et le discriminateur est un PatchGAN (Isola *et al.* (2017)) qui permet de discriminer efficacement si un patch d'une image est vraie ou faux. Le DispNetC (Mayer & Ilg (2016)) et sa version améliorée (développée à la partie précédente) sont utilisés comme réseau de disparité. Deux types de discriminateur de caractéristiques intermédiaire seront étudiés : un discriminateur MLP comme proposé par Zheng *et al.* (2018) [2], un discriminateur convolutionnel inspiré du PatchGAN.
Comme vu précédemment, les discriminateurs seront placés à la fin de l'encodeur soit à la position *conv6b*. On étudiera aussi leurs impacts au niveau des prédictions multi-résolutions.

3.4.3 Détails de l'implémentation

L'implémentation de la méthode et l'apprentissage du modèle sont faits avec *Pytorch*. ADAM (Kingma & Ba (2015)) est employé pour l'optimisation avec comme paramètre ($\beta_1 = 0,5$; $\beta_2 = 0,999$) pour le réseau de translation et ($\beta_1 = 0,95$; $\beta_2 = 0,999$) pour le réseau de disparité.

[1] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

[2] https://github.com/lyndonzheng/Synthetic2Realistic

Générateur et Discriminateur. Pour l'implémentation des équations adverse (équation 3.1, 3.2) le log negatif (BCE) est remplacé par une fonction coût moindre carrée (MSE) qui est plus stable durant l'apprentissage adverse (Zheng *et al.* (2018)). Durant l'optimisation du discriminateur d'image, on divise sa fonction objective par deux, permettant d'équilibrer son apprentissage avec le générateur.

Ainsi pour une translation d'un domaine i vers j, $G_{i\to j}$ va minimiser $\mathbb{E}_{x_i\sim p_i}[(D_j(G_{i\to j}(x_i))-1)^2]$ et D_j va minimiser $\frac{1}{2}[\mathbb{E}_{x_i\sim p_i}[D_j(G_{i\to j}(x_i))^2] + \mathbb{E}_{x_j\sim p_j}[(D_j(x_j)-1)^2]]$.

De plus, pour réduire l'oscillation du modèle de translation, on applique la stratégie de Shrivastava *et al.* (2017) qui utilise les images précédemment générées pour mettre à jour le discriminateur, plutôt que la dernière image produite. Pour cela on utilise un buffer qui enregistre les 10 dernières images générées. Le taux d'apprentissage est mis à 0,0002 avec un batch de 1 pour le L_{T-Net} durant les 10 premières époques, puis subit une décroissance linéaire pour les 10 prochaines époques. Les poids des différentes fonctions coût sont mis à $\lambda_{rec} = 5, \lambda_{cyc} = 10, \lambda_f = 1, \lambda_{recf} = 1$.

Réseau de disparité. Le taux d'apprentissage est fixé à 0,0001 avec un batch de quatre durant 20 époques, puis est divisé par deux toutes les 10 époques. On applique comme dans le chapitre précédent une méthode de planification des poids pour la fonction coût multi-résolution, ce qui permet d'éviter de mixer les gradients lors de l'optimisation de chaque résolution. Les poids des différentes parties de la fonction objective sont mis à $\lambda_d = 100, \lambda_s = 0,01$.

Stratégie d'apprentissage. On va comparer trois types d'apprentissage :

1. *{DispNetCv4 + D_f}*. La première consiste à pré-entraîner le réseau de disparité avec les images synthétiques (SceneFlow), puis de réaliser l'adaptation de domaine à l'aide du discriminateur de caractéristiques intermédiaire et des images réelles KITTI. Nous avons fait face à différents problèmes pour la réalisation de cette stratégie. En effet, la carte de caractéristique traitée par D_f possède des milliers de poids. Lorsqu'on la transmettait telle quelle au discriminateur et qu'on effectuait la mise à jour des poids, le discriminateur faisait diverger les paramètres du générateur qui devenaient des valeurs *NaN*. Pour contrer cela, nous avons réalisé plusieurs modifications : par l'ajout d'une couche de normalisation et *average pooling*, d'une fonction d'activation Sigmoid pour limiter la sortie entre $[0; 1]$,

d'une fonction de reconstruction des caractéristiques (équation 3.7) et par la mise à jour d'une seule couche par apprentissage adverse pour préserver le contenu des caractéristiques importantes à la tache ;

2. *{CycleGAN + DispNetCv4}*. La deuxième considère le CycleGAN comme réseau de translation du synthétique vers le réel. Le réseau est alors optimisé par L_{T-Net}.
   Initialement, nous avions comme objectif d'entrainer le réseau de translation sur le couple de base de données (SceneFlow ; KITTI). Cependant nous avons rencontré des problèmes durant l'apprentissage, conduisant à des résultats non satisfaisants (voir figure **??**). Le premier problème rencontré concernait l'écart de domaine entre les bases SceneFLow et KITTI, qui est très important. Le cycleGAN n'arrive pas à confondre les deux distributions efficacement, ce qui entraîne à des prédictions incorrectes. Afin de créer un pont entre les deux domaines et ainsi faciliter le changement de domaine, il a fallu rajouter la base synthétique vKITTI, représentant un écart de domaine plus petit (environnement similaire).
   Un autre problème était au niveau de la résolution en entrée du CycleGAN. Le réseau est conçu pour une résolution de 256×256, alors que le DispNetCv4 est conçu pour une résolution de 768×384. L'approche initiale était de réaliser un apprentissage *end-to-end* avec le réseau de disparité, en modifiant le CycleGAN pour traiter des images avec une résolution 768×384. Cependant les performances du réseau étaient mauvaises. Nous avons donc gardé la résolution 256×256 pour l'apprentissage du CycleGAN seul. Puis nous avons généré la fausse base de données "réelles" SceneFlow (voir figure 3.9 et 3.8) pour entraîner le DispNetCv4 (en 768×384) à partir de zéro, par les fonctions $\lambda_d L_{disp}$ et $\lambda_s L_{smooth}$.
   Finalement, nous avons rencontré le plus de difficulté dans la conception de la fonction coût et le choix des hyper-paramètres. Un mauvais choix de ces paramètres entraîne un apprentissage instable. Pour chaque discriminateur (D_r, D_s et D_f), il a fallu définir une stratégie d'apprentissage pour éviter une optimisation trop rapide par rapport au générateur, et permettre de bien initialiser le générateur avant d'effectuer l'apprentissage adverse (éviter valeurs *NaN*). Pour cela, nous avons déterminé une epoch seuil à partir de laquelle démarre l'apprentissage du discriminateur, un stride de 4 pour la mise à jour de ses poids, et une diminution du taux d'apprentissage plus lente que celle du générateur. Ensuite, nous avons

dû modifier la fonction coût adverse log negatif (BCE) par la fonction moindre carrée (MSE), qui a permis une meilleure optimisation du générateur. De plus, bien que le CycleGAN produit des images dans le domaine réel, on a remarqué que les pixels subissaient de la distorsion et des changements de couleur et position. Ces perturbations sur l'image translatée ont tendance à perturber la géométrie épipolaire de l'image stéréo, augmentant ainsi fortement l'erreur de disparité. Nous avons alors adapté la fonction coût cyclique par l'ajout d'un poids plus important ($\lambda_{cyc} = 10$) et de l'erreur de disparité issue de la différence entre la carte de disparité des images initiales et celle produite par leurs versions translatées. Cela a permis de forcer le réseau à générer une image translatée plus respectueuse des contraintes spatiales et de couleur de chaque pixel en entrée ;

3. *{CycleGAN + DispNetCv4 + D_f}*. Les images translatées par le CycleGAN vont servir à entraîner le réseau de disparité à partir de zéro, par les fonctions $\lambda_d L_{disp}$, $\lambda_s L_{smooth}$ et $\lambda_f L_{GAN_f}$. D_f discrimine alors les images sources translatées et images cibles.

Outils d'évaluation. Nous allons évaluer les performances des méthodes proposées sur les disparités par la mesure D1 (erreur sur les mauvais pixels). Plus précisément, nous allons employer la mesure D1-all calculant l'erreur sur tout les pixels. Plus cette mesure est basse, meilleurs sont les résultats.

Configurations. Nous allons comparer différentes configurations du même modèle en modifiant les données en entrée, le type d'apprentissage, avec ou sans un discriminateur de caractéristiques... Les réseaux de disparité employés seront le DispNetC et sa version améliorée développée au chapitre précédent, le DispNetCv4. On évaluera comme réseau de translation le CycleGAN de fonction coût L_{T-Net} (3.5). Deux types de D_f seront étudiés au niveau de la *conv6b* : un MLP (D_f^{MLP}) et un CNN (D_f^{CNN}). Un discriminateur CNN sera aussi étudié pour les prédictions multi-résolution ($D_f^{Pred_i}$ pour la prédiction i).

Augmentation de données. On réalise une normalisation des couleurs sur l'ensemble des données source et cible. Durant l'apprentissage, les images sont aléatoirement recadrées aux dimensions 768×384, et subissent des modifications aléatoires de couleurs, contraste et

de luminosité par la fonction ColorJitter de Pytorch avec comme paramètres ($brightness = [0, 1]; contrast = [0, 1]; saturation = 0; hue = 0, 4$).

FIGURE 3.7 Comparaison sur vKITTI des performances entre la configuration initiale du CycleGAN et la version proposée*

*La première colonne correspond aux prédictions et la seconde aux *ground-truth* (GT). La première ligne est la paire issue du CycleGAN initiale et la seconde est celle issue de sa version améliorée.

3.4.4 Résultats

On présente dans cette partie les différents résultats des méthodes entraînées par adaptation de domaine, ainsi que ceux des méthodes supervisées et des méthodes seulement entraînées sur le domaine synthétique. Il y est aussi regroupé les résultats obtenus par des méthodes supervisées de la littérature (GCNet de Kendall *et al.* (2017), PSMNet de Chang & Chen (2018), MADNet de Tonioni *et al.* (2019)) dont les données sont issues de Cai, Poggi, Mattoccia & Mordohai (2020)).

Tableau 3.1 Résultats de l'estimation des disparités par adaptation sur KITTI. L'erreur D1-all est indiquée dans le tableau

Méthode	**Supervision**[a]	**KITTI 2015**	**KITTI 2012**
PSMNet	NS	26,62	27,02
MADNet	NS	43,98	39,17
DispNetC	NS	16,13	12,91
DispNetCv4	NS	11,17	8,56
GCNet	NS	14,68	**6,22**
DispNetCv4 + D_f^{MLP}	NS+A	8,68	8,14
DispNetCv4 + D_f^{CNN}	NS+A	8,74	6,85
DispNetCv4 + $D_f^{Pred_4}$	NS+A	8,44	7,79
DispNetCv4 + $D_f^{Pred_4}$ w/o L_{recf}	NS+A	9,11	7,81
CycleGAN + DispNetCv4	NS+A	8,96	7,27
CycleGAN + DispNetCv4 + $D_f^{Pred_4}$	NS+A	**8,10**	6,39
DispNetC	S	4,41	4,04
DispNetCv4	S	2,92	2,09
GCNet	S	2,23	1,37
PSMNet	S	**1,83**	**1,11**

[a]On compare les modèles supervisés (S) et non-supervisés (NS) avec ceux adaptés au domaine cible (NS+A)

Résultats sur KITTI. Les résultats sur les bases cibles KITTI sont regroupés dans le tableau 3.1 et les résultats qualitatifs des améliorations sont présentés à la figure 3.10. On peut voir ainsi l'apport des différentes techniques d'adaptation de domaine sur les performances du modèle de disparité. Le discriminateur intermédiaire de caractéristique et le réseau de translation permettent d'atteindre de meilleurs résultats dans le domaine réel non annoté, ce qui valide l'intérêt de tels

méthodes pour adapter un modèle entre deux domaines très différents. Plus particulièrement, en entraînant le réseau de disparité DispNetCv4 avec des données sources translatées par un CycleGAN, et en ajoutant un discriminateur intermédiaire au niveau de la 4ème prédiction, l'erreur D1 sur KITTI2015 est diminuée d'environ 27%, et de 25% sur KITTI2012, par rapport au DispNetCv4 seulement entraîné sur le domaine synthétique.

Parmi les méthodes non-supervisées sur le domaine cible, le $CycleGAN+DispNetCv4+D_f^{Pred_4}$ obtient l'erreur D1 la plus faible sur KITTI2015 mais le GCNet est légèrement meilleur sur KITTI2012 avec D1 = $6,22\%$. Cela s'explique premièrement par la différence de complexité entre les deux versions de KITTI : on observe que les méthodes ont toujours une bien meilleure précision sur KITTI 2012, car les annotations sont moins denses et moins rigoureuses. Pour KITTI 2012, les *ground-truth* sont produites à partir de l'accumulation des points de 10 frames (5 avant et 5 après de la frame référence), et les régions ambiguës sont retirées à la mains. Tandis que pour KITTI 2015, il est appliqué la même méthode, mais les *ground-truth* sont en plus complétées à l'aide de modèles CAD des objets dynamiques (voiture, vélo, ...), d'une annotation plus détaillée de l'arrière plan, et d'un algorithme de *stereo-matching* pour assister la correction des valeurs ambiguës de profondeur. Ainsi, le GCNet arrive mieux à s'adapter sur la KITTI 2012 qui demande des prédictions moins détaillées que la KITTI 2015.

En second, le GCNet contient moins de paramètres que le PSMNet et le DispNetCv4 (2,6M versus 5,2M et 80M respectivement), ce qui le rend moins vulnérable au sur-apprentissage sur le style du domaine source (couleur, texture...) et lui permet une meilleure généralisation (Cai *et al.* (2020)). C'est aussi la raison pour laquelle les performances de généralisation du PSMNet sont très faible sur KITTI ($D1 = 26,62\%$ sur KITTI 2015 $D1 = 27,02\%$ sur KITTI 2012) alors qu'il possède les meilleurs résultats lorsqu'il est supervisé.

Trois types de discriminateur intermédiaire ont été étudié : le discriminateur $D_f^{Pred_4}$ est celui assurant les meilleures performances sur KITTI2015 avec D1=8,44%. En effet, après avoir étudié le discriminateur sur les sept prédictions du réseau de disparité (voir tableau 3.2), la 4ème prédiction est celle permettant une meilleure adaptation du modèle sur le domaine cible. Par ailleurs, le discriminateur D_f^{CNN} atteint le meilleur résultat sur KITTI2012 avec D1=6,85%.

Le réseau de translation permet d'entraîner le réseau de disparité à sa tache, directement sur le domaine cible avec des annotations. Quand la translation est la plus fidèle possible, le modèle peut atteindre des performances similaires à une configuration supervisée. Dans notre cas, le réseau de translation CycleGAN permet d'améliorer la précision du réseau de disparité par rapport au DispNetCv4 seulement entraîné sur le domaine synthétique (diminution de 20% de l'erreur D1 sur KITTI2015 et 15% sur KITTI2012), mais les résultats restent éloignés du cas supervisé. Cela est du entre autres au fait que, lors de la translation, certain pixels subissent de la distorsion et des changements de couleur et position. Ces modifications dans l'image translatée ne prennent pas en compte les correspondances entre l'image gauche et droite. Ainsi, pour ces pixels, le processus de *stereo-matching* devient incorrect du fait que la contrainte épipolaire n'est plus respectée. Les résultats de translation sont présentés sur les figures 3.8 et 3.9.

Tableau 3.2 Estimation des disparités non-supervisée pour le discriminateur de caractéristiques à différentes positions de prédictions[a]

Méthode	KITTI 2015	KITTI 2012
DispNetCv4 + $D_f^{Pred_1}$	8,64	8,16
DispNetCv4 + $D_f^{Pred_3}$	8,51	8,01
DispNetCv4 + $D_f^{Pred_4}$	**8,44**	**7,79**
DispNetCv4 + $D_f^{Pred_5}$	9,30	8,26
DispNetCv4 + $D_f^{Pred_6}$	9,63	8,88

[a]L'erreur D1-all est indiquée dans le tableau.

Résultats sur ETH3D et MB. Dans le tableau 3.3, on y présente les résultats sur les bases tests ETH3D et MB pour les méthodes non supervisées entrainées avec et sans adaptation sur KITTI. On cherche ici à évaluer les performances sur des bases réelles jamais observées et différentes de la base cible. Les méthodes développées possèdent de très bons résultats sur ETH3D : le $DispNetCv4 + D_f^{Pred_4}$ est le modèle atteignant les meilleurs performances avec D1=4,21%. L'amélioration $DispNetCv4$ obtient aussi une faible erreur de disparité, ce qui montre ses très bonnes capacités de généralisation par rapport au DispNetC de base.

Cependant, les résultats des méthodes proposées sont moins bons sur MIDDLEBURY. Les méthodes de disparité 3D de la littérature (GCNet, PSMNet) obtiennent une plus faible erreur. Cela est du, entre-autres, à la grande dimension des images de MB (1920×1080) par rapport à l'entrée du DispNetC (768×384) et aux autres bases de données (ETH3D = 752×480, KITTI = 1240×376). Les régions observées sur les images de MB auront moins de détailles et certaines zones ne seront pas traitées, ce qui affecte la qualité de l'encodage des informations par le réseau. Les résultats qualitatifs sont présentés à la figure 3.11.

TABLEAU 3.3 Résultats sur ETH3D et MB pour l'estimation des disparités par adaptation[a]

Méthode	**Supervision**	**ETH3D**	**MB**
GCNet	NS	8.03	30.42
PSMNet	NS	18,91	**26,92**
DispNetC	NS	11,14	45,60
DispNetCv4	NS	4,43	35,51
DispNetCv4 + D_f^{MLP}	NS	5,50	31,58
DispNetCv4 + D_f^{CNN}	NS	5,32	34,13
DispNetCv4 + $D_f^{Pred_4}$	NS	**4,21**	32,29
CycleGAN + DispNetCv4	NS	4,57	34,76
CycleGAN + DispNetCv4 + $D_f^{Pred_4}$	NS	4,23	32,47

[a]Chaque modèle est entraîné en considérant KITTI comme base cible. Les erreurs D1-all sont indiquées dans le tableau.

FIGURE 3.8 Exemple d'images réelles générées par le réseau de translation CycleGAN sur vKITTI. La ligne du haut correspond aux images synthétiques, la ligne du bas leurs versions translatées

FIGURE 3.9 Exemple d'images réelles générées par le réseau de translation CycleGAN sur SceneFlow. La ligne du haut correspond aux images synthétiques, la ligne du bas leurs versions translatées

3.5 Conclusion

Dans cette partie, nous avons cherché à répondre au problème de changement de domaine non-supervisé pour la tache d'estimation des disparités par stéréo. Nous avons proposé une méthode associant l'apprentissage adverse et la translation d'image vers image non apparié, permettant d'adapter un réseau de disparité à partir d'images synthétiques annotées vers le domaine réel sans annotation. Les expérimentations sur différentes bases de données ont démontré l'efficacité de l'approche d'adaptation à réduire l'écart entre des domaines synthétique et réel très différents, et la bonne capacité de généralisation à des domaines non-vues. La méthode proposée s'intègre facilement au réseau de disparité : elle emploie le CycleGAN pour générer une base d'images stéréo synthétiques translatées vers le réel, et un discriminateur associé à une méthode de régularisation du contenu va s'assurer que les représentations internes au réseau de disparité soit invariante au domaine.

Bien que les résultats de la méthode montrent un gain en précision par rapport à l'approche de base, des erreurs persistes. Il est alors intéressant d'explorer d'autres approches dans le futur. Le CycleGAN ne permet pas encore de translater fidèlement des images stéréo, du fait qu'il ne prend pas en compte la contrainte de disparité. Pour cela, il faut chercher à intégrer une fonction coût s'assurant que les correspondances entre les images gauche et droite restent inchangées, par une méthode de reconstruction comme proposé par Godard *et al.* (2017). Mais aussi, si les ressources nécessaires sont disponibles, en effectuant un apprentissage *end-to-end* du réseau de translation et de disparité, ce qui permettrait de rajouter la contrainte liée à la tache dans le processus de translation d'image.

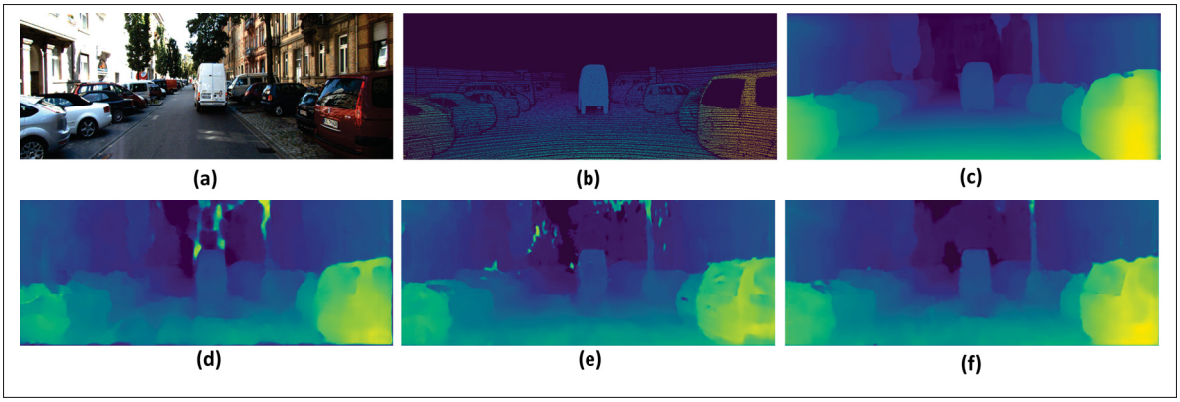

FIGURE 3.10 Résultats qualitatifs des méthodes de disparité non-supervisées sur KITTI2015. (a) Image gauche, (b) *ground-truth*, (c) carte de disparité par DispNetCv4 supervisé sur KITTI, (d)-(f) cartes de disparité par $DispNetC$, DispNetCv4 et $CycleGAN + DispNetCv4 + D_f^{Pred_4}$ non-supervisés sur KITTI

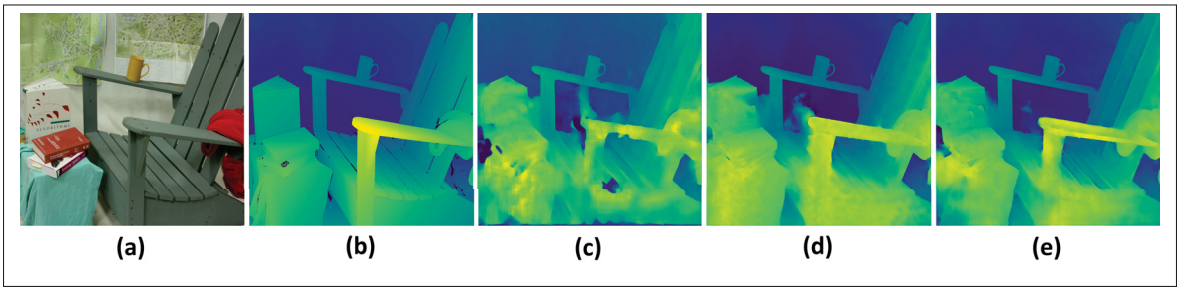

FIGURE 3.11 Résultats qualitatifs des méthodes de disparité non-supervisées sur MB. (a) Image gauche, (b) *ground-truth*, (c)-(e) cartes de disparité par $DispNetC$, DispNetCv4 et $CycleGAN + DispNetCv4 + D_f^{Pred_4}$ non-supervisés sur KITTI

CHAPITRE 4

DÉTECTION D'OBJET 3D À PARTIR DES DISPARITÉS

4.1 Introduction

Mise en contexte. L'objectif principal est de proposer une méthode basée sur des images stéréo pour de la détection 3D d'objets, en condition réelle. Notre cas d'étude nous a poussé à choisir une approche associant un réseau de disparité à partir d'images stéréo, avec une méthode réalisant de la détection 3D à l'aide des cartes de disparité produites. Dans les parties précédentes, nous avons introduit un réseau de disparité, le DispNetC, dont on a amélioré par différentes approches, sa précision, robustesse et généralisation afin qu'il puisse performer aussi bien lors de son apprentissage sur des données synthétiques labélisées, que lors d'une application réelle sans supervision.
A travers ce chapitre, nous allons évaluer l'efficacité de l'amélioration DispNetCv4 à représenter l'information 3D pour la détection, dans le cas d'un apprentissage supervisé mais aussi non-supervisé en employant la méthode d'adaptation de domaine vu dans le chapitre précédent. Les expériences sont réalisées sur la base de donnée réelle KITTI.

Importance de la représentation 3D. Nous avons montré dans le premier chapitre qu'il existe un écart important entre les méthodes de détection 3D basées sur des données lidar et celles basées sur des images stéréo. La façon d'obtenir l'information 3D étant physiquement différente, on observe des précisions très éloignées entre les deux capteurs : par exemple, une image stéréo permet de produire une carte de profondeur dont l'erreur augmente quadratiquement avec la distance. Tandis que pour les capteurs basés sur la propagation d'une onde, l'erreur de profondeur dans le nuage augmentera approximativement linéairement avec la distance (Wang *et al.* (2019)). Pour minimiser l'erreur due à la représentation de la donnée 3D à partir d'images, nous allons employer l'approche de détection du Pseudo-Lidar de Wang *et al.* (2019). Ce réseau transforme la carte de disparité 2D en un nuage de points 3D similaire à ceux produits par lidar, puis le transmet au réseau réalisant la détection 3D. En considérant un nuage de points 3D plutôt qu'une

carte de profondeur 2D, on évite les erreurs de détection dues aux discontinuités au niveau des frontières d'objets ayant des profondeurs très éloignées. Une convolution 2D ou 3D applique un patch local entre des pixels voisins d'une image, pouvant avoir des valeurs de profondeur très éloignées, ce qui n'est pas le cas lorsqu'on traite un nuage de points 3D où les pixels voisins sont réellement proches (Wang *et al.* (2019)). Il en est de même pour le traitement des différentes apparences d'un objet : une carte de profondeur ne permet pas au convolutions d'observer un objet de la même manière lorsqu'il est loin (dans son entièreté) que lorsqu'il est proche (juste une partie). Alors que le regroupement des points dans un nuage 3D permet de définir un objet quelque soit sa position dans l'espace (Wang *et al.* (2019)).

4.2 Méthode

Dans cette partie, nous allons présenter la méthode de détection 3D par images stéréo. Plus précisément, nous allons employer une méthode de détection 3D basée sur un réseau de disparité, qui est une méthode en deux étapes : elle associe un réseau de disparité basé sur des images stéréo avec un détecteur 3D par lidar. Cette approche est utilisée au sein de plusieurs méthodes de détection 3D par image stéréo de la littérature comme Pseudo-Lidar (Wang *et al.* (2019)), Pseudo-Lidar++ (You *et al.* (2020)) ou DispRCNN (Sun *et al.* (2020)).

L'amélioration DispNetCv4 sera utilisée pour l'estimation des cartes de disparité. Les pixels de disparité sont transformés en profondeur puis projetés dans la scène 3D afin de construire le nuage de points "pseudo-lidar" (voir Wang *et al.* (2019)). Cette nouvelle représentation permet d'employer un détecteur 3D par lidar. Pour notre étude, nous allons étudier le réseau AVOD proposé par Ku *et al.* (2018) qui est un détecteur 3D multimodale en deux étapes. Il reçoit en entrée l'image référence de la scène et le nuage de points en vue d'oiseau (carte BEV), et utilise ces données pour générer des propositions 3D puis prédire les boites et classes des objets détectés.

Dans un premier temps, nous allons présenter le réseau de disparité et la méthode de conversion de la carte de profondeur en nuage de points 3D, puis nous allons introduire le réseau AVOD pour l'estimation des boites 3D. Le principe général de la méthode est présenté à la figure 4.1.

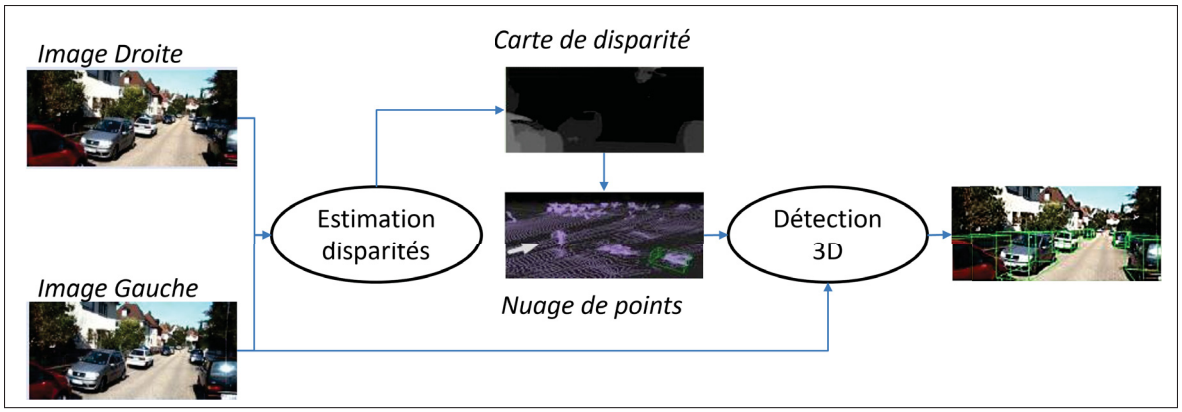

FIGURE 4.1 Méthode de détection 3D à partir des disparités

4.2.1 Réseau de disparité

Estimation de la profondeur. Le réseau de disparité reçoit en entrée une pair d'image gauche et droite $(ImgL, ImgR)$ issue d'un système de caméra stéréoscopique ayant une baseline B (écart entre les caméras), une distance focale f et des centres optiques (C_1, C_2) pour la caméra gauche et droite respectivement. Il prédit en sortie la carte de disparité y_D de même dimension en considérant l'image gauche comme référence. La carte de profondeur Z est ensuite obtenue suivant la relation 1.1 présentée précédemment. Soit $R_I = (O, \vec{u}, \vec{v})$ le repère de l'image visualisée. On considère ici que X est un pixel, soit $X = (u, v)$ où u est sa composante horizontale et v sa composante verticale.

Génération du nuage de points. Connaissant les paramètres de la caméra de référence $f = (u_f, v_f)$, $C_1 = (u_{c1}, v_{c1})$ et la carte de profondeur de la scène observée Z, on peut projeter dans le repère de la caméra $R_c = (C_1, \vec{x_c}, \vec{y_c}, \vec{z_c})$ l'ensemble des pixels $X(u, v)$ de la carte. Leurs positions 3D (x, y, z) sont obtenues par les formules suivantes (Wang *et al.* (2019)) :

$$\begin{aligned} z &= Z((u, v)) \\ x &= \frac{(u - u_{c1}) \times z}{u_f} \\ y &= \frac{(v - v_{c1}) \times z}{v_f} \end{aligned} \tag{4.1}$$

On obtient alors le nuage de points 3D (appelé *pseudo-lidar*) $\{x_n, y_n, z_n\}_{n=1}^{N}$ où N est le nombre total de pixels.
Suivant l'approche de Wang *et al.* (2019), plusieurs étapes de post-traitement (suppression de points, redimensionnement, alignement, mesure de réflectivité...) sont appliquées au nuage *pseudo-lidar* pour conserver seulement les points contenus dans le champs de vision de la caméra référence, et rendre le nuage compatible à la méthode de détection 3D conçue pour de vrais nuages de points lidar. Il observe qu'un nuage de points issu de la carte de disparité est plus dense que celui produit par lidar. En effet, les points obtenus par lidar sont distribués selon quelques lignes horizontales (64 ou 128).

4.2.2 Réseau de Détection 3D

Après avoir traité le nuage *pseudo-lidar*, il est possible d'appliquer n'importe quel réseau de détection 3D par lidar. Comme nous avons vu précédemment que les méthodes fusion permettent d'atteindre les meilleures performances, nous allons ainsi employer le détecteur AVOD de Ku *et al.* (2018) pour la prédiction 3D. L'architecture du réseau est présentée à la figure 4.2.
AVOD est un détecteur 3D multimodale, c'est à dire qu'il reçoit en entrée deux signaux : l'image référence et la carte BEV issue du nuage de points 3D pour produire deux cartes de caractéristiques de haute résolution. Plus précisément, on fournit en parallèle au réseau une image de la scène pour en extraire les informations de texture et couleur RGB, et une carte BEV pour en extraire les informations 3D. La carte BEV est générée à partir d'une représentation voxellique du nuage *pseudo-lidar* : la largeur W et la profondeur P deviennent les dimensions spatiales tandis que la hauteur est encodée dans les channels C (Ku *et al.* (2018)). Le nouveau repère devient donc $R_{BEV} = (O, \vec{x_{BEV}}, \vec{z_{BEV}}, \vec{y_{BEV}})$, et la carte est de dimension $(W \times P \times C)$.
Bien que les pixels de différentes hauteurs soient regroupés dans une position du plan, cela ne diminue pas la qualité de la représentation 3D. En effet, on fait l'hypothèse que les pixels à une même position (x_{BEV}, z_{BEV}) mais ayant un y_{BEV} différent, appartiennent au même objet.
En plus d'être multimodale, AVOD est un détecteur en deux étapes comme le R-CNN de Girshick *et al.* (2014) : un RPN exploite des boites 3D prédéfinies (*3D anchor*), générées en chaque

position de la carte BEV et projetées sur les deux vues, pour extraire les RoIs des cartes de caractéristiques image et BEV. Les RoIs sont alors fusionnées pour produire des propositions de boites plus précises et un score *objectness* (comme vu dans Qin *et al.* (2019)). Finalement, l'étape de détection est effectuée à partir de la fusion de chaque couple de RoIs issues de la projection des nouvelles propositions 3D sur les cartes de caractéristiques des deux vues en entrée. Le détecteur prédit la boite 3D, l'orientation et la classe de chaque proposition d'objet.

4.3 Résultats des expériences

Dans cette section, nous allons évaluer la précision de la méthode sur la base de donnée KITTI, mais ainsi que d'autres paramètres, pour différentes configurations du réseau de disparité. Les résultats de la méthode seront comparés avec ceux de la littérature.

4.3.1 Base de données

La base de donnée KITTI pour la détection d'objet (Geiger *et al.* (2012)) est utilisée pour évaluer notre approche. La fraction d'apprentissage contient 3712 images stéréo et celle de validation (utilisée pour évaluer les modèles) en contient 3769. KITTI fournit le nuage de points lidar, les matrices de calibration des caméras et les labels des boites 3D. Seulement la catégorie "voiture" est considérée pour l'entraînement et l'évaluation.
La base de donnée KITTI pour les disparités (Geiger *et al.* (2012)) est employée pour la validation du modèle de disparité.

4.3.2 Détails de l'implémentation

L'implémentation de la méthode et l'apprentissage des modèles sont faits avec *Pytorch*. ADAM (Kingma & Ba (2015)) est employé pour l'optimisation avec comme paramètres ($\beta_1 = 0,95$; $\beta_2 = 0,999$) pour le réseau de disparité et ($\beta_1 = 0,5$; $\beta_2 = 0,999$) pour le réseau de translation.

Les étapes présentées par la suite, pour l'apprentissage des méthodes, sont inspirées de Wang *et al.* (2019) [1].

Réseau de disparité. Le réseau de disparité employé est le DispNetCv4. En nous inspirant des travaux de Wang *et al.* (2019), le réseau est entraîné de zéro sur la base de donnée KITTI pour la détection d'objet et non pour les disparités. En effet, les auteurs ont observé que les données de la base KITTI pour les disparités sont présentes dans le set de validation de la base KITTI pour la détection 3D. Ainsi, les vérités terrain *pseudo-disparité* sont obtenues en projetant les données lidar dans le plan image 2D.
Les matrices de calibration du système de caméras sont utilisées pour obtenir le nuage de points à partir de la carte de profondeur.

Réseau de détection 3D. L'implémentation du détecteur 3D et le choix des hyper-paramètres sont ceux proposés par Ku *et al.* (2018) pour le AVOD-FPN. Il est entraîné de zéro sur la base d'apprentissage KITTI dont les données lidar sont remplacées par les versions *pseudo-lidar* générées par le réseau de disparité. Plus particulièrement, Les poids utilisés pour le réseau de détection sont directement issus de Wang *et al.* (2019).

Outils d'évaluation. Les méthodes étant évaluées et comparées sur la classe "voiture" de KITTI, nous allons seulement mesurer la qualité de la boite 3D par la précision AP_{3D} en considérant un seuil IoU de 0,7 et 0,5.

Configurations. Nous allons évaluer les performances du DispNetCv4 lors d'un apprentissage supervisé, non-supervisé (NS) et non-supervisé par adaptation de domaine (NS+A). La méthode d'adaptation employée est celle sélectionnée au chapitre précédent.
La méthode sera comparée au Pseudo-Lidar qui emploie le PSMNet (Chang & Chen (2018)) pour les disparités et AVOD pour la détection 3D en stéréo. Les autres résultats présentés sont issus de la littérature (voir Wang *et al.* (2019) et You *et al.* (2020)).

[1] code disponible à https://github.com/mileyan/pseudo_lidar

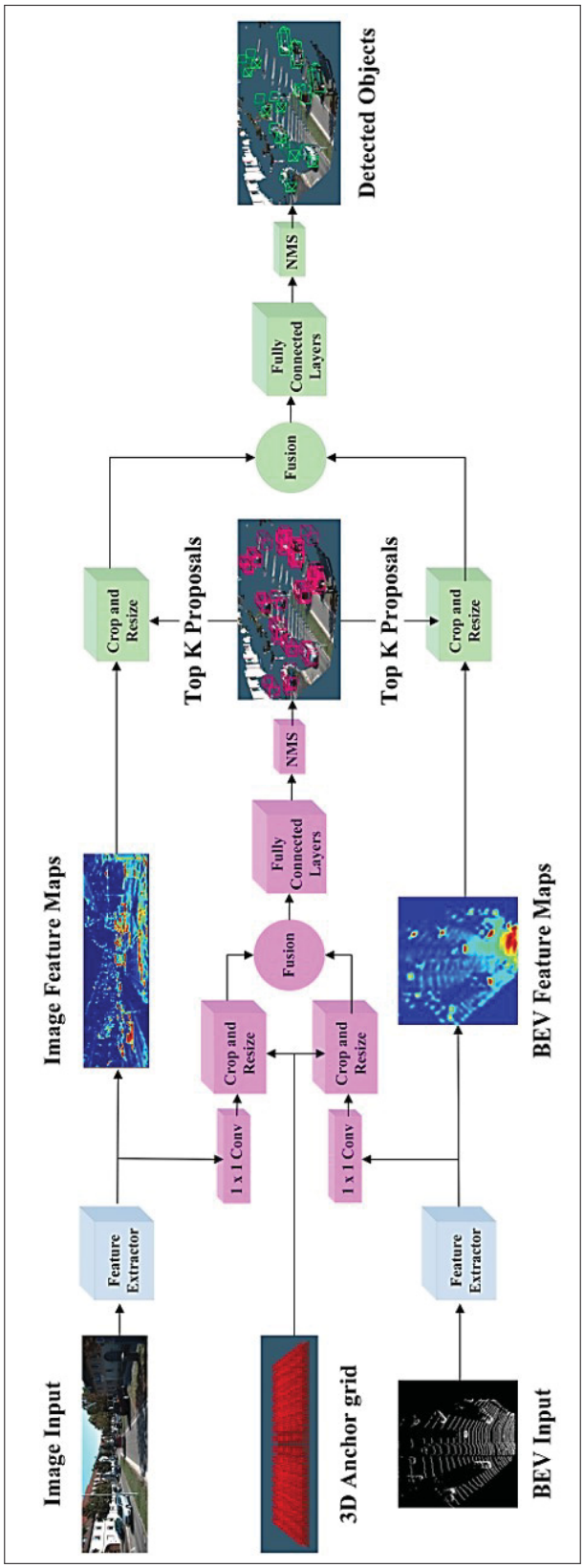

FIGURE 4.2 Illustration du AVOD. En bleu l'extracteur des caractéristiques, en rose le RPN, et en vert la partie détecteur. Le réseau reçoit en entrée la carte BEV, une image et des boites 3D prédéfinies en chaque position de la BEV (*3D Anchor grid*)
Tirée de Ku *et al.* (2018)

4.3.3 Résultats

Les résultats sur le set de validation KITTI sont regroupés dans les tableaux 4.1 pour l'apprentissage supervisé des disparités et 4.2 pour l'apprentissage non-supervisé. La méthode de détection proposé est DispNetCv4+AVOD. Elle est comparée avec sa version de base DispNetC+AVOD et avec des méthodes de la littérature employant divers signal en entrée.
En supplément, le temps de prédiction et l'empreinte mémoire lors de la prédiction sont aussi évalués par les tableaux 4.3 et 4.4 respectivement.

Détection par l'estimation supervisée des disparités. Les résultats du tableau 4.1 montrent que l'amélioration DispNetCv4 permet un gain considérable en précision par rapport à la méthode de base. Ainsi, on observe une augmentation d'environ 45% en précision pour IoU = 0,7 (modéré) par rapport au DispNetC+AVOD. La comparaison est effectuée selon la mesure $\%AP_{3D}$ (IoU = 0,7) car elle est employée pour classer les meilleures méthodes sur KITTI. Ainsi, les résultats indiquent qu'en diminuant l'erreur de disparité, on arrive à produire une donnée 3D de haute qualité, permettant au détecteur une meilleure compréhension de la scène et ainsi une meilleure détection.
De plus, la précision obtenue dépasse légèrement celle du réseau état de l'art Pseudo-Lidar pour IoU = 0,7 (modéré) (45,6 contre 45,3) ce qui est encourageant pour la suite. Cependant pour IoU = 0,7 (difficile), l'erreur reste inférieur à celle du Pseudo-Lidar. Bien que le PSMNet utilisé dans le Pseudo-Lidar soit plus précis que le DispNetCv4, cela n'implique pas forcément qu'il soit plus précis pour n'importe quelle valeur de profondeur estimée. Ici on peut voir que les convolutions du PSMNet lui permettent de mieux représenter les objets difficiles à voir (petit et loitain).
Des exemples de détections sont présentés à la figure 4.3.

Détection par l'estimation non-supervisée des disparités. Les résultats présentés dans le tableau 4.2 mettent en avant l'importance du principe d'adaptation lorsque les données cibles ne sont pas disponibles. Tout d'abord, grâce à notre méthode d'adaptation du réseau de disparité combinant un CycleGAN pour générer la "fausse" base d'apprentissage réelle, et un discriminateur de caractéristiques pour confondre les représentations internes du réseau,

on observe une augmentation d'environ 20% en précision pour le IoU = 0,7 (modéré) de notre approche de détection par rapport à un apprentissage sans adaptation. Ainsi, le principe d'adaptation permet non seulement de diminuer l'erreur de disparité, mais aussi de produire une représentation de la donnée 3D la plus fidèle possible, ce qui assure une meilleure détection. De plus, elle surpasse la méthode Pseudo-Lidar lors qu'elle est entraînée sans supervision, ce qui prouve ses capacités de généralisation grâce aux améliorations, mais aussi que la représentation de la donnée 3D garde une certaine qualité malgré que l'apprentissage soit non-supervisé. Cela montre aussi que les méthodes de disparité 3D, comme le PSMNet utilisé dans le Pseudo-Lidar, vont être très performantes lorsqu'on les entraine d'une manière supervisée mais seront moins adaptées à un apprentissage non-supervisé.

TABLEAU 4.1 Précision $\%AP_{3D}$ de la détection 3D supervisé sur KITTI. Les résultats sont pour les objets de la classe "voiture" avec la difficulté modérée (Mod) et difficile (Diff)

Méthode	**Signal Entrée**	$\%AP_{3D}$ **(IoU = 0,5)**		$\%AP_{3D}$ **(IoU = 0,7)**	
		Mod	Diff	Mod	Diff
Mono3D	Mono[a]	18,2	15,5	2,3	2,3
Pseudo-Lidar	Mono	42,8	36,3	17,2	16,2
3DOP	Stereo	34,6	30,1	5,1	4,1
DispNetC + AVOD	Stereo	60,2	53,3	31,5	29,0
Stereo-RCNN	Stereo	66,3	57,2	36,7	31,1
Pseudo-Lidar	Stereo	76,4	61,2	45,3	39,0
DispNetCv4 + AVOD	Stereo	77,5	62,0	45,6	36,1
Disp R-CNN	Stereo	**79,8**	69,8	47,2	39,7
ZoomNet	Stereo	**79,8**	70,5	**50,5**	43,6
Pseudo-Lidar++	Stereo	78,6	**75,1**	50,1	**45,3**
Pseudo-Lidar++	Lidar+Stereo	86,9	84,2	63,8	57,4
AVOD	Lidar+Mono	89,2	88,2	73,5	67,1

[a]Monoculaire

Mémoire et Temps de prédiction. On peut voir grâce au tableau 4.3 que le DispNetCv4+AVOD surpasse une grande partie des approches de la littérature en vitesse : il prédit les boites 3D en un temps record tout en assurant une bonne précision. Les méthodes stéréo plus rapide sont le RT3DStereo et le DispNetC+AVOD qui atteignent des précisions très inférieurs. Mais la méthode assure aussi une empreinte mémoire inférieur à 4Go comme le montre le tableau 4.4,

ce qui est inférieur à la mémoire consommée par le Pseudo-Lidar et permet de répondre au cahier des charges.

TABLEAU 4.2 Précision $\%AP_{3D}$ de la détection 3D par l'estimation des disparités non-supervisé sur KITTI. Les résultats sont pour les objets de la classe "voiture" avec la difficulté modérée (Mod) et difficile (Diff)

Méthode	**Supervision**[a]	$\%AP_{3D}$ **(IoU = 0,5)**		$\%AP_{3D}$ **(IoU = 0,7)**	
		Mod	Diff	Mod	Diff
DispNetC + AVOD	S	60,2	53,3	31,5	29,0
DispNetCv4 + AVOD	S	77,5	62,0	45,6	36,1
DispNetC + AVOD	NS	37,2	25,6	14,3	9,1
Pseudo-Lidar	NS	48,2	32,5	20,4	16,3
DispNetCv3 + AVOD	NS	47,3	33,0	19,7	15,4
DispNetCv4 + AVOD	NS	51,4	36,5	23,4	18,8
DispNetCv3 + AVOD	NS+A	55,8	43,5	23,8	19,9
DispNetCv4 + AVOD	NS+A	57,9	49,9	28,0	24,2

[a] On compare les modèles supervisés (S) et non-supervisés (NS) avec ceux adaptés au domaine cible (NS+A)

TABLEAU 4.3 Comparaison de la vitesse de détection avec les méthodes de la littérature

Méthode	**Temps de prédiction (f/s)**
3DOP	0,8
DSGN	1,4
Pseudo-Lidar++	< 2,0
Pseudo-Lidar	2,0
Stereo-RCNN	2,4
3DOEM	5,8
DispNetCv4 + AVOD	7,2
DispNetC + AVOD	10,0
AVOD	12,5
RT3DStereo	**12,7**

TABLEAU 4.4 Comparaison du coût de calcul lors de la détection avec les méthodes de la littérature

Méthode	Mémoire (Go)
Pseudo-Lidar	6,5
DispNetCv4 + AVOD	3,3
DispNetC + AVOD	3,0
AVOD	**2**

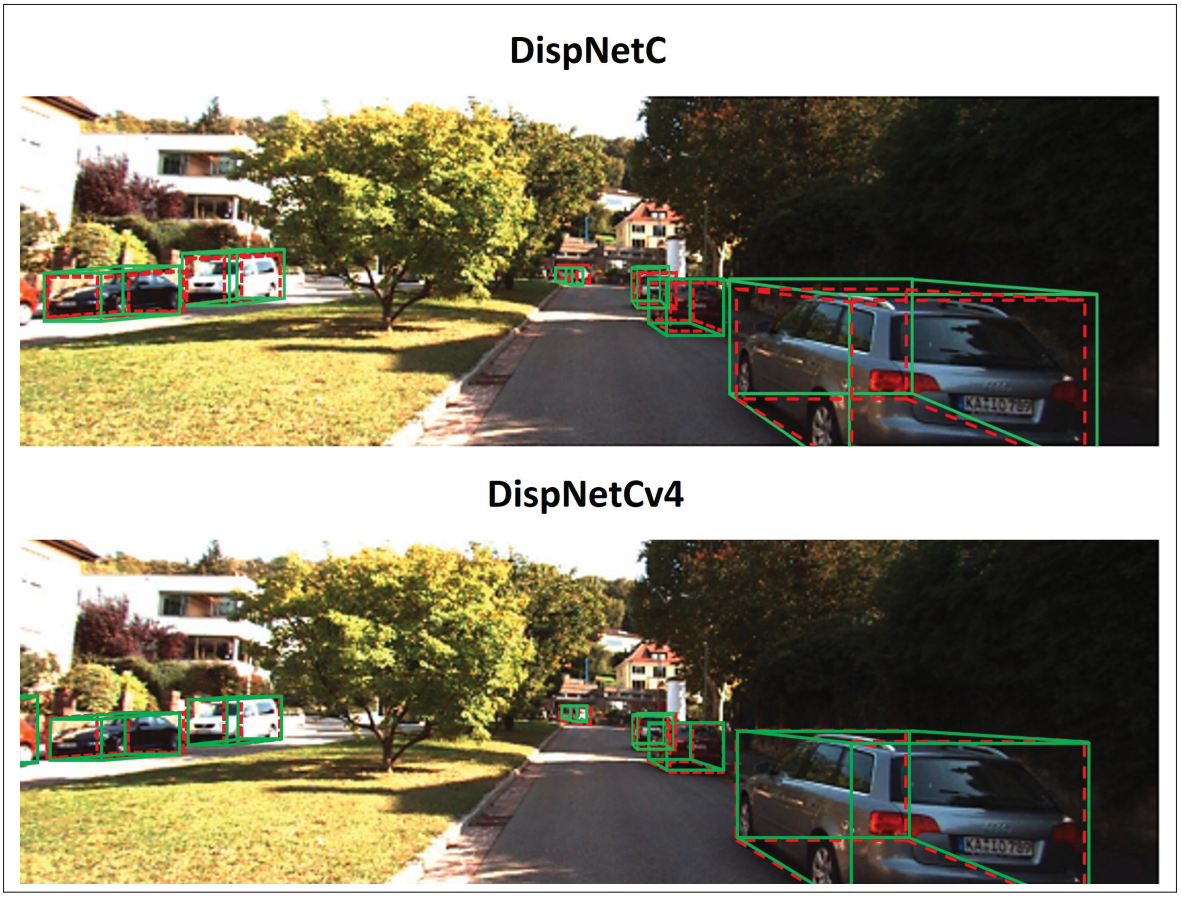

FIGURE 4.3 Comparaison qualitative de détection 3D sur KITTI 2015 entre DispNetC et DispNetCv4 avec le détecteur AVOD*

*Boite *ground-truth* en rouge et les boites prédites en vert.

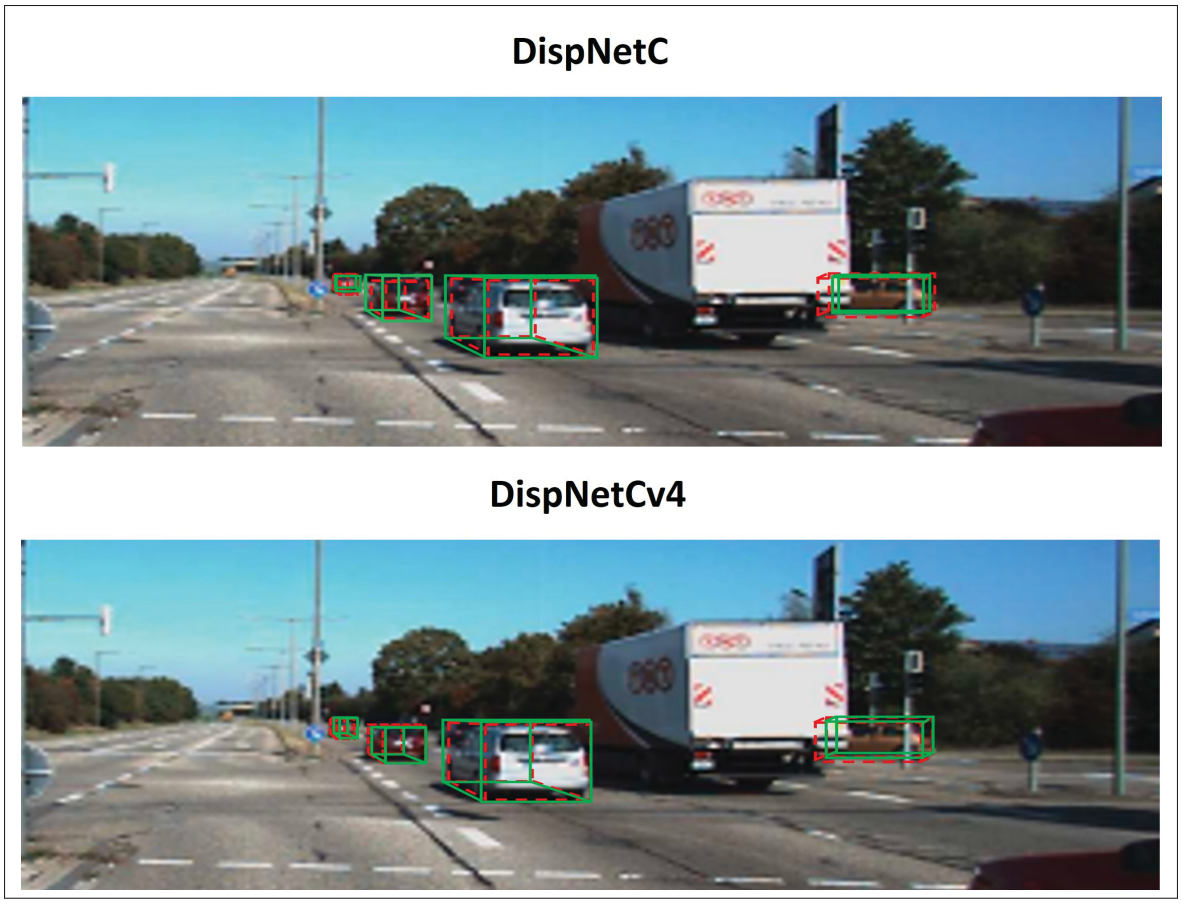

FIGURE 4.4 Comparaison qualitative de détection 3D sur KITTI 2015 entre DispNetC et DispNetCv4 avec le détecteur AVOD

4.4 Conclusion

Nous avons pu voir que la précision d'un détecteur 3D par images stéréos dépend fortement de la représentation 3D de la scène, produite par le réseau de disparité. En s'inspirant du Pseudo-Lidar, nous avons construit un détecteur 3D à partir de notre réseau DispNetCv4 dont les cartes de disparités générées sont converties en nuage de points "pseudo-lidar". Les résultats des expériences ont montré que lors de la prédiction, notre approche permet de gagner en précision, vitesse et empreinte mémoire par rapport à la méthode initiale de Wang *et al.* (2019).
Mais on a aussi pu observer que lorsque les cartes de disparités réelles ne sont pas disponibles, notre méthode d'adaptation non supervisée sur les disparités assure des cartes plus détaillées, permettant ainsi de fournir une représentation 3D mieux structurée pour le détecteur.

CONCLUSION ET RECOMMANDATIONS

L'étude réalisée nous a permis d'explorer l'ensemble des méthodes de la littérature réalisant la détection 3D basée sur des images stéréoscopiques. On a pu voir que les limitations de ces méthodes sont dues à la complexité de la tache : produire une représentation 3D précise de la scène seulement à partir d'images nécessite une méthode robuste, obligeant souvent de faire un compromis entre précision, vitesse et consommation mémoire. Bien que les méthodes de détection 3D par images stéréo se décomposent en 4 grandes familles, on remarque qu'elles partagent presque toutes une étape commune : l'estimation des disparités. Connaître les correspondances entre les pixels de l'image gauche avec ceux de l'image droite est nécessaire pour produire une représentation 3D de la scène à l'aide de deux images seulement. Cependant, cette étape est très compliquée à réaliser à cause des contraintes de luminosité, d'occlusion, de régions sans texture et répétitive... et une petite erreur de disparité aura un impact très important sur l'estimation de la profondeur.

En s'intéressant à la tache d'estimation des disparités, on a pu voir qu'il existe deux grandes familles de méthode : celles employant des convolutions 2D et une architecture U-Net, permettant de gagner en temps et mémoire lors de la prédiction, et celles utilisant des convolutions 3D dont le principe s'inspire des méthodes traditionnelles, permettant d'atteindre une très faible erreur de disparité. Mais un problème essentiel persiste pour chacune des familles : le manque de données réelles labélisées oblige les méthodes à exploiter des données synthétiques pour l'apprentissage, puis à effectuer un affinage sur les quelques données réelles. Cette stratégie d'apprentissage n'est pourtant pas efficace dû à un écart trop important entre les distributions des domaines synthétique et réel. L'une des solutions est alors d'appliquer le principe d'adaptation de domaine non supervisé et plus précisément une approche associant l'apprentissage adverse avec la reconstruction d'image, dont le but est de confondre les deux distributions.

C'est ainsi en cherchant à résoudre les problématiques précédentes, et en s'inspirant des travaux de la littérature, que nous avons développé une méthode de disparité permettant d'atteindre le

meilleur compromis entre vitesse, mémoire et précision lors de la prédiction, tout en assurant une bonne généralisation et adaptation sur les données réelles. La finalité de notre étude est de proposer une méthode produisant une représentation précise de la profondeur pour la tache de détection 3D.

Nous avons choisi le DispNetC comme réseau de base dû à sa simplicité, rapidité et légèreté. En s'inspirant des méthodes U-Net en segmentation sémantique, nous avons ajouté au réseau de base plusieurs améliorations d'agrégation spatiale et locale, d'attention spatial, de channel et de résolution permettant d'augmenter sa précision et robustesse, tout en gardant une vitesse importante. Puis, dans l'objectif de répondre au problème de disponibilité de données réelles, nous avons associé deux méthodes d'adaptation de domaine non-supervisée assurant une meilleure précision sur le domaine réel : le CycleGAN translatant les images synthétiques vers le réel, avec un discriminateur de caractéristiques permettant au réseau d'apprendre une représentation invariante au domaine. Finalement nous avons évalué les capacités de notre méthode à produire une représentation de la scène plus précise par la tache de détection 3D. L'approche du Pseudo-Lidar a été employé pour la détection, du fait de ses capacités à produire une représentation de la profondeur plus détaillée, en transformant la carte de disparité en nuage de points. Les résultats obtenus ont permis de conclure sur l'importance du réseau de disparité pour la détection 3D par stéréo : les améliorations proposées affinent l'information 3D apprise par le détecteur 3D, et renforcent ses capacités d'adaptation sur le domaine réel.

Comme on a pu voir dans l'introduction, le mémoire s'inscrit dans un projet plus large dont l'objectif principale est d'estimer la distance latérale entre un cycliste et un automobiliste lors d'un dépassement. L'un des prochains objectifs sera de considérer la classe "vélo" en plus de "voiture" lors de la détection 3D. Cependant seul la base KITTI propose des annotations 3D pour la classe "vélo", et celle-ci apparaît très peu de fois dans la base de données (4% des annotations contre 70,8% pour la classe "voiture"). Ainsi, l'une des solutions sera de produire une nouvelle base de données pour la détection 3D avec plus d'apparition de cyclistes, et capturant

des situations de dépassement vélo-voiture. La finalité serait d'effectuer un déploiement de la solution complète en situation réelle.

Il serait intéressant par la suite d'appliquer cette stratégie d'apprentissage sur d'autres réseaux de disparités plus précis, comme le Fast DS-CS, HitNet ou DSGN, mais qui sont des architectures plus complexes à prendre en main. De plus, le principe d'adaptation a seulement été étudié sur le réseau de disparité, mais il pourrait être étudié sur l'ensemble de la méthode de détection 3D par stéréo. Il serait aussi intéressant d'appliquer des méthodes permettant d'affiner les cartes de disparité avant la détection, comme à l'aide de quelque points lidar en entrée supplémentaire pour guider le nuage pseudo-lidar (voir You *et al.* (2020)), ou bien grâce à un réseau de *depth completion* (complétage de carte de profondeur) comme proposé par Ma, Cavalheiro & Karaman (2019) ou bien Cheng, Wang & Yang (2020), qui estime des cartes de profondeur denses à partir de cartes clairsemées.

BIBLIOGRAPHIE

Arnold, E., Al-Jarrah, O. Y. & al. (2019). A Survey on 3D Object Detection Methods for Autonomous Driving Applications. *IEEE Transactions on Intelligent Transportation Systems*.

Arruda, V., Paixão, T., Berriel, R., De Souza, A., Badue, C., Sebe, N. & Oliveira-Santos, T. (2019, 07). Cross-Domain Car Detection Using Unsupervised Image-to-Image Translation : From Day to Night. pp. 1-8. doi : 10.1109/IJCNN.2019.8852008.

Atapour-Abarghouei, A. & Breckon, T. (2018). Real-Time Monocular Depth Estimation Using Synthetic Data with Domain Adaptation via Image Style Transfer. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2800-2810.

Azad, R., Asadi-Aghbolaghi, M., Fathy, M. & Escalera, S. (2019). Bi-Directional ConvLSTM U-Net with Densley Connected Convolutions. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 406-415. doi : 10.1109/ICCVW.2019.00052.

Ben Ayed, I. (2019). SYS843 : Réseaux de Neurones et Systèmes flous. École de Technologie Supérieur (ETS).

Bhattacharyya, P., Huang, C. & Czarnecki, K. (2021). SA-Det3D : Self-Attention Based Context-Aware 3D Object Detection.

Cabon, Y., Murray, N. & Humenberger, M. (2020). Virtual KITTI 2.

Cai, C., Poggi, M., Mattoccia, S. & Mordohai, P. (2020). Matching-space Stereo Networks for Cross-domain Generalization. *2020 International Conference on 3D Vision (3DV)*, 364-373.

Chane, m. (2021). Classification des images médicales : comprendre le réseau de neurones convolutifs (CNN) [Web]. Repéré à https://www.imaios.com/fr/Societe/blog/Classification-des-images-medicales-comprendre-le-reseau-de-neurones-convolutifs-CNN.

Chang, J.-R. & Chen, Y.-S. (2018). Pyramid Stereo Matching Network. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5410-5418.

Chen, C. & Chen, X. (2019). On the Over-Smoothing Problem of CNN Based Disparity Estimation. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 8996-9004.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. & Yuille, A. L. (2018a). DeepLab : Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4), 834-848. doi : 10.1109/TPAMI.2017.2699184.

Chen, X., Kundu, K., Zhu, Y., Berneshawi, A. G., Ma, H., Fidler, S. & Urtasun, R. (2015). 3D Object Proposals for Accurate Object Class Detection. *NIPS*.

Chen, X., Kundu, K., Zhu, Y., Ma, H., Fidler, S. & Urtasun, R. (2018b). 3D Object Proposals Using Stereo Imagery for Accurate Object Class Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40, 1259-1272.

Chen, Y., Liu, S. & al. (2020). DSGN : Deep Stereo Geometry Network for 3D Object Detection. pp. 12536-12545.

Chen, Y., Li, W., Sakaridis, C., Dai, D. & Van Gool, L. (2018c, 06). Domain Adaptive Faster R-CNN for Object Detection in the Wild. pp. 3339-3348. doi : 10.1109/CVPR.2018.00352.

Cheng, X., Wang, P. & Yang, R. (2019). Learning Depth with Convolutional Spatial Propagation Network. *IEEE Transactions on Pattern Analysis and Machine Intelligence(T-PAMI)*, 1-1.

Cheng, X., Wang, P. & Yang, R. (2020). Learning Depth with Convolutional Spatial Propagation Network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42, 2361-2379.

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

Dosovitskiy, A., Fischer, P., Ilg, E. & al. (2015). FlowNet : Learning optical flow with convolutional networks. *International Conference on Computer Vision*.

Duggal, S., Wang, S., Ma, W.-C., Hu, R. & Urtasun, R. (2019). DeepPruner : Learning Efficient Stereo Matching via Differentiable PatchMatch.

Gaidon, A., Wang, Q., Cabon, Y. & Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 4340–4349.

Garg, R., Kumar, B. V., Carneiro, G. & Reid, I. D. (2016). Unsupervised CNN for Single View Depth Estimation : Geometry to the Rescue. *ECCV*.

Geiger, A., Lenz, P. & Urtasun, R. (2012). Are we ready for Autonomous Driving ? The KITTI Vision Benchmark Suite. *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580-587. doi : 10.1109/CVPR.2014.81.

Godard, C., Aodha, O. M. & Brostow, G. J. (2017). Unsupervised Monocular Depth Estimation with Left-Right Consistency. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6602-6611.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C. & Bengio, Y. (2014). Generative Adversarial Nets. *NIPS*.

Goodfellow, I. J., Bengio, Y. & Courville, A. C. (2015). Deep Learning. *Nature*, 521, 436-444.

Gu, R., Wang, G., Song, T., Huang, R., Aertsen, M., Deprest, J., Ourselin, S., Vercauteren, T. & Zhang, S. (2021). CA-Net : Comprehensive Attention Convolutional Neural Networks for Explainable Medical Image Segmentation. *IEEE Transactions on Medical Imaging*, 40(2), 699-711. doi : 10.1109/TMI.2020.3035253.

Gu, X., Fan, Z., Zhu, S., Dai, Z., Tan, F. & Tan, P. (2020). Cascade cost volume for high-resolution multi-view stereo and stereo matching. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2495–2504.

He, C., Zeng, H., Huang, J., Hua, X.-S. & Zhang, L. (2020). Structure Aware Single-stage 3D Object Detection from Point Cloud. *CVPR*.

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778. doi : 10.1109/CVPR.2016.90.

Hirschmueller, H. (2008). Stereo processing by semiglobal matching and mutual information. *PAMI*, 30, 328-41.

H.Konigshof, Salscheider, N. & al. (2019). Realtime 3D object detection for automated driving using stereo vision and semantic information. *IEEE Intelligent Transportation Systems Conference (ITSC)*, 1405-10.

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A. A. & Darrell, T. (2018). CyCADA : Cycle-Consistent Adversarial Domain Adaptation. *ICML*.

Hu, J., Shen, L., Albanie, S., Sun, G. & Wu, E. (2020). Squeeze-and-Excitation Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(8), 2011-2023. doi : 10.1109/TPAMI.2019.2913372.

Inoue, N., Yamasaki, T. & Aizawa, K. (2018, 06). Cross-Domain Weakly-Supervised Object Detection Through Progressive Domain Adaptation. pp. 5001-5009. doi : 10.1109/CVPR.2018.00525.

Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5967-5976.

Jha, D., Smedsrud, P. H., Riegler, M. A., Johansen, D., Lange, T. D., Halvorsen, P. & D. Johansen, H. (2019). ResUNet++ : An Advanced Architecture for Medical Image Segmentation. *2019 IEEE International Symposium on Multimedia (ISM)*, pp. 225-2255. doi : 10.1109/ISM46123.2019.00049.

Johnson, J., Alahi, A. & Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. *ECCV*.

Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A. & Bry, A. (2017). End-to-End Learning of Geometry and Context for Deep Stereo Regression. *Proceedings of the International Conference on Computer Vision (ICCV)*.

Khodabandeh, M., Vahdat, A., Ranjbar, M. & Macready, W. G. (2019). A Robust Learning Approach to Domain Adaptive Object Detection. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 480-490.

Kingma, D. P. & Ba, J. (2015). Adam : A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Kordon, A. K. (2020). *Artificial Intelligence-Based Data Science Solutions*. Cham : Springer International Publishing.

Ku, J., Mozifian, M., Lee, J., Harakeh, A. & Waslander, S. (2018). Joint 3D Proposal Generation and Object Detection from View Aggregation. *IROS*.

Kundu, J. N., Uppala, P. K., Pahuja, A. & Babu, R. V. (2018). AdaDepth : Unsupervised Content Congruent Adaptation for Depth Estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2656-2665.

Laga, H. (2019). A Survey on Deep Learning Architectures for Image-based Depth Reconstruction. *ArXiv*, 28.

Li, P., Qin, T. & al. (2018). Stereo vision-based semantic 3D object and ego-motion tracking for autonomous driving. pp. 664-79.

Li, P., Chen, X. & al. (2019). Stereo R-CNN Based 3D Object Detection for Autonomous Driving. pp. 7636-44.

Li, W., Li, F., Luo, Y., Wang, P. & sun, J. (2020). Deep Domain Adaptive Object Detection : a Survey. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1808-1813. doi : 10.1109/SSCI47803.2020.9308604.

Liang, Z., Feng, Y., Guo, Y. & al. (2018). Learning for disparity estimation through feature constancy. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2811–2820.

Liao, Y., Huang, L., Wang, Y., Kodagoda, S., Yu, Y. & Liu, Y. (2017). Parse geometry from a line : Monocular depth estimation with partial laser observation. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 5059-5066.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. & Zitnick, C. (2014). Microsoft COCO : Common Objects in Context.

Liu, L., Ouyang, W. & al. (2019). Deep Learning for Generic Object Detection : A Survey. *International Journal of Computer Vision*.

Liu, R., Yang, C., Sun, W., Wang, X. & Li, H. (2020a). StereoGAN : Bridging Synthetic-to-Real Domain Gap by Joint Optimization of Domain Translation and Stereo Matching. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12754-12763.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C.-Y. & Berg, A. C. (2016). SSD : Single Shot MultiBox Detector. *ECCV*.

Liu, Z., Zhao, X., Huang, T., Hu, R., Zhou, Y. & Bai, X. (2020b). TANet : Robust 3D Object Detection from Point Clouds with Triple Attention. *AAAI*. Repéré à https://arxiv.org/pdf/1912.05163.pdf.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2, 1150-1157 vol.2.

Ma, F. & Karaman, S. (2018). Sparse-to-Dense : Depth Prediction from Sparse Depth Samples and a Single Image. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 1-8.

Ma, F., Cavalheiro, G. V. & Karaman, S. (2019). Self-Supervised Sparse-to-Dense : Self-Supervised Depth Completion from LiDAR and Monocular Camera. *2019 International Conference on Robotics and Automation (ICRA)*, 3288-3295.

Madadi, Y., Seydi, V., Nasrollahi, K., Hossieni, R. & Moeslund, T. B. (2020). Deep visual unsupervised domain adaptation for classification tasks : a survey. *IET Image Process.*, 14, 3283-3299.

Maisonneuve, V. & Lemieux, N. (2018). La Vérif : peut-on mesurer la distance entre un véhicule et un vélo ? [Webpage]. Repéré à https://ici.radio-canada.ca/nouvelle/1095036/verif-securite-route-vehicule-velo-accident-loi-cycliste.

Marr, D. (1982). Vision : a computational investigation into the human representation and processing of visual information.

Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A. & Brox, T. (2016). A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. Repéré à http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16.

Mayer, N. & Ilg, E. (2016). A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4040-4048.

Menze, M., Heipke, C. & Geiger, A. (2015). Joint 3D Estimation of Vehicles and Scene Flow. *ISPRS Workshop on Image Sequence Analysis (ISA)*.

Ojala, T., Pietikäinen, M. & Mäenpää, T. (2002). Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24, 971-987.

OpenCV. Camera calibration With OpenCV [Web]. Repéré à https://docs.opencv.org/3.4/d4/d94/tutorial_camera_calibration.html.

Osuna, E., Freund, R. & Girosit, F. (1997). Training support vector machines : An application to face detection. *CVPR*, 130-136.

Pang, J. H., Sun, W. X., Ren, J. S. J., Yang, C. X. & Yan, Q. (2017). Cascade residual learning : a two-stage convolutional neural network for stereo matching. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 878–886.

Pedersoli, M. (2020). SYS866 : Apprentissage Profond. École de Technologie Supérieur (ETS).

Peng, W., Pan, H. & al. (2020). IDA-3D : Instance-Depth-Aware 3D Object Detection from Stereo Vision for Autonomous Driving. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pham, C. C. & Jeon, J. W. (2017). Robust object proposals re-ranking for object detection in autonomous driving using convolutional neural networks. *Signal Processing : Image Communication*, 53, 110-22.

Pon, A. D. & Ku, J. (2019). Object-Centric Stereo Matching for 3D Object Detection. *Arxiv*, 7.

Qin, Z., Wang, J. & al. (2019). Triangulation Learning Network : From Monocular to Stereo 3D Object Detection. pp. 7607-15.

Radio-Canada, R. (2016). Dépassement des cyclistes : une loi difficile à appliquer, disent les policiers [Webpage]. Repéré à https://ici.radio-canada.ca/nouvelle/800494/velo-automobile-cycliste-automobiliste-route-distance.

Rahman, M. M., Tan, Y. & al. (2020). Recent Advances in 3D Object Detection in the Era of Deep Neural Networks : A Survey. *IEEE Transactions on Image Processing*, 29.

Ranjan, A. & Black., M. J. (2017). Optical flow estimation using a spatial pyramid network. *IEEE Conference on Computer Vision and Pattern Recognition*.

Ren, S., He, K., Girshick, R. & Sun, J. (2015). Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems (NIPS)*.

Rodriguez, A. L. & Mikolajczyk, K. (2019). Domain Adaptation for Object Detection via Style Consistency. arXiv. Repéré à https://arxiv.org/abs/1911.10033.

Ronneberger, O., Fischer, P. & Brox, T. (2015). U-net : Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241.

Rowley, H., Baluja, S. & Kanade, T. (1998). Neural network based face detection. *IEEE TPAMI*, 20(1), 23-38.

SAAQ, S. d. l. a. d. Q. (2021). Bilan routier 2021 - Faits saillants [PDF]. Repéré à https://saaq.gouv.qc.ca/fileadmin/documents/publications/bilan-routier-2021.pdf.

Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nesic, N., Wang, X. & Westling, P. (2014). High-Resolution Stereo Datasets with Subpixel-Accurate Ground Truth. *GCPR*.

Schöps, T., Schönberger, J. L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M. & Geiger, A. (2017). A Multi-view Stereo Benchmark with High-Resolution Images and Multi-camera Videos. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2538-2547.

Shen, Z., Maheshwari, H., Yao, W. & Savvides, M. (2019). SCL : Towards Accurate Domain Adaptive Object Detection via Gradient Detach Based Stacked Complementary Losses. *ArXiv*, abs/1911.02559.

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J. M., Wang, W. & Webb, R. (2017). Learning from Simulated and Unsupervised Images through Adversarial Training. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2242-2251.

Spencer, J. & Bowden, R. (2019). Scale-Adaptive Neural Dense Features : Learning via Hierarchical Context Aggregation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6193-202.

SPVM, S. d. p. d. l. V. d. M. (2022). Dépasser un cycliste [Webpage]. Repéré à spvm.qc.ca.

Statistique Canada, S. (2019). Les circonstances entourant les décès liés au cyclisme au Canada, 2006 à 2017 [PDF]. Repéré à https://www150.statcan.gc.ca/n1/fr/pub/82-625-x/2019001/article/00009-fra.pdf?st=_RabA_gK.

Sun, J., Chen, L. & al. (2020). Disp R-CNN : Stereo 3D Object Detection via Shape Prior Guided Instance Disparity Estimation. pp. 10548-10557.

Tan, M. & Le, Q. V. (2019). Efficientnet : Rethinking model scaling for convolutional neural networks. *36th International Conference on Machine Learning*.

Tan, M., Pang, R. & Le, Q. V. (2020). EfficientDet : Scalable and Efficient Object Detection. *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tankovich, V., Häne, C., Zhang, Y., Kowdle, A., Fanello, S. & Bouaziz, S. (2021). HITNet : Hierarchical Iterative Tile Refinement Network for Real-time Stereo Matching. *CVPR*.

Tardif, F. (2021). Les infractions et les sanctions reliées à la conduite d'un véhicule routier, 2010-2019 [PDF]. Repéré à https://saaq.gouv.qc.ca/fileadmin/documents/publications/espace-recherche/dossier-statistique-infraction-routier-2019.pdf.

Tonioni, A., Tosi, F., Poggi, M., Mattoccia, S. & Stefano, L. D. (2019, June). Real-Time self-adaptive deep stereo. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tzeng, E., Hoffman, J., Saenko, K. & Darrell, T. (2017). Adversarial Discriminative Domain Adaptation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2962-2971.

Vijayanarasimhan, S., Ricco, S., Schmid, C., Sukthankar, R. & Fragkiadaki, K. (2017). SfM-Net : Learning of Structure and Motion from Video. *ArXiv*, abs/1704.07804.

Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *CVPR*, 1, 1-8.

Vélo Québec, V. (2021). L'état du vélo au Québec en 2020 [PDF]. Repéré à https://www.velo.qc.ca/wp-content/uploads/2021/06/vq-edv2020-fr.pdf.

Wang, Q., Shi, S., Zheng, S., Zhao, K. & Chu, X. (2020). FADNet : A Fast and Accurate Network for Disparity Estimation. *arXiv*.

Wang, Y., Chao, W.-L. & al. (2019). Pseudo-LiDAR from Visual Depth Estimation : Bridging the Gap in 3D Object Detection for Autonomous Driving. pp. 8437-45.

Weng, X. & Kitani, K. (2019). A Baseline for 3D Multi-Object Tracking. *arXiv :1907.03961*. Repéré à https://arxiv.org/pdf/1907.03961.pdf.

Vision par ordinateur. (2006). Dans *Wikipedia*. Repéré le 2021-05-20 à https://fr.wikipedia.org/wiki/Vision_par_ordinateur.

Stéréoscopie. (2014). Dans *Wikipedia*.

Wilson, G. & Cook, D. J. (2020). A Survey of Unsupervised Deep Domain Adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11, 1 - 46.

Xu, H. & Zhang, J. (2020). AANet : Adaptive Aggregation Network for Efficient Stereo Matching. *CVPR*.

Xu, Z. & al. (2020). ZoomNet : Part-Aware Adaptive Zooming Neural Network for 3D Object Detection. 2, 7.

Yang, Z., Sun, Y., Liu, S., Shen, X. & Jia, J. (2019). STD : Sparse-to-Dense 3D Object Detector for Point Cloud. *ICCV*. Repéré à http://arxiv.org/abs/1907.10471.

Yee, K. & Chakrabarti, A. (2020). Fast Deep Stereo with 2D Convolutional Processing of Cost Signatures. *WACV*.

Yin, Z. & Shi, J. (2018). GeoNet : Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1983-1992.

You, Y., Wang, Y. & al. (2020). Pseudo-LiDAR++ : Accurate Depth for 3D Object Detection in Autonomous Driving. *International Conference on Learning Representations*.

Zbontar, J. & LeCun, Y. (2015). Computing the Stereo Matching Cost with a Convolutional Neural Network. pp. 592–1599.

Zhang, Y., Chen, Y., Bai, X., Yu, S., Yu, K., Li, Z. & Yang, K. (2020). Adaptive Unimodal Cost Volume Filtering for Deep Stereo Matching. *AAAI*.

Zhao, H., Shi, J., Qi, X., Wang, X. & Jia, J. (2017). Pyramid Scene Parsing Network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6230-6239. doi : 10.1109/CVPR.2017.660.

Zhao, S., Yue, X., Zhang, S., Li, B., Zhao, H., Wu, B., Krishna, R., Gonzalez, J., Sangiovanni-Vincentelli, A. L., Seshia, S. A. & Keutzer, K. (2022). A Review of Single-Source Deep Unsupervised Visual Domain Adaptation. *IEEE Transactions on Neural Networks and Learning Systems*, 33, 473-493.

Zheng, C., Cham, T. & Cai, J. (2018). T2Net : Synthetic-to-Realistic Translation for Solving Single-Image Depth Estimation Tasks. *ECCV*.

Zhou, K., Meng, X. & al. (2020). Review of Stereo Matching Algorithms Based on Deep Learning. *Computational Intelligence and Neuroscience*.

Zhou, T., Brown, M. A., Snavely, N. & Lowe, D. G. (2017). Unsupervised Learning of Depth and Ego-Motion from Video. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6612-6619.

Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. (2017). Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2242-2251.