Généralisation de domaine pour la segmentation sémantique de données LiDAR pour le véhicule autonome

Jules Sanchez

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à Mines Paris - PSL

Généralisation de domaine pour la segmentation sémantique de données LiDAR pour le véhicule autonome

Soutenue par
Jules SANCHEZ
Le 05 décembre 2023

Dirigée par
François Goulette
codirigée par
Jean-Emmanuel Deschaud

Composition du jury :

Raphaëlle, CHAINE Professeure, Université de Lyon	*Présidente*
Paul, CHECCHIN Professeur, Université de Clermont-Ferrand	*Rapporteur*
Loïc, LANDRIEU HDR, Ecole des Ponts Paristech	*Rapporteur*
Jean-Emmanuel, DESCHAUD HDR, Mines Paris - PSL	*Examinateur*
François, GOULETTE Professeur, ENSTA	*Directeur de thèse*

École doctorale n° 621
Ingénierie des Systèmes, Matériaux, Mécanique, Energétique

Spécialité
Informatique temps réel, robotique et automatique

MINES PARIS | PSL

Remerciements

L'exercice de la thèse est un travail à la fois très solitaire, où l'on se retrouve souvent seul face à de nombreux défis et problèmes à résoudre, mais aussi profondément collectif. Alors je dédie cette section à toutes les personnes qui ont rendu cette expérience vivable, enrichissante, appréciable et inoubliable.

Merci à mes deux directeurs de thèse François et Jean-Emmanuel qui ont fait preuve d'une grande bienveillance, de beaucoup de patience et de pédagogie. Après plus de trois ans à travailler avec eux, je ne peux que les recommander à n'importe quel étudiant qui lirait ce manuscrit.

Merci au jury d'avoir accepté de venir m'évaluer, et notamment aux deux rapporteurs Loic Landrieu et Paul Checchin pour leurs remarques qui ont permis d'enrichir ce travail.

Merci à l'équipe NPM3D, Sofiane, Jean-Pierre, Pierre, Louis, Hugo, Fabio, Samir, Jonas et Agapius (membres honoraires) pour l'intégralité des bons moments passés ensemble que ca soit autour d'un bureau ou d'une table à manger. Plus globalement merci à l'ensemble des gens du CAOR pour créer une atmosphère propice à la recherche et à l'épanouissement.

Merci à tous mes amis que j'ai rencontré aux cours des dernières années et qui ont conitnué de me soutenir dans les bons et mauvais moments durant tout ce temps. Alors, en esperant n'oublier personnes, merci à Marco, Florynde, Victor, Hélène, Colin, Benoit, Colin, 19, Arthur, Bertrand, Leila, Rodrigue, Anna, Bruno, Anna, Elise, Tristan, Martin, Arthur, Wilson, Lucas, Joel (les doublons ne sont pas des erreurs, bonne chance pour vous reconnaitre). On prend un verre ensemble le plus tôt possible !

Merci à Mohamed de m'avoir montré que la recherche était une voie poursuivable dans laquelle on pouvait s'amuser. C'est un plaisir d'avoir pu travailler avec toi durant mes deux stages.

Merci Gehy pour l'amour dont tu m'inondes, sans toi mon quotidien serait bien moins magique.

Merci à ma famille, mes parents et mes trois frères qui ont supporté mes caprices d'organisation et de disponibilités. Et plus spécifiquement à ma maman pour son soutien indéficitible pendant près de 25 ans d'étude. Merci et je t'aime.

Sans vous, je n'aurai pas pu le faire, alors encore une fois merci.

Résumé en français

La perception LiDAR pour le véhicule autonome est parvenue à atteindre des résultats convenables sur les différents benchmarks en ligne dans le cadre monodomaine, c'est-à-dire quand le domaine d'entraînement est le même que celui d'évaluation. A partir de là, les champs de recherche se sont diversifiés et orientés sur les questions de transférabilité, de robustesse et de généralisation.

Ce travail se concentre sur les questions de généralisation pour la segmentation sémantique LiDAR. Un panorama global des performances de généralisation, monosource et multisource, des méthodes de segmentation existante est réalisée. Pour réaliser équitablement ces expériences, un jeu de données, ParisLuco3D, est créé spécifiquement pour évaluer la généralisation.

De plus, une nouvelle méthode de généralisation de domaine monosource, 3DLabelProp, est proposée. Cette méthode se distingue des stratégies existantes en exploitant la géométrie des données pour faire de l'alignement de domaine plutôt que des stratégies par apprentissage. Au-delà de la segmentation sémantique, 3DLabelProp est aussi appliquée à la tâche de moving object segmentation.

Abstract

LiDAR-based perception for autonomous vehicles has reached statisfying performance on the various online monodomain benchmarks, i.e when the training and evaluation domains are the same. From there, research areas diversified and focused on the transferability of perception models, their robstuness and their généralisation capabilities.

This work tackles domain generalization of LiDAR semantic segmentation. An in-depth benchmark of the mono- and multi-source generalization abilities of various state-of-the-art semantic segmentation models is proposed. In order to provide meaningful analysis and conclusions, a generalization-oriented dataset is provided, ParisLuco3D.

Furthermore, a novel mono-source domain generalization method, 3DLabelProp, is presented. This method seperates itself from existing approaches by exploiting geometric information to achieve domain alignment instead of focusing on learning-based domain alignement. Besided semantic segmentation, 3DLabelProp is also applied for moving object segmentation.

Généralisation de domaine pour la segmentation sémantique de données LiDAR pour le véhicule autonome

Jules Sanchez

05 décembre 2023

Table des matières

Liste des figures

Liste des tableaux

Glossaire & Acronymes

Glossaire

off-road :	terrain accidenté dans lequel les véhicules peuvent évoluer, comme les chemins de terre et les champs.
segmentation sémantique :	tâche qui consiste à associer à chaque élément atomique (pixel ou point) une classe, permettant de distinguer des régions uniformes selon ce qu'elles représentent.
label :	synonyme de classe, correspond à la valeur à laquelle on associe des points dans la segmentation sémantique. Dans ce travail, cela correspond au type d'objet.
open-source :	paradigme de partage de codes ou de jeux de données consistant à les rendre publiques et disponibles.
généralisation de domaine :	capacité à conserver les compétences d'un algorithme lors que le domaine d'évaluation est différent de celui d'entraînement, sans exposition au domaine d'évaluation.
monodomaine :	cadre de travail où le domaine d'entraînement et d'évaluation sont les mêmes.
multidomaine :	cadre de travail où le domaine d'entraînement et d'évaluation sont différents.
monosource :	utilisation d'un jeu de données lors de l'entraînement.
multisource :	utilisation de plusieurs jeux de données lors de l'entraînement.
domain shifts :	ensemble des variations pouvant rendre deux domaines différents l'un de l'autre.
vanishing gradient :	phénomène apparaissant dans les réseaux profonds lors de l'entraînement où le gradient dans les couches les plus profondes tend vers 0 et empêche la convergence.

encoder : partie d'un réseau de neurones qui extrait une réprésentation des données et les réduits à un vecteur.

decoder : partie d'un réseau de neurones prenant en entrée une représentation des données vectorielle et en extrait l'information désirée selon la tâche.

range image : image obtenue par la projection sphérique d'un nuage de points, cette image contient l'information de distance au capteur de chaque pixel.

sensor shift : l'ensemble des facteurs liés au capteur qui peuvent avoir un impact sur la topologie du nuage de points acquis. Cela correspond donc d'une part aux spécifications du capteur comme sa technologie, sa résolution verticale et horizontale, son nombre de fibres, et d'autre part à ses caractéristiques externes notamment son positionnement.

scene shift : l'ensemble des facteurs liés à la composition de la scène acquise dont son agencement, la quantité d'éléments qui la composent, les facteurs climatiques et météorologiques ainsi les règles de circulation.

appearance shift : l'ensemble des facteurs liés à l'aspect visuel de chaque élément de la scène comme la végétation, les véhicules et la route.

nuage pseudo-dense : nuage de points obtenus par l'accumulation d'une séquence de nuages de points mis dans un référentiel commun.

coarse labels : jeu de labels grossiers permettant d'être appliqué à tous les jeux de données pour la segmentation sémantique pour le véhicule autonome.

Acronymes

IEEE :	Institute of Electrical and Electronics Engineers
LiDAR :	Light Detection And Ranging
SAE :	Society of Automotive Engineers
MOS :	Moving Object Segmentation
COLA :	COarse LAbels
SK :	SemanticKITTI
NS :	nuScenes
K360 :	KITTI-360
SP :	SemanticPOSS
PFF :	PandaFF
P64 :	Panda64
W :	Waymo
PL3D :	ParisLuco3D
SKT :	SemanticKITTI-Tracking
AP :	Apollo

Chapitre 1

Introduction

Sommaire

1.1 Véhicules autonomes

1.1.1 Contexte

Depuis *Elmer and Elsie*, les premiers robots autonomes capables de suivre la lumière dans les années 40, la robotique autonome a fait des avancées spectaculaires, avec le début des courses de taxis payantes 100% autonomes aux Etats-Unis en 2023. Dans cet élan d'enthousiasme financier, industriel et académique, le véhicule autonome est devenu un sujet central de la robotique. Cette importance est cristallisée notamment par la création de la IEEE *Intelligent Transportation Systems Society* en 2005, la codification légale de la définition d'un véhicule autonome dans de nombreux pays et organismes internationaux (R. 3151-1 du code des transports en France) ainsi que la publication de normes internationales pour encadrer ces définitions (SAE J3016, voir tableau 1.1).

Niveau SAE	Description	Exemple
0 - *No automation*	Toutes les tâches sont réalisées par le conducteur.	N/A
1 - *Driver assistance*	Quelques aides à la conduite peuvent effectuer des tâches légères sur la vitesse ou le contrôle.	Limiteur de vitesse
2 - *Partial automation*	Le véhicule peut avoir des fonctions mixtes. Le conducteur doit participer à la conduite.	Régulateur de vitesse
3 - *Conditional automation*	Le conducteur doit pouvoir reprendre le contrôle du véhicule, mais n'a pas besoin de participer à la conduite.	Gestion des bouchons
4 - *High automation*	Le véhicule peut réaliser la conduite en conditions restreintes.	Taxi automatique localisé
5 - *Full automation*	Le véhicule peut réaliser la conduite en toutes conditions.	

TABLE 1.1 – Niveaux SAE (*Society of Automotive Engineers*), décrivant les niveaux d'automatisation des véhicules autonomes.

Scientifiquement, les débuts réalistes des véhicules autonomes datent des années 80 et 90 avec les projets American Navlab et ALV aux Etats-Unis, et le projet European Eureka Prometheus en Europe. Bien que l'automatisation complète (niveau 5 au sens SAE J3016) des véhicules autonomes n'ait pas encore été atteinte, les niveaux inférieurs ont été successivement atteints et intégrés dans des véhicules du quotidien (suivi de ligne, régulateur de vitesse, taxi autonome).

L'objectif à long terme est d'atteindre le niveau 5, et pour cela plusieurs sous-tâches permettant le fonctionnement d'un véhicule autonome ont été identifiées. Bien que la nomenclature exacte de ces tâches soit débattue, [1] propose d'en identifier 5 disjointes : **Perception**, **Localization**, **Path Planning**, **Decision Making** & **Control**, voir la figure 1.1. Le travail proposé s'intéresse uniquement à la question de la perception.

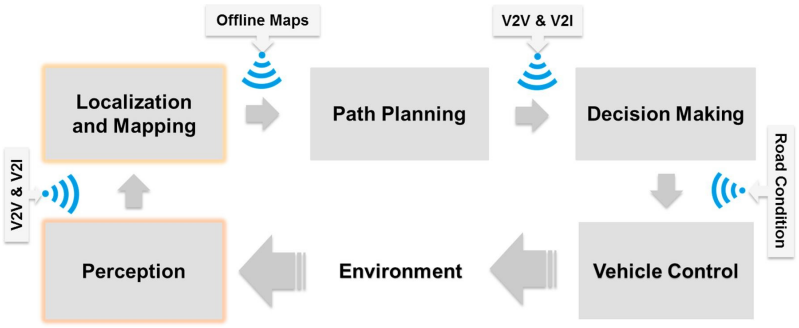


FIGURE 1.1 – Schéma des blocs fonctionnels nécessaires au véhicule autonome par [1].

1.1.2 La perception pour le véhicule autonome

En pratique, l'environnement est "vu" par les différents capteurs embarqués sur le véhicule, dont les caméras, les LiDAR et les RADAR. [2] propose une classification exhaustive de ces différents capteurs utilisés et leurs intérêts.

L'objectif de la perception est la compréhension et l'interprétation des sorties de ces capteurs. Le terme de compréhension est vaste et regroupe implicitement différentes tâches différenciées par la granularité de l'information qu'elles souhaitent extraire ainsi que par leur format. Un schéma listant les différentes tâches de la perception pour le véhicule autonome se trouve sur la figure 1.2.

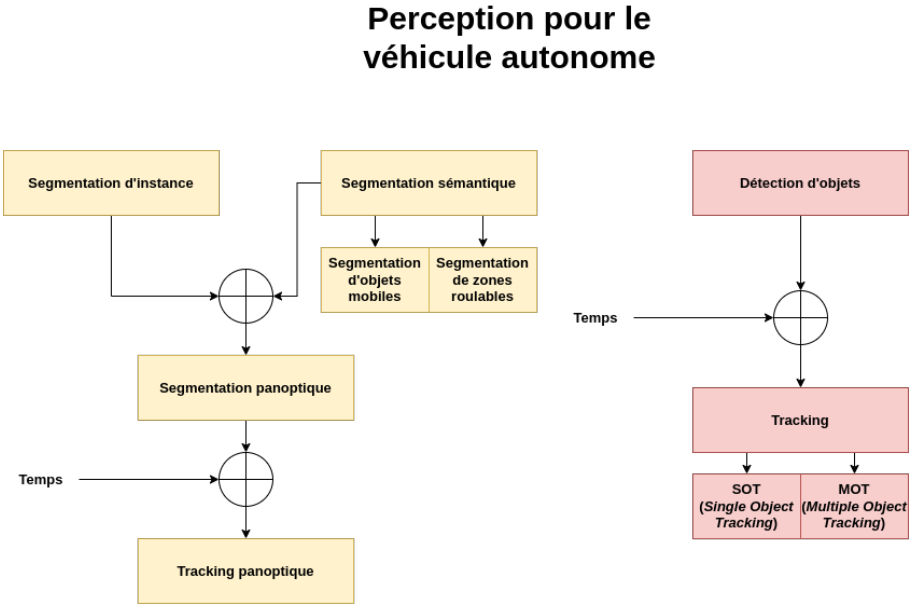
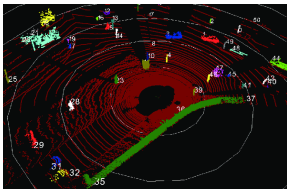


FIGURE 1.2 – Les tâches de perception pour le véhicule autonome.

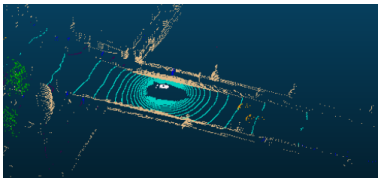
Les tâches de perception peuvent être découpées en deux grandes catégories : basée segmentation et basée détection.

La perception basée segmentation donne une information atomique, c'est-à-dire par pixel pour les images ou par point pour les nuages de points. Parmi les types de segmentation, la segmentation de zone roulable est une tâche particulièrement importante pour les véhicules *off-road* car elle permet de distinguer les endroits permettant un passage sûr. La tâche de tracking panoptique est la plus complexe car elle fusionne une information sémantique, la connaissance du type d'objet auquel appartient chaque point, et une information d'instance, la distinction des instances entre elles et leur suivi dans le temps.

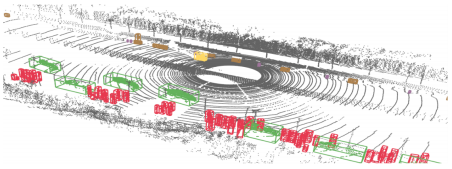
La perception basée détection donne une information au niveau de l'objet, sans nécessairement avoir besoin d'un ancrage sur les points ou pixels, et permet de mettre en évidence des informations occultées ainsi que des informations d'orientation. La tâche principale est le tracking qui permet de faire de la détection d'objet dans le temps. Le tracking est important pour le *motion forecasting* qui est la prédiction du déplacement des agents dans la circulation.



(a) Segmentation panoptique.



(b) Segmentation sémantique.



(c) Détection d'objet.

FIGURE 1.3 – Illustration de certaines tâches de perception pour le véhicule autonome.

Parmi ces tâches, deux se distinguent par l'intérêt que la communauté scientifique leur porte : la segmentation sémantique et la détection d'objets. Cela s'explique par l'utilité directe de ces tâches dans l'extraction d'information pour la navigation des véhicules, la segmentation sémantique permettant de localiser notamment les zones roulables et la détection d'objets permettant d'identifier précisément les autres usagers de l'espace de navigation. De plus, elles peuvent être vues comme des tâches initiales dont dépendent des tâches plus sophistiquées. Il est difficile d'envisager un bon algorithme de tracking alors que la détection d'objet est insatisfaisante. Certaines des tâches de perception sont illustrées sur la figure 1.3.

1.1.3 Enjeux et contraintes

Dans la dernière décennie, les tâches de perception ont recueilli un intérêt croissant de la part de la communauté scientifique. L'effet résultant principal est l'augmentation du nombre de jeux de données *open-source*, ce qui a permis l'avènement des méthodes d'apprentissage profond pour la perception automobile. Malgré les très grands gains de performance quantitative, illustrés par les écarts de performances entre les méthodes basées sur l'apprentissage profond et celle fondées sur l'apprentissage machine sur les différents *benchmarks*[1], la perception est loin d'être considérée comme résolue, et notamment [1] met en avant différents défis à surmonter :

— (1) perception dans le cadre de mauvaises conditions d'éclairage et de météo,
— (2) perception dans le cadre d'environnements urbains complexes,
— (3) conduite autonome sans la disponibilité d'information de perception a priori,
— (4) utilisation de véhicules connectés pour améliorer la précision, la fiabilité et la confiance de la perception,
— (5) développement de mesures de sécurité en cas de capteurs, ou modules de perception, défaillants.

Au vu de ces défis mis en avant et des performances actuelles de l'état de l'art, présentées dans le chapitre suivant, ce travail se concentrera sur le traitement des données LiDAR.

Tout d'abord, la donnée LiDAR, contrairement aux images RGB habituelles, a certaines propriétés précieuses pour la perception. Plus précisément, les capacités d'acquisition des capteurs LiDAR ne sont pas dégradées en cas de mauvaises conditions d'illumination (défi 1) et permettent d'obtenir une information géométrique précise (défi 4). Il faut cependant noter que les LiDAR sont sensibles aux conditions météorologiques extrêmes tels que les pluies intenses, les brouillards épais et la neige.

De plus, comme mis en avant par [2], de nombreux LiDAR sont déjà embarqués dans les véhicules utilisés dans le monde académique et industriel.

L'utilisation des données LiDAR s'accompagne en contrepartie de contraintes différentes de celles du traitement d'images traditionnelles. Pour commencer, habituellement les capteurs LiDAR produisent une grande quantité de données et donc nécessitent des techniques de traitement de données particulièrement efficaces. Le temps d'acquisition habituel pour un capteur LiDAR rotatif, qui est la technologie la plus utilisée pour le véhicule autonome, est de 100 ms. On considère donc que pour être temps-réel, les méthodes doivent atteindre une fréquence d'exécution de 10 Hz.

En outre, alors que les caméras et appareils photos peuvent être regroupés en majoritairement deux technologies, *pin-hole camera* et *fisheye camera*, il existe un grand nombre de types de capteurs LiDAR. De plus, les questions de technologie se répercutent aussi sur la réflectivité. Cette grandeur est renvoyée par les capteurs LiDAR qui donnent pour chaque point une information d'intensité du signal retour du capteur. Bien que cette valeur soit liée à des questions physiques, comme le type des matériaux des surfaces intersectées et l'angle d'incidence des rayons d'acquisition, leur calcul exact est propriétaire. Cela fait

1. https ://npm3d.fr/paris-lille-3d

qu'à chaque modèle de LiDAR est associé une réflectivité différente. Il arrive même que pour un modèle donné, une mise à jour vienne changer le calcul de cette grandeur.

Par ailleurs, les questions de résolution de l'acquisition sont plus complexes pour le LiDAR, qui capture une information géométrique précise, et il est donc plus ardu de rendre les modules de perception robustes à un changement d'installation de capteurs.

Ce travail s'intéresse particulièrement à deux des défis présentés, qui sont la question de la fiabilité de la perception et de son utilisation en milieux urbains denses (défis 2 & 4). Plus précisément, la segmentation sémantique sera étudiée comme tâche de perception. Elle est choisie pour la finesse de l'information qu'elle extrait d'une scène, aussi bien de l'arrière-plan que des acteurs de la route. De plus, atteindre de bonnes performances sur cette tâche permet de réaliser de la segmentation panoptique, de la *moving object segmentation* ainsi que du tracking panoptique, faisant de la segmentation sémantique une tâche essentielle.

1.2 Objectif de cette thèse

1.2.1 La généralisation de domaine

Le cœur de ce travail est consacré à la segmentation sémantique de données LiDAR pour le véhicule autonome en milieu urbain. Pour rappel, la segmentation sémantique a pour objectif d'associer à chaque point d'un nuage de points LiDAR une valeur sémantique appelé label, c'est-à-dire à quel type d'élément ce point appartient. Lors de la présentation des jeux de données existants pour cette tâche, on observera que chaque jeu de labels est différent. Cependant, il existe un consensus sur les catégories d'éléments à distinguer, l'ensemble des jeux de données s'accordant a minima sur les catégories suivantes : véhicules. : véhicules, piétons, sol, objets et végétation.

La majorité des travaux de la littérature 3D s'est intéressée à la question de l'adaptation de domaine, qui a des applications industrielles directes, qui s'intéresse à la question de la transférabilité des connaissances. Cependant, pour s'intéresser à la fiabilité des systèmes de perception, l'angle choisi est celui de la **généralisation de domaine**. Il s'agit, dans le cadre de l'apprentissage profond, de la capacité d'un modèle à conserver ses performances lorsque le domaine d'inférence est différent du ou des domaines vus pendant l'entraînement. Contrairement à l'adaptation de domaine, la généralisation de domaine suppose l'impossibilité d'avoir un accès préalable aux données d'inférence, et donc les performances de généralisation sont scellées dès la fin de l'entraînement.

Plus précisément, la généralisation de domaine s'intéresse à la résilience des performances soit lorsque le contexte de la scène change, tel que le type et le comportement des usagers, soit lorsque le contexte d'acquisition change. Conserver ses performances quand la résolution du LiDAR change pourrait garantir une conservation des performances en cas de défaillance partielle du LiDAR embarqué.

Les performances récentes des derniers modèles de segmentation ont atteint des résultats très satisfaisants quantitativement (environ 70% mIoU à travers les 19 classes du jeu de données de référence SemanticKITTI [3]) dans le cas monodomaine, lorsque le

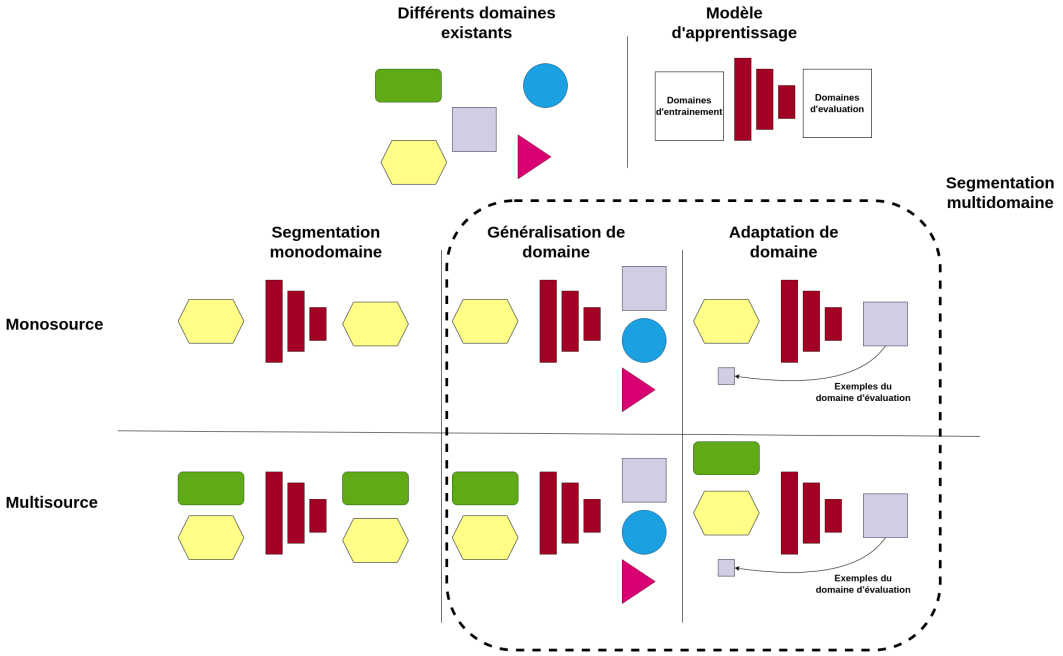


FIGURE 1.4 – Illustration des concepts de segmentation monodomaine, généralisation de domaine et adaptation de domaine.

domaine d'entraînement et d'évaluation est le même. Il est donc naturel de s'intéresser aux performances multidomaine, lorsque le domaine d'entraînement est différent de celui d'évaluation. Les variations de domaines, appelés *domain shifts*, sont diverses et seront présentées exhaustivement par la suite, mais on peut mentionner notamment le changement de type de scène tel que passer d'une banlieue pavillonnaire à un centre urbain dense. Cet exemple est un cas d'application concret pour atteindre la norme SAE 5 car il n'est pas réalisable logistiquement d'acquérir des données dans la totalité des types de scènes pour l'entraînement.

Un schéma illustrant les différents contextes de travail, soit monodomaine, généralisation de domaine et adaptation de domaine est proposé dans la figure 1.4.

1.2.2 Contributions & plan

Mes contributions peuvent être synthétisées en trois points : la création d'un jeu de données, ParisLuco3D [Soumission 3DV, 2024, en cours de révision], permettant de réaliser de l'évaluation multidomaine de manière précise ; le premier panorama des performances de généralisation de domaine des principales méthodes de l'état de l'art dans le cadre monosource [ICCV, 2023], où un seul type de domaine est disponible à l'entraînement, et dans le cadre multisource [ICRA, 2023] [T-RO], où plusieurs domaines sont disponibles à l'entraînement ; finalement une nouvelle méthode de généralisation de domaine, 3DLabelProp [ICCV, 2023] [PAMI], basée sur l'alignement de domaine géométrique, est proposée.

Plus précisément, l'objectif général de ce travail est de comprendre les interactions entre les types d'architecture de segmentation sémantique actuels et les différents *domain shifts*. Pour cela, un large état de l'art, d'une part des méthodes de segmentation, et d'autre part des jeux de données existants est dressé dans le chapitre 2.

La classification des jeux de données entre entraînement et évaluation et leur analyse met en évidence la difficulté de réaliser de l'évaluation multidomaine, essentielle pour la généralisation de domaine. Pour répondre à ce problème, ParisLuco3D est introduit. L'intérêt de ParisLuco3D est double : premièrement, ce jeu de données est acquis dans le centre de Paris (6e arrondissement) et présente une densité inhabituelle de piétons dans la scène ; deuxièmement, son jeu de labels exhaustifs permet une évaluation multidomaine précise. Ceci est présenté dans le chapitre 3.

Alors que la plupart des méthodes de généralisation monosource se comparent à des méthodes tirées de l'apprentissage machine traditionnel ou du traitement d'image 2D, toutes ces méthodes se concentrent sur une seule architecture et ne s'intéressent pas à comprendre l'influence des designs architecturaux et du choix de la représentation des nuages de points dans les performances de généralisation. Pour combler ce trou dans la littérature, un panorama de la généralisation monosource est effectué dans le chapitre 4.

Au vu de la faiblesse des performances natives de généralisation mises en évidence, nous rejoignons les conclusions tirées par les autres chercheurs du domaine de la nécessité de méthodes de généralisation spécifiquement pensées pour la 3D. Pour cela, deux stratégies sont proposées. Premièrement, on utilise la particularité géométrique de la donnée 3D, associée à sa séquentialité comme cela est fait dans la localisation pour réaliser de l'alignement de domaine et rendre plus robuste les méthodes de segmentation aux changements de capteurs, et donc rendre ces méthodes plus fiables aux défaillances capteurs avec 3DLabelProp dans le chapitre 4. Deuxièmement, dans le chapitre 5, une étude de généralisation multisource est réalisée avec une stratégie appelée COLA (COarse LAbelling) exploitant des *coarse labels* et le grand nombre de nuages de points disponibles pour s'entraîner. Cela permet de rendre les méthodes de segmentation plus robustes aux changements de scènes en exploitant la variété des jeux de données.

Pour conclure dans le chapitre 6, une extension du travail est réalisée sur la *moving object segmentation* pour mettre en évidence que les stratégies précédemment présentées peuvent être appliqué à d'autres tâches que la segmentation sémantique.

1.3 Communications scientifiques

Ce travail de thèse a abouti à plusieurs communications scientifiques, avec certains travaux en cours de finalisation.

Conférences avec actes

[3DV, 2024] J. Sanchez, L. Soum-Fontez, J.-E. Deschaud et F. Goulette, "ParisLuco3D : A high-quality target dataset for domain generalization of LiDAR perception," 2023 International Conference on 3D Vision (3DV), Davos, Switzerland, 2024. **Soumis, en attente de décision**

[ICCV, 2023] J. Sanchez, J.-E. Deschaud et F. Goulette, "Domain generalization of 3D semantic segmentation in autonomous driving," 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2023.

[ICRA, 2023] J. Sanchez, J.-E. Deschaud and F. Goulette, "COLA : COarse LAbel pre-training for 3D semantic segmentation of sparse LiDAR datasets," 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, 2023.

Séminaires sans actes

[ISIS, 2023] J. Sanchez, J.-E. Deschaud et F. Goulette, "Généralisation de domaines monosource pour la segmentation sémantique LiDAR pour le véhicule autonome : utiliser la géométrie pour faire de l'alignement de domaine," Journée Vision 3D et apprentissage, GDR ISIS, Paris, France, 2023.

[NPM3D, 2023] J. Sanchez, J.-E. Deschaud et F. Goulette, "Généralisation de domaine pour la segmentation sémantique 3D pour véhicules autonomes," Séminaire NPM3D, Issy-Les-Moulineaux, France, 2023.

Publications dans des revues avec comité de lecture

[T-RO] J. Sanchez, J.-E. Deschaud et F. Goulette, "Coarse labels : Multi-source domain generalization and pre-training for LiDAR semantic segmentation," IEEE Transactions on Robotics. **En cours de rédaction**

[PAMI] J. Sanchez, J.-E. Deschaud et F. Goulette, "3DLabelProp : A method for single-source domain generalization of semantic and moving object LiDAR segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence. **En cours de rédaction**

Chapitre 2

Segmentation sémantique LiDAR pour le véhicule autonome

Résumé

La segmentation sémantique de données LiDAR pour le véhicule autonome est un axe de recherche récent, ayant principalement émergé avec l'apparition des premières données disponibles publiquement en 2019. Elle repose sur des méthodes d'apprentissage profond. Initialement inspirées de la vision 2D, elle s'est progressivement émancipée pour répondre aux spécificités des données géométriques. Dans ce chapitre, je commencerai par introduire les méthodes utilisées pour faire de la segmentation sémantique 3D. Ensuite, les jeux de données principaux de la segmentation sémantique LIDAR, qui seront utilisés dans le reste de ce travail, seront présentés.

Sommaire

2.1 Segmentation sémantique

Comme mentionné lors de l'introduction, la segmentation sémantique a pour objectif d'associer à chaque élément atomique d'une représentation une valeur sémantique. En pratique, pour les images, on associera une valeur à chaque pixel pour les images, et à chaque point pour les nuages de points. Les valeurs sémantiques sont généralement appelées des labels ou des classes.

Historiquement, la vision artificielle s'intéressait majoritairement à distinguer le premier plan (*foreground*) de l'arrière-plan (*background*) grâce à des méthodes basées sur les graphes [4], l'agrandissement de région (*region growing*) [5], ou l'utilisation de lignes de niveau [6].

Par la suite, des méthodes plus sophistiquées ont été développées pour étudier plusieurs classes différentes à l'intérieur de la notion de *background* et de *foreground*.

Bien que ces méthodes soient intéressantes par leurs propriétés et leur rapidité, le cœur de notre travail ne s'intéresse qu'aux méthodes d'apprentissage profond. Plus précisément, je vais réaliser un rapide tour d'horizon de l'histoire de l'apprentissage profond pour la vision artificielle, puis je recentrerai le sujet sur la question de la segmentation sémantique pour la vision 2D, acquise par caméra ou appareil photo. Pour terminer, je me concentrerai plus précisément sur ces questions pour la 3D.

2.1.1 La segmentation sémantique 2D

A. L'apprentissage profond pour la vision artificielle

Le traitement automatisé de l'information visuelle a connu un très grand tremplin avec l'introduction des réseaux de neurones convolutionnels. Ceux-ci ont d'abord été introduits pour la reconnaissance d'écriture manuscrite avec LeNet [7], puis ont été appliqués à une plus grande variété de tâches comme la classification d'images [8].

En quelques mots, les réseaux convolutionnels reposent sur l'alternance de trois opérations neuronales précises qui ensemble forment une couche convolutionnelle : l'application d'un filtre de convolution, un processus de sous-échantillonnage et l'application d'une non-linéarité. Un schéma simplifié du fonctionnement d'une telle couche peut être trouvé sur la figure 2.1.

Prenons ici le temps d'introduire quelques notions et termes techniques relatifs, d'une part, aux couches convolutionnelles, et d'autre part, aux réseaux profonds dans leur ensemble.

Tout d'abord, pour les couches convolutionnelles, les notions de noyau de convolution et de champ réceptif seront utilisées tout au long de ce travail et se doivent d'être proprement définies. Le principe du noyau de convolution est facilement compréhensible à partir de la figure 2.1, et correspond à un motif utilisé par la couche convolutionnelle pour agréger des informations. Bien que sur la figure, le noyau de convolution soit carré, il peut être de forme quelconque. Dans [9], la convolution et tous ses sous-cas sont réintroduits mathématiquement et le noyau de convolution

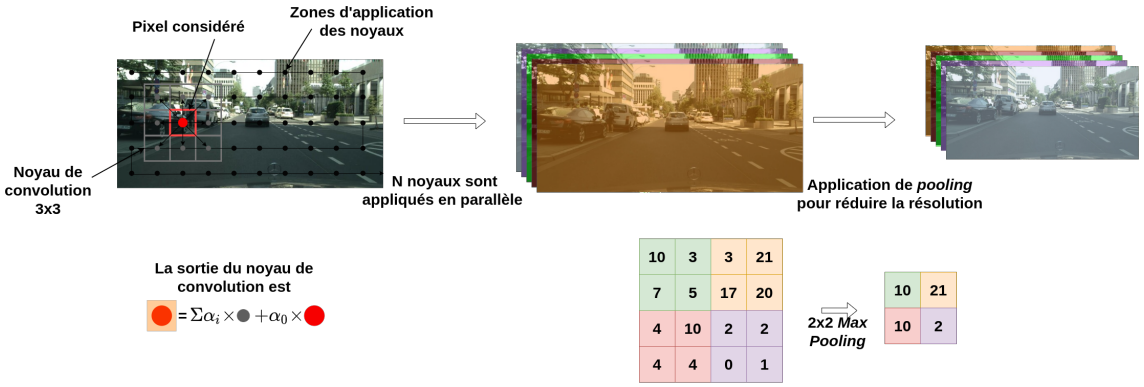


FIGURE 2.1 – Schéma simplifié d'une couche convolutionnelle.

correspond au voisinage utilisé lors de l'étape d'agrégation. Le champ réceptif est plutôt un élément d'analyse que de définition mathématique de la convolution et correspond à l'ensemble des parties de l'image couverte par une convolution donnée. Bien qu'à la première étape d'un réseau de neurones convolutionnel, le champ réceptif et le noyau de convolution se confondent, au fur et à mesure que l'on progresse dans le réseau et que l'on diminue la résolution de l'image, le champ réceptif augmente alors que le noyau de convolution reste le même. Utiliser un grand champ réceptif amène à acquérir des informations loin du point considéré au sein de l'image, alors qu'un petit champ réceptif prend en compte à des informations proches.

Pour ce qui est des réseaux profonds, ici nous présentons un élément clef dans leur analyse, qui est le nombre de paramètres. Ce dernier correspond au nombre d'éléments à optimiser quand on entraîne un réseau profond. C'est un paramètre généralement utilisé pour mesurer la complexité d'un réseau et, souvent, indirectement son temps nécessaire pour l'entraînement (en nombre d'epochs) et son temps d'inférence (en secondes ou millisecondes par image). Le terme de profondeur est souvent lié au nombre de couches à l'intérieur de l'architecture ; bien que souvent corrélé au nombre de paramètres, ce sont deux notions distinctes.

Cependant, bien que ces approches historiques aient été décisives pour le développement du domaine, c'est l'introduction d'AlexNet [10] et du premier réseau réellement profond qui marque le début de l'essor de l'apprentissage et des réseaux de neurones convolutionels dans le traitement automatisé des images. Pour visualiser la complexité d'AlexNet par rapport aux approches précédentes, une comparaison du réseau avec celui de LeNet est présentée sur la figure 2.2.

En plus de l'introduction de ces réseaux profonds, une autre amélioration significative des réseaux de neurones convolutionnels est à mentionner : l'introduction des *skip-connection*, les connexions résiduelles, par ResNet [11]. Ces connexions permettent de minimiser un effet appelé le *vanishing gradient*, qui a permis de simplifier très fortement l'entraînement des architectures très profondes. Encore une fois, cela a abouti à une forte augmentation du nombre de couches dans les réseaux de neurones,

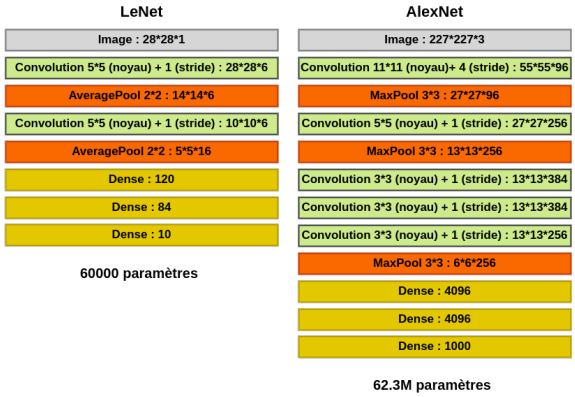


FIGURE 2.2 – Comparaison de la profondeur d'AlexNet et de LeNet. Les couches ReLU ne sont pas représentées.

ce qui est illustré dans la figure 2.3.

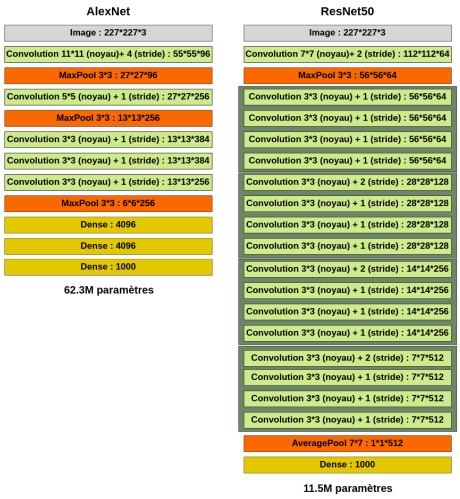


FIGURE 2.3 – Comparaison de la profondeur de ResNet et d'AlexNet.

Alors que les réseaux de neurones convolutionnels ont dominé la vision articielle pendant une décennie, depuis 2020 les *transformers* sont devenus un nouveau paradigme d'apprentissage profond pour les images. Utilisés originellement dans le traitement automatisé du langage [12], les transformers ne reposent pas sur la couche de convolution présentée précédemment. À la place, les transformers sont conçus pour traiter des séquences grâce au *self-attention* qui est la capacité à pondérer l'importance d'un élément dans une séquence pour son traitement. Pour réaliser cela, les couches transformers sont composées de trois éléments : couche de normalisation, réseau d'attention à plusieurs têtes et *multi-layer perceptron* (MLP). Une illustration explicative de cette couche peut être trouvée sur la figure 2.4. Leur application en

imagerie consiste à découper une image en (généralement) 16 morceaux, et à traiter ces morceaux comme une séquence [13].

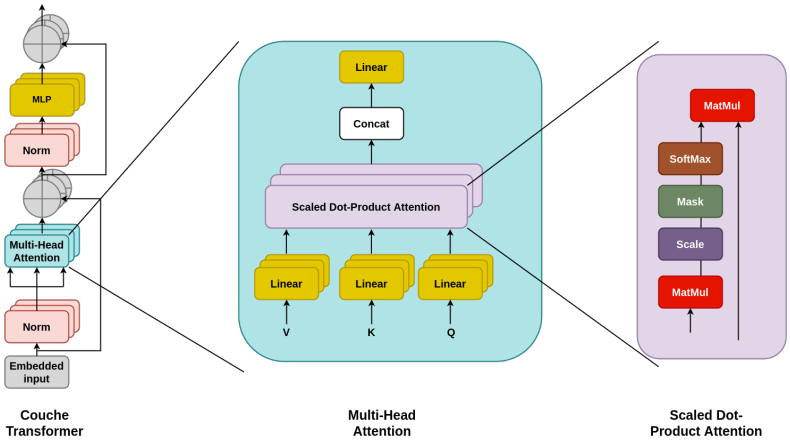


FIGURE 2.4 – Schéma explicatif d'une couche *transformer*. Inspiré par [12] et [13].

B. Segmentation sémantique

Les premiers réseaux émergeant pour le traitement d'images automatisé étaient des réseaux de classification (AlexNet [10], VGG [14]), et les premières architectures de segmentation s'en sont inspirées. La différence majeure avec les méthodes de classification est la nécessité pour la segmentation de conserver la résolution de l'image entre l'entrée et la sortie. Les premières stratégies consistaient à adopter la même architecture que pour la classification, mais en remplaçant la couche de classification directement par une couche de segmentation ayant la résolution adaptée [15]. D'autres méthodes utilisent les *feature maps* du réseau de classification qui sont extraites et fusionnées pour réaliser une prédiction de segmentation [16].

Ensuite, des méthodes plus modernes ont fait leur apparition. Elles reposent sur des couches de déconvolution [17] (parfois appelées convolution transposée), qui effectuent une opération inverse à la convolution et qui ont une résolution plus élevée pour leurs sorties que leurs entrées. Ces architectures, appelées *encoder-decoder*, sont devenues typiques des réseaux de segmentation. À partir de cette structure générale, des architectures plus précises sont nées, comme les U-Net [18]. Cette architecture rajoute des connexions entre l'*encoder* et le *decoder* et est particulièrement adaptée aux petits jeux de données. Une illustration des architectures *encoder-decoder* et des U-Net est présentée sur la figure 2.5. Par la suite, d'autres stratégies ont été employées comme les convolutions dilatées [19] qui permettent d'avoir un champ réceptif plus grand dans les convolutions pour le même nombre de paramètres, ou l'utilisation de post-traitements comme l'ajout d'un *conditional random field (CRF)* pour régulariser la sortie des réseaux [20].

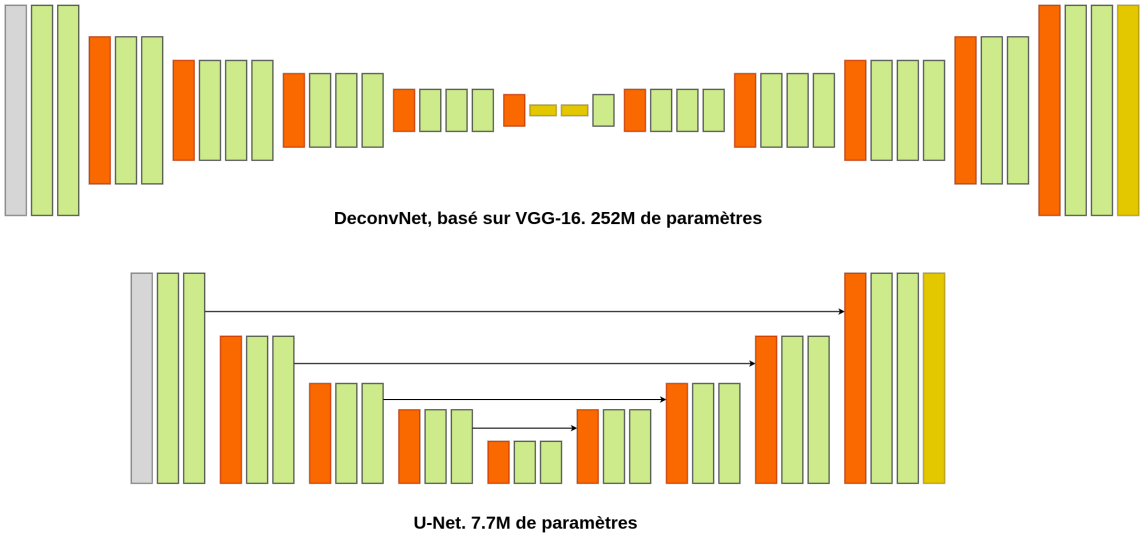


FIGURE 2.5 – Comparaison entre une architecture *encoder-decoder* et un U-Net.

La plupart des architectures tentent d'introduire une notion d'information multiéchelle en fusionnant des informations de différentes profondeurs au sein de l'*encoder*. Récemment, PSPNet [21] fait partie des méthodes obtenant les meilleurs résultats sur cette tâche. Avec son module *pyramid pooling*, il extrait des informations à différents endroits et résolutions avant de les fusionner par concaténation.

Les architectures précédemment mentionnées sont toutes basées sur les réseaux de convolution, cependant les transformers ont aussi permis d'aboutir à des architectures de segmentation sémantique. Les deux architectures de transformers de référence sont SETR [22] et SegFormer [23]. Nous ne rentrerons pas dans le détail du fonctionnement de ces architectures, car notre travail s'est concentré sur les architectures de convolution.

Ce panorama est très succinct, car il n'est pas le cœur de notre travail, mais il a permis d'introduire quelques notions clefs dans l'étude des architectures de segmentation. Plus précisément, ce sont surtout les réseaux de la famille des U-Net qui nous intéressent, car ils ont plusieurs propriétés très importantes dans le cadre du traitement de la donnée 3D : ils ont nativement peu de paramètres n'ont besoin que d'une faible quantité de données pour atteindre de bonnes performances.

2.1.2 Les spécificités de la segmentation sémantique 3D pour le véhicule autonome

Le processus typique des méthodes traditionnelles pour la perception 3D peut être divisé en trois étapes : d'abord la sélection de voisinages, puis l'extraction de caractéristiques préétablies et finalement la classification ou la *clusterization*. Beaucoup de méthodes pour préétablir les caractéristiques à extraire existent, mais

on peut notamment citer celles se basant sur la géométrie [24, 25], celles se basant sur la covariance [26] et celles se basant sur la distribution des formes [27].

Après l'apparition d'AlexNet [10], la vision 3D aussi a cherché à se renouveler et à intégrer ces nouvelles architectures profondes dans le traitement des nuages de points. Cependant, contrairement aux images traditionnelles, les nuages de points sont intrinsèquement désordonnés et épars et donc naïvement inadaptés à l'opération de convolution telle que présentée auparavant. Cela a mené à trois façons principales de traiter les nuages de points par des méthodes d'apprentissage profond :

— Voxeliser le nuage de points pour forcer une représentation ordonnée du nuage. À partir de là, on peut facilement créer des couches de convolution 3D avec des noyaux de convolution tridimensionnels. Le travail pionnier sur ce sujet est VoxNet [28]. À cause de la taille et de la nature éparse des nuages de points, ces méthodes ont longtemps été considérées comme inutilisables à cause de l'empreinte mémoire nécessaire pour stocker les voxels vides. Plusieurs méthodes ont été développées pour essayer d'échapper à ce problème, grâce à des *octree* [29], ou plus récemment grâce aux *sparse convolutions* (convolutions éparses) [30]. Bien que [30] définisse théoriquement les convolutions éparses et leurs intérêts en 3D, ce sont les travaux parallèles de SSCNet [31] et de MinkowskiNet [9] qui ont démontré l'utilité de ce type de convolution et ont démocratisé leur utilisation pour le traitement de nuages de points 3D.

— Projeter le nuage de points sur une représentation 2D et utiliser la littérature disponible sur le traitement des images 2D pour traiter le nuage. Il y a plusieurs façons de faire de la projection en 2D dans le cadre du véhicule autonome, par exemple la projection sphérique [32], la projection *bird eye view* [33] et le *scan unfolding* [34].

— Traiter directement le nuage de points grâce à des opérations indépendantes de l'ordre des points dans la représentation machine. Cela a été introduit de manière efficace par PointNet [35] et étendu avec PointNet++ [36] qui a incorporé de l'information de voisinage grâce à la couche neuronale *Set Abstraction*. À partir de là, plusieurs travaux ont cherché à définir une convolution basée sur les points, comme KPConv [37]. Une fois la convolution redéfinie, des réseaux profonds similaires à ceux 2D peuvent l'être également.

Bien que l'apprentissage profond 3D trouve ses racines dans le milieu des années 2010, la mise à disposition de jeux de données pour la segmentation sémantique 3D pour le véhicule autonome date de 2019 avec SemanticKITTI [3], et c'est à partir de ce moment que les méthodes dédiées à cette tâche ont pu se développer. Il est important de noter que la segmentation sémantique pour les nuages de points denses, comme Semantic3D [38], a généralement des résultats qui ne s'exportent pas sur les nuages de points épars typiques du véhicule autonome. De plus, la segmentation sémantique pour le véhicule autonome est soumise à de fortes contraintes sur le temps d'exécution, problématique qui n'existe pas sur les nuages denses, généralement traités en *offline*.

Comme mentionné précédemment, la segmentation sémantique 3D pour le véhicule autonome est généralement divisée en trois groupes, selon le format sous lequel le nuage de points est représenté : basée voxels, basée points et basée projection 2D. Cette représentation traditionnelle est de plus en plus remise en cause par l'apparition de méthodes de représentation par graphe, et par l'utilisation de plusieurs représentations du nuage de points en entrée. Malgré cela, nous utiliserons la classification traditionnelle pour présenter l'état de l'art.

2.1.3 Méthodes basées points

Les premières méthodes se sont basées sur le même paradigme que PointNet [35], du fait de sa faible consommation de mémoire et de sa grande vitesse de segmentation. L'idée derrière PointNet est d'appliquer des opérations indépendantes de l'ordre des points dans la représentation mémoire. Dans le cas de PointNet, cela est fait par l'application successive de multiplications matricielles, de MLP sur chaque point indépendamment et du *pooling* sur l'ensemble des points. Ainsi, PointNet est capable d'extraire des caractéristiques locales et globales qui peuvent être concaténées aux caractéristiques d'entrée pour réaliser la segmentation. L'écueil principal de cette approche est qu'elle ne modélise aucune information de voisinage pendant l'extraction des caractéristiques. Pour pallier cela, PointNet++ [36] ajoute la couche *Set Abstraction* qui extrait une caractéristique locale calculée à partir du voisinage du point considéré. Ce voisinage étant extrait grâce à une distance géométrique, il est bien indépendant de l'ordre des points. Cette couche est appliquée à différentes échelles pour extraire des voisinages de plus en plus grands et ainsi construire des caractéristiques plus pertinentes pour chaque point.

Parallèlement, d'autres méthodes se sont employées à réaliser des couches d'apprentissage similaires, en s'assurant de la modélisation de relations hiérarchiques comme Cascaded Neural Network [39] qui applique la même opération à trois résolutions différentes. PointASNL [40] incorpore une couche *Local Non Local* pour s'assurer que l'apprentissage des caractéristiques prend en compte les dépendances géométriques entre les points.

Après les résultats prometteurs présentés par PointNet++, d'autres travaux se sont intéressés à la segmentation sémantique basée points. Pour cela, un ensemble de méthodes se sont essayées à redéfinir la couche convolutionnelle typique des approches 2D, mais avec un noyau de convolution adapté à la représentation désorganisée des nuages de points. L'approche qui a su tirer son épingle du jeu est KPConv [37] qui utilise un ensemble de points pour définir le noyau de convolution, les *kernel points*. La deuxième innovation de cette méthode est l'utilisation d'une recherche de voisin par rayon. Les voisins du point considéré sont extraits grâce à cette méthode, et ensuite l'opération de convolution est réalisée sur chacun des voisins relatifs à chaque *kernel point*, d'une façon similaire à ceux de la convolution 2D grâce à un terme de corrélation linéaire. Une illustration peut être vue sur la figure 2.6.

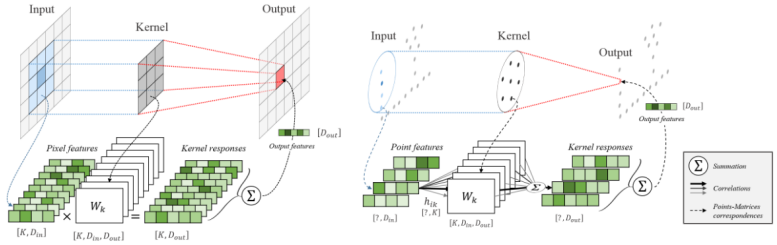


Figure 2. Comparison between an image convolution (left) and a KPConv (right) on 2D points for a simpler illustration. In the image, each pixel feature vector is multiplied by a weight matrix $(W_k)_{k<K}$ assigned by the alignment of the kernel with the image. In KPConv, input points are not aligned with kernel points, and their number can vary. Therefore, each point feature f_i is multiplied by all the kernel weight matrices, with a correlation coefficient h_{ik} depending on its relative position to kernel points.

FIGURE 2.6 – Illustration de KPConv, extrait de [37].

Cependant, d'autres méthodes définissent la convolution de manière différente. Par exemple, ShellNet [41] définit la convolution comme un ensemble de coquilles concentriques autour du point considéré regroupant les points selon leur distance euclidienne, ensuite une opération *maxpool* est appliquée dans chaque coquille indépendamment ce qui permet de finalement utiliser une convolution 1D pour extraire la caractéristique locale. SpiderCNN [42] définit la convolution grâce au produit entre une fonction escalier, qui permet d'extraire l'information géodésique locale, et un polynôme de Taylor. TangentConv [43] applique une convolution 2D dans le plan tangent au point considéré. Geo-CNN [44] extrait des caractéristiques de bord entre le point considéré et ses voisins selon les trois axes euclidiens et les agrège selon l'angle entre le segment reliant le point considéré à son voisin et les axes euclidiens. Cette liste n'est pas exhaustive, mais contient les approches ayant démontré les meilleures capacités d'extraction de caractéristiques pour les nuages de points, et présentant des paradigmes de convolution différents.

D'autres types de méthodes modélisent le nuage de points comme un graphe, généralement en créant des *neighbor-graphs*. Ces graphes sont créés en mettant des arêtes entre chaque point et ses plus proches voisins, soit avec une recherche par rayon, soit avec une approche KNN. Ces voisins peuvent être extraits dans l'espace euclidien, ou dans l'espace des caractéristiques. Parmi ces méthodes, on peut citer LSGC [45], SPG [46] et GAC [47]. Ces approches sont largement inspirées par les travaux réalisés pour l'apprentissage profond pour les graphes, mais en prenant en compte la nature géométrique des nuages de points.

Le problème principal de ces méthodes est leur lenteur, car le calcul de voisinage est une opération généralement coûteuse, qui en plus nécessite un fort sous-échantillonnage du nuage de points d'entrée. De ce fait, certains travaux se sont intéressés à ce processus de sous-échantillonnage, dans le but de l'accélérer. Alors que la plupart des travaux utilisent du *furthest point sampling (FPS)*, qui a la garantie de retirer successivement des points éloignés les uns des autres pour garantir de préserver au maximum les informations de la scène, RandLA-Net [48] utilise le *random sampling* à la place. L'intérêt de ce choix est la très grande vitesse d'exécution

de cette technique de sous échantillonage. Malgré cette accélération, les méthodes basées points qui ont les meilleurs résultats quantitatifs en segmentation sémantique sont encore trop lentes pour être véritablement considérées dans le cadre du véhicule autonome.

2.1.4 Méthodes basées projection 2D

Pour atteindre les vitesses nécessaires au traitement embarqué sur les véhicules autonomes, soit autour des 10 scans traités par seconde pour être au moins aussi rapide que la fréquence d'acquisition du capteur, les méthodes basées projection 2D ont été favorisées pendant plusieurs années. Ces méthodes ont à leur disposition toute la littérature de segmentation sémantique 2D et peuvent l'appliquer à une représentation très compacte des nuages de points, permettant d'obtenir des vitesses de traitement très élevées (jusqu'à 40Hz). Les premiers travaux sur ce sujet sont SqueezeSeg [32] et RangeNet++ [49]. Ces deux méthodes reposent sur une projection sphérique qui encode la *range value*, qui correspond à la distance de chaque point acquis au centre du capteur, les positions des points et les caractéristiques de ceux-ci dans une image 2D. Ces images 2D sont appelées les *range images* et un exemple est présenté sur la figure 2.7. Sur la figure, on observe les calculs de projections qui permettent d'associer à chaque point 3D des coordonnées 2D. Ces images ont des propriétés géométriques spécifiques, car chaque ligne de l'image correspond aux acquisitions d'un laser particulier dans le capteur d'origine.

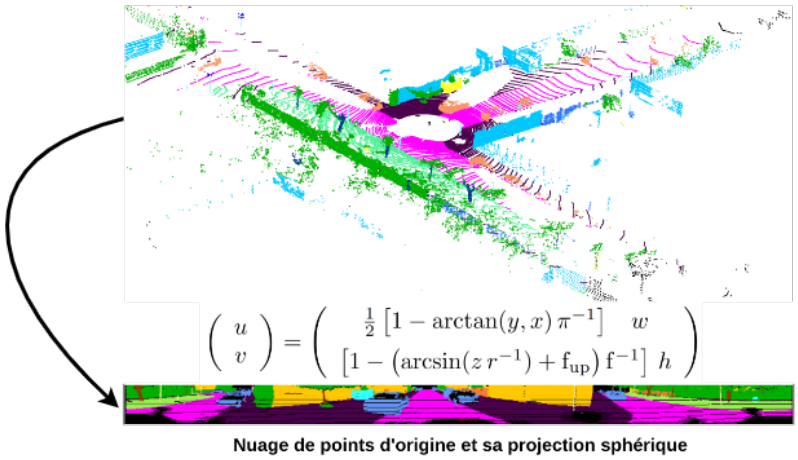


FIGURE 2.7 – Procédure de projection sphérique.

SqueezeSeg utilise une architecture de type *encoder-decoder* classique, dans laquelle est incorporée une nouvelle couche *Fire* qui applique une convolution à deux échelles différentes en parallèle. Un CRF est également appliqué en sortie de leur modèle. RangeNet++ utilise une architecture similaire à celle de DarkNet53 [50], en adaptant la forme du noyau de convolution aux propriétés spécifiques des *range*

images. De plus, la construction de la *range image* peut résulter en des phénomènes d'occlusion avec certains points encodés dans le même pixel. Pour obtenir la segmentation de ces points, un module KNN est utilisé en post-traitement.

SqueezeSeg a connu deux améliorations successives avec SqueezeSegV2 [51] et SqueezeSegV3 [52]. Elles ont, respectivement, amélioré la couche *Fire* puis l'ont remplacée avec des convolutions prenant en compte les propriétés géométriques. KPRNet [53] suit la même idée que RangeNet++, mais remplace le post-traitement par un autre utilisant KPConv. Des méthodes se sont intéressés aux propriétés de cette projection sphérique pour l'améliorer et éviter certains artefacts présents dans la *range image*, comme la méthode du Scan unfolding [54].

Ces méthodes historiques, bien que très rapides, souffraient de problèmes de plafonnements des résultats qualitatifs, ce qui en limitait l'utilisation. Plus récemment, plusieurs méthodes ont réussi à pousser les performances des architectures basées sur la projection sphérique, comme CENet [55] qui construit des fonctions de pertes auxiliaires et qui revoit la forme des convolutions. Rangeformer [56] et RangeViT [57] ont intégré des concepts venant de la littérature des transformers pour obtenir des résultats extrêmement compétitifs.

Les méthodes basées sur la projection *bird eye view* ont aussi été grandement utilisées. Cette représentation, extrêmement importante dans la détection d'objet 3D (comme dans [58]), a été utilisée par SalsaNet [59] qui encode la hauteur moyenne et maximum des points, la réflectivité moyenne et le nombre de points projetés dans chaque pixel de l'image 2D. Cette représentation des nuages et utilisée en entrée d'une architecture *encoder-decoder*, avec des briques extraites de ResNet [11]. PolarNet [60] utilise un découpage polaire plutôt qu'euclidien du plan xy, afin de se rapprocher du schéma d'acquisition des capteurs LiDAR rotatifs.

Des méthodes ont parfois combiné ces 2 types de projection, comme TORNADO-Net [61] qui utilise la projection *bird eye view* pour enrichir les caractéristiques d'entrée de la projection sphérique.

Finalement, WaffleIron [62] se positionne entre les méthodes basées points et les méthodes basées projection 2D. Dans ce travail, ils projettent les points selon les trois plans euclidiens xy, yz, xz, et appliquent des convolutions denses 2D sur ces représentations projetées. En utilisant cette triple projection, ils s'assurent d'être peu sensibles aux problèmes d'occlusion des autres approches, et ils gardent la rapidité des méthodes basées projection 2D.

2.1.5 Méthodes basées voxel

Voxeliser un nuage de points correspond à découper l'espace en sous-éléments volumétriques, pour lesquels il est facile de définir un voisinage. L'approche typique consiste à découper l'espace en cubes, comme représenté sur la figure 2.8. Une fois l'espace découpé, il y a plusieurs façons de stocker l'information voxelique : soit conserver les caractéristiques moyennes de l'ensemble des points qui se trouvent

dans ce voxel, soit conserver celles d'un seul d'entre eux choisi aléatoirement.

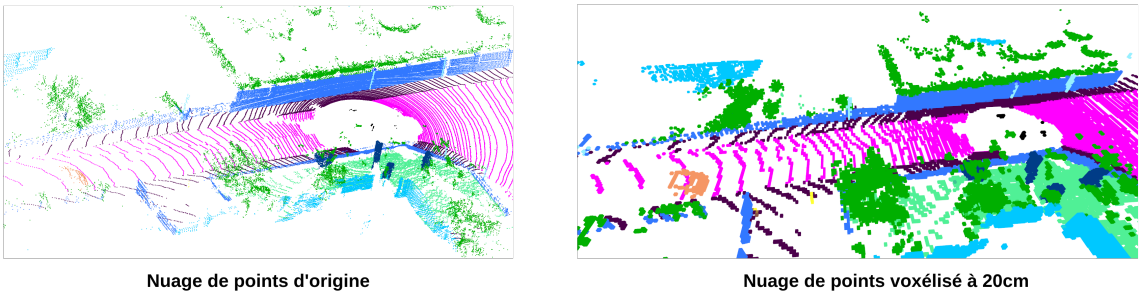


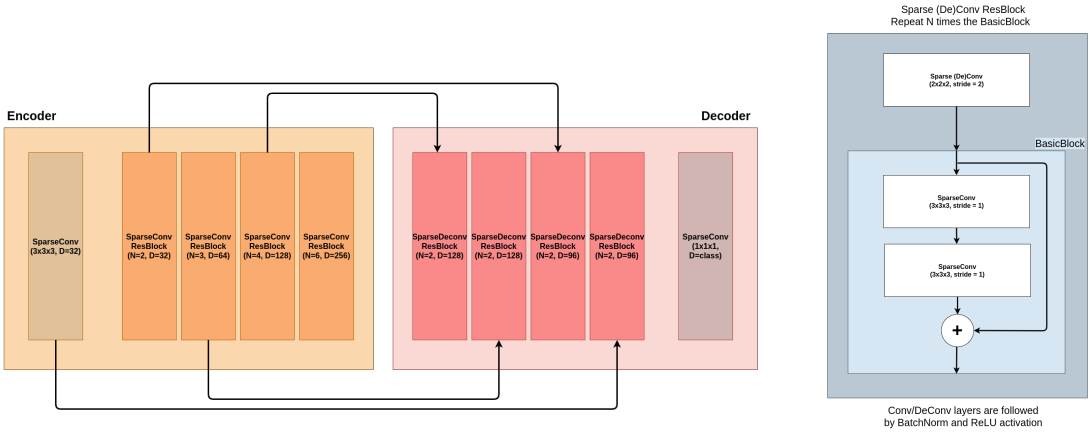
FIGURE 2.8 – Illustration d'un processus de voxelisation en cube.

Comme mentionné en introduction dans la sous-section 2.1.2, les premières méthodes basées voxel avaient un coût prohibitif. Cela résultait en des voxelisations très grossières qui provoquaient une perte d'information. Certains travaux s'y intéressaient tout de même, notamment comme procédé de sous-échantillonnage multiéchelle [63]. De plus, VoxNet [28] pose déjà les bases des architectures profondes à appliquer aux représentations voxeliques.

Cependant, c'est l'introduction des convolutions éparses qui a permis la très forte réduction des coûts mémoire et donc l'essor des méthodes basées voxels. Les convolutions éparses reposent sur deux concepts clefs qui permettent cette révolution technique : seuls les voxels non vides sont stockés en mémoire et les convolutions ne sont appliquées qu'aux voxels occupés. Cela évite de provoquer la dilatation du champ de caractéristiques entre chaque couche convolutionnelle, et réduit l'empreinte mémoire, car la boîte englobante d'un nuage de points est majoritairement vide. Le premier travail en tirant parti est SSCNet [31] qui propose un *sparse U-Net*, reproduisant l'architecture de U-Net [18] avec des convolutions éparses. Par la suite, les convolutions éparses ont été formalisées dans [9] et une architecture similaire à la précédente a été mise au point, le *sparse residual U-Net (SRU-Net)* qui combine sparse U-Net et des blocs résiduels à la façon de ResNet [11]. Une telle architecture est présentée sur la figure 2.9.

Ces deux architectures ont servi de référence et de colonne vertébrale à la plupart des autres approches basées voxel. $(AF)^2$-S3Net [64] propose notamment d'incorporer de l'attention multiéchelle pour créer des dépendances géométriques entre l'environnement et les éléments de la scène. Cylinder3D [65, 66] découpe l'espace dans un référentiel cylindrique plutôt qu'euclidien, le pendant voxélique de PolarNet. Ils introduisent aussi deux améliorations par rapport à l'approche standard : ils font fonctionner un PointNet dans chaque voxel pour créer une caractéristique locale à chaque voxel malgré le sous-échantillonnage ; ils introduisent les convolutions asymétriques qui réalisent deux convolutions non cubiques en parallèle (3x1x3 et 1x3x3) plutôt qu'une cubique (3x3x3).

En plus de stratégies architecturales, des stratégies d'entraînement ont été mises



(a) SRU-Net. (b) Conv ResBlock.

FIGURE 2.9 – Illustration d'un SRU-Net.

en place pour améliorer les performances. On peut notamment citer JS3C [67] qui apprend la *scene completion* en même temps que la segmentation sémantique. Ils supposent que les informations géométriques apprises pour réaliser la complétion sont utilisées pour la segmentation.

Pour conclure, le dernier type de méthode fusionne les informations des différents types de représentations. PVCNN [68] et SPVCNN [69] fusionnent les caractéristiques extraites par une architecture basée voxels et une autre basée points. À intervalles réguliers, ces deux architectures partagent leurs caractéristiques exploitées pour enrichir leurs représentations. De la même façon, Deep FusionNet [70] associe un PointNet basé sur des voxels et des convolutions éparses. Enfin, RPVNet [71] fusionne des informations des trois représentations mises en avant de manière adaptative à plusieurs étapes d'une architecture similaire à U-Net.

Le compromis entre vitesse d'exécution et performance que réalisent les méthodes de segmentation selon le type de données d'entrée est illustré sur la figure 2.10. Les résultats sont reportés sur SemanticKITTI (SK). On voit que les méthodes plus lentes ont tendance à obtenir de meilleurs résultats en termes de mIoU, on rappelle que la barrière du temps réel est à 10 Hz.

2.1.6 Utilisation de la temporalité des données LiDAR

Tous les travaux présentés précédemment n'utilisent qu'un seul scan LiDAR comme entrée de leurs modèles de segmentation. Cependant, les données pour le véhicule autonome ont la particularité d'être acquises séquentiellement, créant un flux vidéo de nuages de points 3D. Certains travaux ont commencé à prendre cette modalité en compte et à incorporer des informations de scans consécutifs en entrée.

ASAP-Net [72] propose de séparer les interactions spatiales et temporelles en

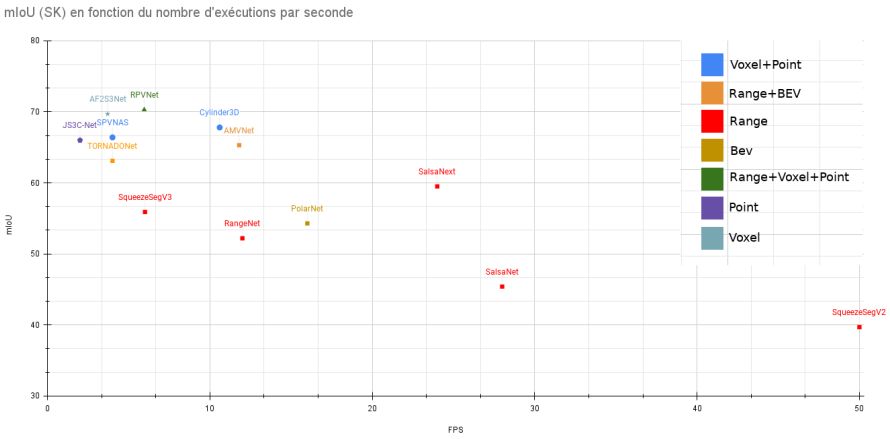


FIGURE 2.10 – Comparaison entre les performances quantitatives et le temps d'exécution d'une partie de l'état de l'art selon le type de données d'entrée.

proposant d'expliciter une corrélation spatio-temporelle entre des scans consécutifs grâce à un mécanisme d'attention temporel. SpSequenceNet [73] concatène des cartes de caractéristiques des scans passés pour enrichir les descriptions du scan courant. [74] intègre un RNN pour garder en mémoire les caractéristiques des scans passés. Helix4D [75] exploite le schéma d'acquisition des LiDARs rotatifs pour proposer une représentation cylindrique de la vidéo d'acquisition et traite des tranches d'acquisitions plutôt que les scans entiers. Nous rentrerons plus dans le détail des intuitions derrière ces méthodes dans la sous-section 4.3.3.

La segmentation sémantique 3D pour le véhicule autonome a atteint des performances quantitatives de très hautes qualités dès 2022, et la recherche autour de ce sujet a commencé sur des sujets parallèles à cette question. Ces nouveaux centres d'intérêt couvrent notamment la segmentation panoptique, la *Moving Object Segmentation (MOS)* mentionnée dans la sous-section 6.1.1, les questions relatives au préentraînement des architectures 3D couvert dans la sous-section 5.3.1, la généralisation et l'adaptation de domaine discuté dans la sous-section 4.1.3 et la sous-section 4.1.4. Dans chacune de ces sections, nous prendrons le temps de discuter des enjeux liés à ces questions et les travaux phares et récents à leur sujet.

2.2 Jeux de données existants

Les jeux de données pour la segmentation sémantique 3D se sont multipliés ces dernières années, donnant accès à de plus nombreux exemples pour les chercheurs et les industriels. Pour le cas particulier de la segmentation de données LiDAR pour le véhicule autonome, il faut distinguer les données synthétiques générées initialement avec GTA V [32] puis avec CARLA [76, 77], et les données réelles. Dans ce

travail, uniquement des données réelles ont été utilisées, et plus précisément SemanticKITTI [3], KITTI-360 [78], nuScenes [79], Waymo [80], SemanticPOSS [81] et Pandaset [82]. Ces jeux de données peuvent être à vocation soit d'entraînement (SemanticKITTI, KITTI-360, nuScenes), soit de test, et sont alors appelés les jeux cibles (SemanticPOSS, Pandaset, Waymo).

2.2.1 SemanticKITTI

SemanticKITTI [83] est le jeu de données historique pour la segmentation sémantique de donnée LiDAR pour le véhicule autonome. Il est basé sur KITTI [83] et a été publié en 2019. La plateforme d'acquisition est multicapteurs, avec un système caméra-LiDAR-IMU, cependant nous nous intéressons seulement aux données 3D acquises par le LiDAR. Un schéma minimaliste de la plateforme est présenté sur la figure 2.11. Cette configuration utilisant un LiDAR rotatif à 360° posé sur le toit de la voiture sera retrouvé dans chaque jeu de données.

Le LiDAR employé est un Velodyne HDL-64E [1], un LiDAR rotatif à 64 fibres, avec une portée d'un peu plus de 50m. En raison de l'éparsité de la scène au-delà de 50m, les auteurs ne proposent une annotation des scans que jusqu'à cette distance, par exemple la figure 2.12.

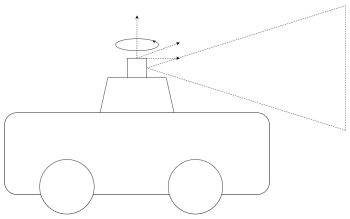


FIGURE 2.11 – Schéma d'une plateforme classique d'acquisition LiDAR pour le véhicule autonome.

Ce jeu de données a été acquis à Karlsruhe en Allemagne, en zone périurbaine. Ce choix de scène résulte en une faible quantité d'usagers de la route (piétons, véhicules) et une grande quantité de bâtiments. Un exemple de trajectoire d'acquisition est présenté sur la figure 2.13. Il s'agit de celle rendue disponible par les auteurs du jeu de données. Cependant, bien que la trajectoire globale soit précise, le recalage local peut être amélioré. Pour cette raison, les trajectoires ont été recalculées au sein du laboratoire, et ces trajectoires localement plus précises seront celles employées par la suite. Elles sont obtenues grâce à l'exécution de CT-ICP [84]. Un exemple de mise dans le même référentiel d'une partie de la séquence 00 est présenté sur la figure 2.14. Cet algorithme sera aussi utilisé pour obtenir les trajectoires suivies dans les autres jeux de données.

1. Data-sheet du HDL-64E

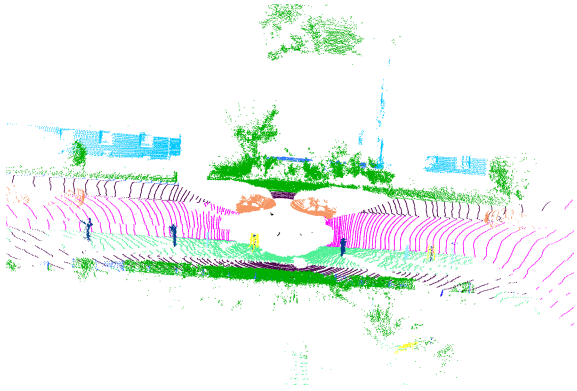


FIGURE 2.12 – Scan extrait de SemanticKITTI, colorisé selon les labels sémantiques.

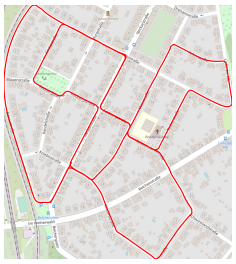


FIGURE 2.13 – Visualisation GPS de la séquence 00 de SemanticKITTI.

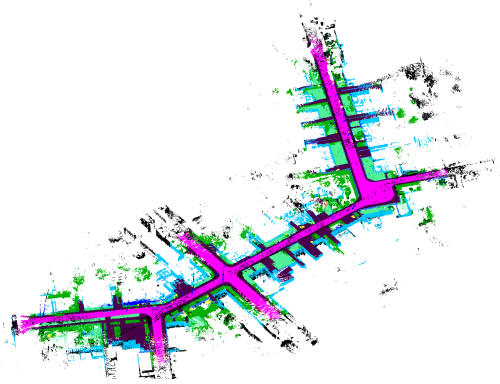


FIGURE 2.14 – Extrait de la séquence 00 de SemanticKITTI mise dans un référentiel global.

Vingt-et-une séquences ont été ainsi acquises, et sont divisées en trois sous-parties : entraînement, validation et test. La partie dédiée à l'entraînement correspond aux séquences de 00 à 10, hormis la séquence 08 qui est la séquence dédiée à la validation. Les séquences 11 à 20 sont utilisées pour le *test set*. Les annotations ne sont pas disponibles, ces dernières seront donc exclues dans la suite de l'analyse de ce jeu de données.

Chaque séquence a une durée d'acquisition différente, les détails du temps d'acquisition et du nombre de scans résultants étant rassemblés dans le tableau 2.1. Quelques autres informations sont aussi présentes dans le tableau pour illustrer la diversité des séquences.

n°	durée (s)	distance (m)	# scans	# véhicules	# piétons	# cyclistes
00	57.6	3719	4541	576	25	113
01	110.1	2456	1101	318	0	1
02	466.1	5070	4661	320	14	41
03	80.1	562	801	42	2	3
04	27.1	393	271	38	3	0
05	276.1	2205	2761	206	18	20
06	110.1	1234	1101	133	8	23
07	110.1	694	1101	187	24	24
08	407.1	3220	4071	462	62	124
09	159.1	1703	1591	173	28	11
10	120.1	918	1201	81	17	9

TABLE 2.1 – Détails des séquences de SemanticKITTI.

Les annotations choisies correspondent à 27 labels différents, réduits en pratique à 19 pour l'évaluation. Cette réduction résulte de la fusion des labels correspondants à des véhicules statiques et à leurs équivalents dynamiques. La liste des labels, ainsi que leur distribution est présentée sur la figure 2.15. On observe un schéma classique de tous les jeux de données utilisés, il existe un large déséquilibre entre les labels les plus représentés (route, construction, végétation) et les moins représentés (piétons, cyclistes). De plus, on observe un fort déséquilibre dans la représentation de chaque classe selon les séquences, très spécifiquement *motorcyclist* et *bicyclist* qui sont majoritairement contenus dans la séquence 08, alors qu'elle ne fait pas partie de l'entraînement. En outre, SemanticKITTI propose des annotations pour les tâches de *moving object segmentation* (MOS) et de segmentation panoptique, des détails vis-à-vis de ces annotations étant présentés dans le tableau 2.2.

# véhicules	# véhicules en mouvement	# piétons	# piétons en mouvement	# cyclistes	# cyclistes en mouvement
2044	492	99	102	273	96

TABLE 2.2 – Détails sur les annotations de MOS et de segmentation panoptique de SemanticKITTI.

Bien que ce jeu de données présente certains défauts, notamment dans sa diversité de scènes et sa répartition des labels, il s'agit du jeu de référence dans le domaine. Il sera donc le plus utilisé pour les expériences durant le reste de ce travail, notamment en tant que jeu d'entraînement.

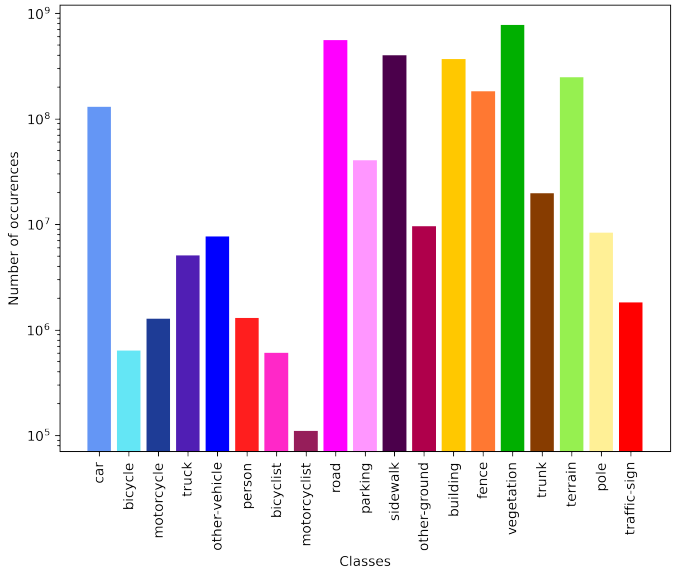


FIGURE 2.15 – Distribution des labels dans SemanticKITTI.

2.2.2 KITTI-360

KITTI-360 [78] peut être vu comme une extension de SemanticKITTI. Sorti en 2020, il a été acquis dans un périmètre élargi autour de Karlsruhe en Allemagne, avec le même système d'acquisition que SemanticKITTI. Un exemple des scans acquis est présenté sur la figure 2.16, et comme précédemment un exemple de trajectoire et de nuage de points accumulés sont présentés sur la figure 2.17 et la figure 2.18.

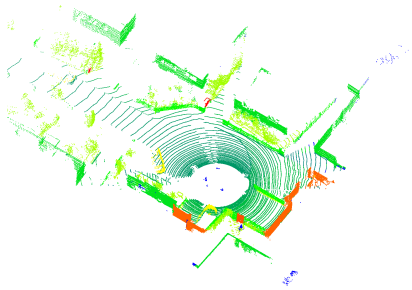


FIGURE 2.16 – Scan extrait de KITTI-360, colorisé selon les labels sémantiques.

Ce jeu de données est découpé en 9 séquences. Au début de cette thèse, il n'existait pas de séparation formelle entre entraînement et validation, nous utilisons donc les séquences 03 et 05 pour la validation, le reste pour l'entraînement. Les détails quantitatifs sur les durées et longueurs des séquences peuvent être trouvés dans le tableau 2.3.

FIGURE 2.17 – Visualisation GPS de la séquence 00 de KITTI-360.

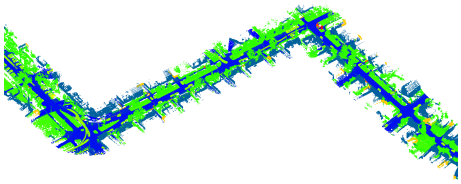


FIGURE 2.18 – Extrait de la séquence 00 de KITTI-360 placée dans un référentiel global.

n°	durée (s)	distance (m)	# scans
00	1048	8378	10483
02	1089	8712	10889
03	99	1342	988
04	814	7477	8137
05	613	4567	6131
06	906	3230	6131
07	266	4756	2664
09	1316	10444	13164
10	279	3230	2789

TABLE 2.3 – Détails des séquences de KITTI-360.

Pour toutes ces séquences, les annotations sont disponibles et sont divisées entre objet statique et objet dynamique. Une particularité de ce jeu de données est que les annotations ne sont pas disponibles pour chaque scan individuellement, mais sont données sur les nuages de points accumulés. Pour retrouver les annotations par scan, j'ai développé un script Python [2] qui repositionne les scans individuels dans les nuages accumulés et fait une recherche par plus proche voisin dans ce nuage accumulé pour en déduire l'annotation.

Les annotations sont réduites à 19 labels, mais qui sont différents de SemanticKITTI. On peut retrouver la liste des labels et leur distribution sur la figure 2.19. Ce jeu de données est le plus grand parmi ceux utilisés dans ce travail, et servira

2. `https://github.com/JulesSanchez/recoverKITTI360label`

notamment de base lorsque l'on souhaitera émuler l'accès à une large base de données. Sa large superposition avec SemanticKITTI fait que lorsque l'un est le jeu de données test, on ne peut pas utiliser l'autre pour l'entraînement, pour éviter le surapprentissage.

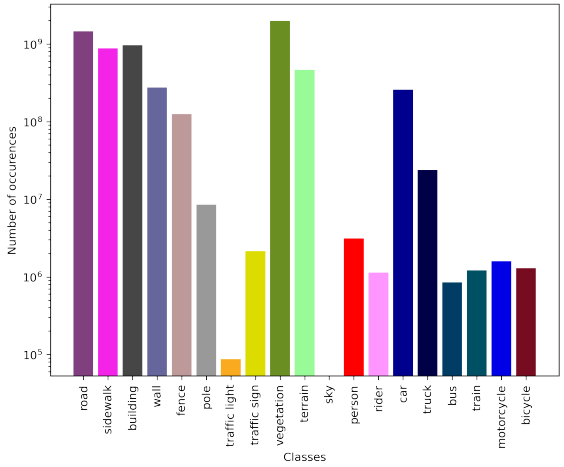


FIGURE 2.19 – Distribution des labels dans KITTI-360.

2.2.3 nuScenes

nuScenes [79] est, avec SemanticKITTI, le second jeu de données de référence pour la segmentation sémantique. Il est sorti en 2020 et a été acquis avec un LiDAR HDL32E [3], LiDAR rotatif à 32 fibres, fixé sur le toit du véhicule. La portée couvre une zone allant jusqu'à près de 100m et la totalité de l'acquisition est annotée. Un exemple de scan de nuScenes est présenté sur la figure 2.20.

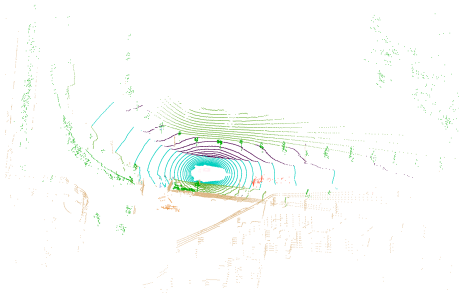


FIGURE 2.20 – Scan extrait de nuScenes, colorisé selon les labels sémantiques.

Les données proviennent de deux grandes villes : Singapour et Boston. Dans chacune de ces villes, deux quartiers ont été explorés à chaque fois. Ce sont des centres

3. Datasheet du HDL32E

urbains très denses avec une grande quantité d'usagers des voiries en comparaison avec les données précédentes acquises en zones périurbaines. Une des quatre trajectoires d'acquisition peut être observée sur la figure 2.21. Ces trajectoires sont ensuite sous-divisées en 1000 scènes de 20 secondes chacune. Un exemple d'une scène accumulée peut être retrouvé sur la figure 2.22. Les scènes sont acquises à 20Hz, mais ne sont annotées pour la sémantique qu'à 2Hz.

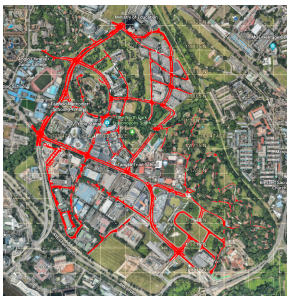


FIGURE 2.21 – Visualisation GPS de l'acquisition *Boston Seaport* dans nuScenes.

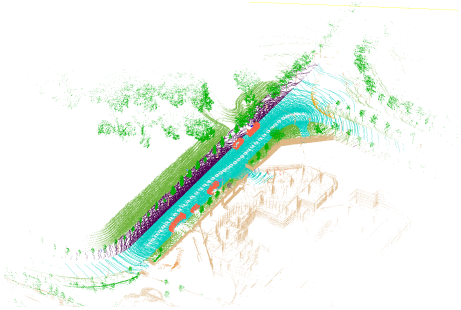


FIGURE 2.22 – La séquence 0001 de nuScenes mise dans un référentiel global.

Ces scènes sont divisées en trois sous-parties : entraînement, validation et test. Ici, nous nous intéressons uniquement aux scènes d'entraînement et de validation qui sont celles pour lesquelles les annotations sont disponibles. Il y en a 700 pour l'entraînement et 150 pour la validation. Les informations agrégées et moyennées pour ces séquences peuvent être retrouvées dans le tableau 2.4. Les statistiques par séquence ne sont pas montrées ici en raison leur grand nombre et leur uniformité dans la durée d'acquisition.

durée (s)	distance (m)	# scans	# véhicules	# piétons	# cyclistes
20	100	39	39	13	2

TABLE 2.4 – Information moyenne pour les séquences de nuScenes.

Les scans sont annotés suivant 32 labels, qui sont réduits à 16 en pratique pour

l'évaluation de la qualité de la segmentation sur ce jeu de données. La liste de ces labels et leur distribution peut être retrouvée dans la figure 2.23.

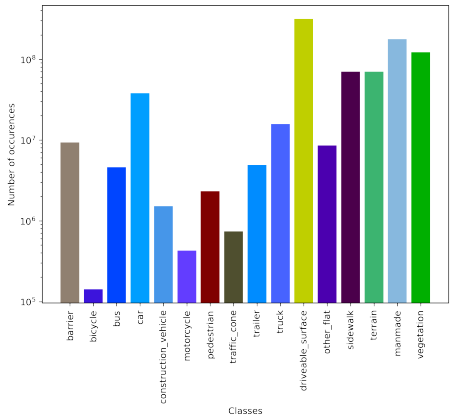


FIGURE 2.23 – Distribution des labels dans nuScenes.

Étant initialement un jeu de données de détection et de *tracking*, nuScenes propose des informations de segmentation panoptique et de MOS ; les détails de ces informations peuvent être retrouvés dans le tableau 2.5.

# véhicules	# véhicules en mouvement	# piétons	# piétons en mouvement	# cyclistes	# cyclistes en mouvement
26891	8499	3730	8143	1141	359

TABLE 2.5 – Détails sur les annotations de MOS et de segmentation panoptique de nuScenes.

2.2.4 Waymo

Waymo Open Dataset[80] est un jeu de données de détection et de *motion prediction*. Cependant, en 2022 ont été ajoutées des annotations de segmentation sémantique 3D. Cinq capteurs LiDAR sont positionnés sur le véhicule, mais seul celui posé pour le toit nous intéresse, car c'est le seul pour lequel des annotations sont rendues publiques. Il s'agit un LiDAR rotatif à 64 fibres ; le modèle du capteur n'est pas rendu public, les spécifications techniques se trouvent dans la publication associée au jeu de données [80]. Les données sont sauvegardées sous le format de *range image*, ce qui fait que le nuage de points récupéré est légèrement différent de celui acquis. En plus du prétraitement, tous les points au-delà de 75m sont retirés. Sur la figure 2.24 se trouve un exemple de scan.

Ce jeu de données a été acquis aux États-Unis, à travers six villes : Phoenix, Seattle, Detroit, Los Angeles, Mountain View et San Francisco. Ces acquisitions

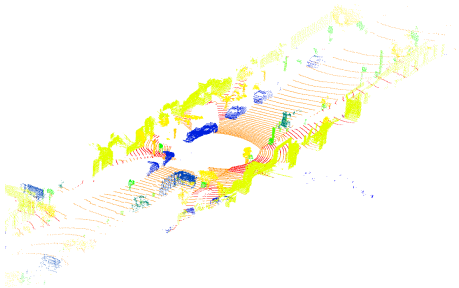


FIGURE 2.24 – Scan extrait de Waymo, colorisé selon les labels sémantiques.

sont divisées en 1000 séquences de 20 secondes chacune. Bien que l'acquisition soit à 10Hz, l'annotation sémantique est à 2Hz, la même fréquence d'annotation que nuScenes. Les données GPS ne sont pas rendues publiques, cependant on peut observer un nuage accumulé sur la figure 2.25. Les informations agrégées et moyennées des séquences peuvent être retrouvées dans le tableau 2.6.

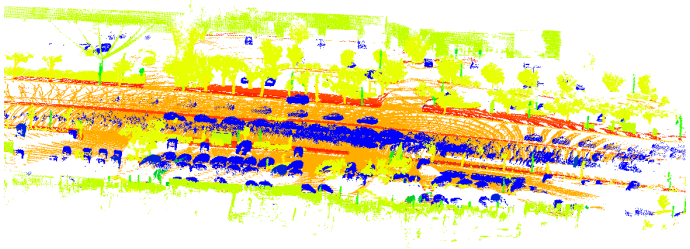


FIGURE 2.25 – La séquence 00 de Waymo placée dans un référentiel global.

durée (s)	distance (m)	# scans	# véhicules	# piétons	# cyclistes
20	103,3	29	60	23	1

TABLE 2.6 – Information moyenne pour les séquences de Waymo.

Les scans sont annotés suivant 23 labels. La liste de ces labels et leur distribution peut être retrouvée dans la figure 2.26. Bien que ce jeu de données soit conséquent, il est peu utilisé dans la littérature, car il est récent. Il sera principalement utilisé comme jeu de données cible.

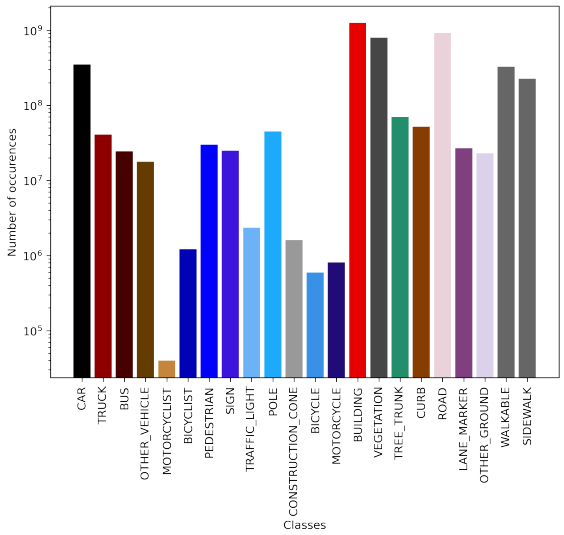


FIGURE 2.26 – Distribution des labels dans Waymo.

2.2.5 SemanticPOSS

SemanticPOSS [81] est un petit jeu de données publié en 2020. Il a été acquis avec un système Pandora [4] multicapteur, qui contient notamment un LiDAR rotatif à 40 fibres fixé sur le toit. Un exemple de scan peut être trouvé sur la figure 2.27.

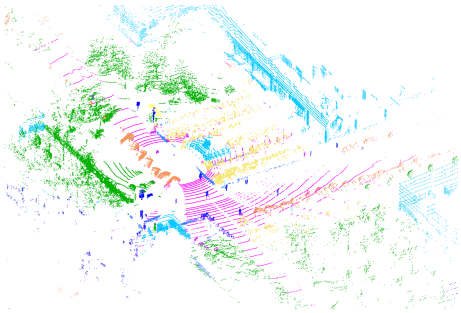


FIGURE 2.27 – Scan extrait de SemanticPOSS, colorisé selon les labels sémantiques.

La particularité de ce jeu de données est le contexte d'acquisition. Il a été acquis sur un campus universitaire, celui de l'université de Pékin, et de ce fait une grande quantité de piétons et de cyclistes peuvent être observés dans les scènes. Le détail de la trajectoire d'acquisition peut être retrouvé sur la figure 2.28. Cette séquence d'acquisition est divisée en 7 sous-séquences, et pour 6 d'entre elles les annotations sont disponibles. La séquence 00 est visionnée sur la figure 2.29. Le détail des séquences peut être retrouvé dans le tableau 2.7.

4. Datasheet du Pandora

FIGURE 2.28 – Visualisation GPS des séquences de SemanticPOSS. Illustration tirée de [81].

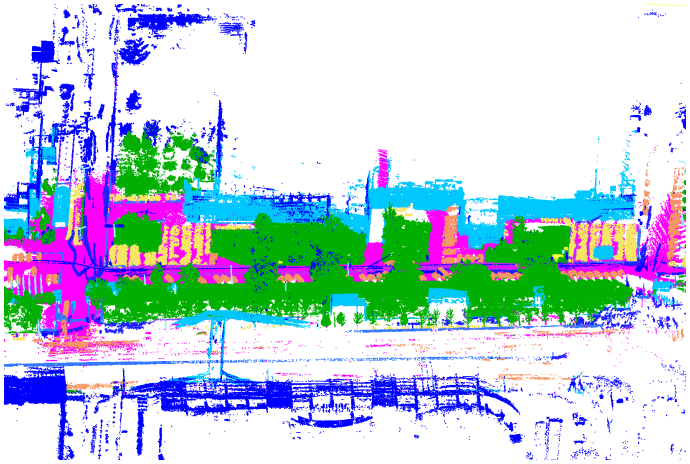


FIGURE 2.29 – La séquence 00 de SemanticPOSS placée dans un référentiel global.

n°	durée (s)	distance (m)	# scans
00	50	220	488
01	50	241	500
02	50	197	500
03	50	248	500
04	50	267	500
05	50	259	500

TABLE 2.7 – Détails des séquences de SemantiquePOSS.

Les scans sont annotés suivant 13 labels. La liste de ces labels et leur distribution est présentée sur la figure 2.30. Ce jeu de données ne donne pas d'information sur les instances. En raison de sa petite taille, ce jeu de données est souvent utilisé en tant que jeu de données cible pour une évaluation, ou pour réaliser un entraînement avec (très) peu de données disponibles comme nous le verrons dans le chapitre 5.

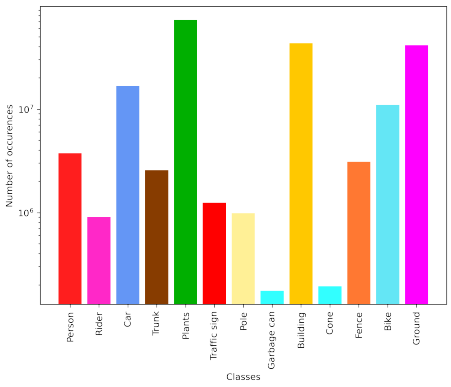


FIGURE 2.30 – Distribution des labels dans SemanticPOSS.

2.2.6 Pandaset

Pandaset[82] est un jeu de données publié en 2020 qui a la particularité d'être acquis avec deux capteurs LiDAR différents simultanément. Les capteurs utilisés sont un Pandar64 [5], LiDAR rotatif avec 64 fibres, et un PandarGT [6], LiDAR *solid-state* équivalent à 150 fibres. Un exemple comprenant un scan acquis par chaque LiDAR peut être trouvé sur la figure 2.31. Nous divisons ainsi ce jeu de données en deux parties, Panda64 qui est le jeu de données limité aux données acquises par le Pandar64 et PandaFF qui est limité aux données acquises par le PandarGT. Ces deux jeux de données ont les mêmes caractéristiques d'acquisition, car ils sont synchronisés temporellement avec une position physique très proche sur le véhicule ; de ce fait, on peut considérer qu'ils scannent les mêmes scènes.

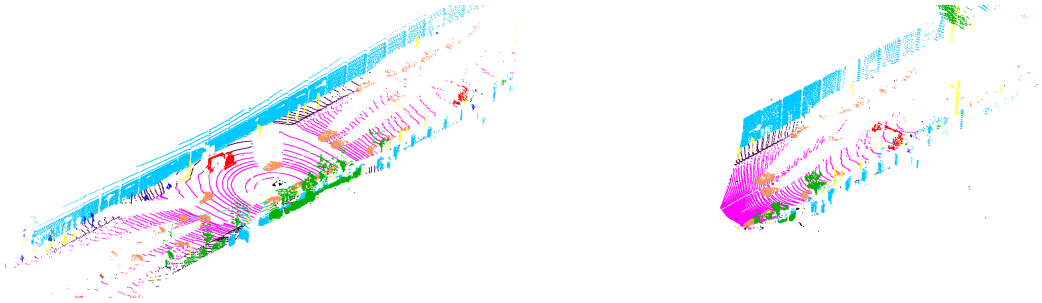


FIGURE 2.31 – Scans extraits de Panda64 (à gauche) et de PandaFF (à droite), colorisés selon les labels sémantiques.

Ce jeu de données a été acquis aux États-Unis, à San Francisco et sur la El Camino Real, ce qui en fait un jeu de données associant des informations urbaines et

5. Datasheet du Pandar64
6. Datasheet du PandarGT

périurbaines. En tout, une centaine de scènes de 8 secondes ont été acquises, mais seulement 76 sont pourvues d'annotations disponibles publiquement, et sont donc celles qui nous intéressent. Quelques-unes des trajectoires d'acquisition peuvent être retrouvées sur la figure 2.32. De plus, la séquence 001 peut être visualisée intégralement sur la figure 2.33. Les informations agrégées et moyennées des séquences peuvent être retrouvées dans le tableau 2.8.

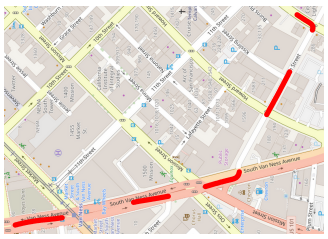


FIGURE 2.32 – Visualisation GPS des séquences 050 à 054 de Pandaset.

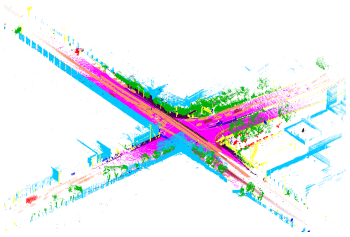


FIGURE 2.33 – La séquence 001 de Panda64 mise dans un référentiel global.

durée (s)	distance (m)	# scans	# véhicules	# piétons	# cyclistes
8	82,3	80	144	26	4

TABLE 2.8 – Information moyenne pour les séquences de Pandaset.

Les scans sont annotés suivant 37 labels. La liste de ces labels et leur distribution est présentée sur la figure 2.34. De plus, ce jeu de données est annoté pour la détection, et propose donc des informations de MOS et de segmentation panoptique ; des détails sur ces informations peuvent être retrouvés dans le tableau 2.9. L'importante quantité de véhicules identifiée peut surprendre mais elle vient en réalité de l'annotation. D'une séquence à l'autre, la numérotation des véhicules est est réinitialisée. Ainsi, le grand nombre de séquences implique un grand nombres de véhicules distincts.

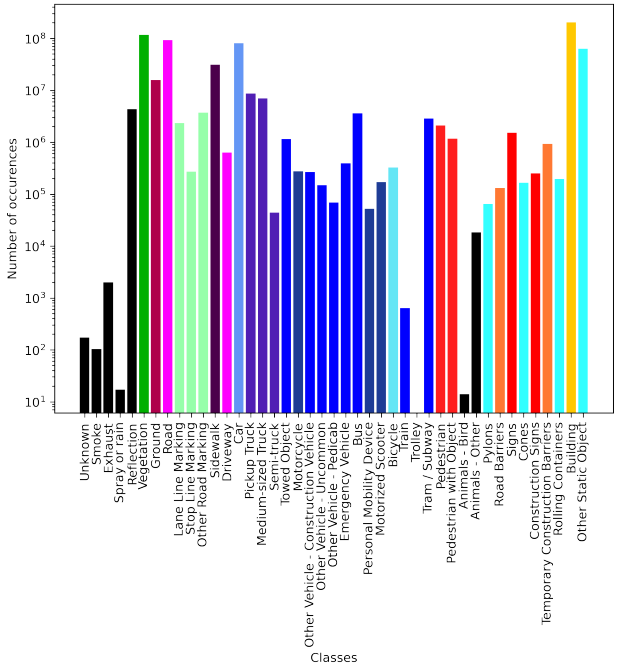


FIGURE 2.34 – Distribution des labels dans Panda64.

# véhicules	# véhicules en mouvement	# piétons	# piétons en mouvement	# cyclistes	# cyclistes en mouvement
7994	2913	488	1512	225	80

TABLE 2.9 – Détails sur les annotations de MOS et de segmentation panoptique de Pandaset.

2.2.7 Récapitulatif

De nombreux jeux de données ont été identifiés dans ce travail pour couvrir un maximum de domaines différents. Les informations principales de chaque jeu de données peuvent être retrouvées dans le tableau 2.10. Il est important de noter que chaque jeu de données présente un jeu de labels unique. Cette sur-diversité de jeux de labels occasionne des complications lors de l'évaluation multidomaine, lorsque les jeux de données d'entraînement et d'évaluation ne sont pas les mêmes.

Nom	# scans	# séquences	# labels	scène	pays	capteur
SemanticKITTI [3]	23000	11	19	périurbaine	Allemagne	HDL-64E
KITTI-360 [78]	67626	9	19	périurbaine	Allemagne	HDL-64E
nuScenes [79]	35000	850	16	urbaine	Singapour Etats-Unis	HDL-32E
Waymo [80]	30000	1150	23	périurbaine urbaine	États-Unis	N/C 64 fibres
SemanticPOSS [81]	3000	6	13	campus	Chine	Pandora
Pandaset [82]	6000	76	37	périurbaine urbaine	États-Unis	Pandar64 PandarGT

TABLE 2.10 – Résumé des jeux de données.

Chapitre 3

Evaluation multidomaine : écueils & solutions

Résumé

Lorsque le domaine de travail change entre l'entraînement et l'évaluation d'une méthode de segmentation sémantique, certaines difficultés se posent. Les jeux de données étant annotés différemment, on ne peut pas appliquer les méthodes habituelles de mesure de performance telles quelles. Comme soulevé lors de l'introduction, l'évaluation multidomaine est importante pour mesurer les performances de la généralisation de domaine. Ce chapitre a vocation à illustrer les difficultés rencontrées pour réaliser de l'évaluation multidomaine. Pour cela, dans un premier temps, j'introduirai clairement la notion de domaine et de multidomaine, et illustrerai des situations dans lesquelles l'évaluation multidomaine s'impose et comment des difficultés émergent. Dans un second temps, le jeu de données ParisLuco3D, ma première contribution majeure, sera proposé comme première piste de réponse pour aboutir à une évaluation multidomaine plus simple garantissant de bien mesurer les résultats d'une méthode de segmentation.

Sommaire

3.1 Difficultés d'évaluation multidomaine

3.1.1 Les notions de domaine et de multidomaine

Familièrement, un domaine est l'espace dans lequel les données évoluent. C'est-à-dire, l'ensemble des valeurs que peut prendre une donnée. Plus formellement, on peut définir le domaine $\mathcal{D}$, de support $X \times Y$, avec X l'espace des paramètres et Y l'espace des labels (aussi appelé jeu de labels), comme étant la distribution de laquelle les données observées ont été tirées. En pratique, on distingue deux domaines différents pour l'entraînement et l'évaluation d'algorithmes d'apprentissage machine : $\mathcal{D}_{entraînement}$ et $\mathcal{D}_{test}$, et on souhaite mesurer les performances sur $\mathcal{D}_{test}$.

Dans le cas particulier des données LiDAR pour le véhicule autonome, un domaine va être caractérisé par le contenu des scènes scannées. Les éléments clefs qui vont caractériser le domaine sont des éléments liés aux spécifications techniques du capteur : technologie, positionnement, champ de vue, résolution angulaire, nombre de fibres ; liés aux spécifications de la plateforme d'acquisition : taille, vitesse de déplacement, chemins empruntés ; liés à la météo ; liés aux éléments de la scène : types d'usagers de la route, forme de la chaussée, sortes de végétations, nombre d'éléments ; et liés aux choix d'annotations. Indirectement, les éléments de scène peuvent être rattachés au type d'urbanisme (urbain, périurbain, routier, autoroutier, campus, etc.) et à la localisation géographique.

Bien que l'on puisse supposer qu'il existe un domaine $\mathcal{D}_{universel}$ depuis lequel la totalité des scènes pour le véhicule autonome est tirée, chaque jeu de données étant restreint géographiquement et temporellement, on peut supposer qu'ils ne permettent de couvrir qu'une sous-partie de ce domaine $\mathcal{D}_{jeu}$.

La notion de multidomaine est le regroupement de plusieurs domaines

$\mathcal{D}_{multidomaine} = \bigcup_i \mathcal{D}_{jeu_i}$ dans le but de s'approcher de la représentation du domaine $\mathcal{D}_{universel}$.

En pratique, le terme de multidomaine est utilisé lorsque plusieurs jeux de données différents sont concaténés pour former un nouveau jeu de données, plus varié. Cela peut cependant présenter des difficultés et la concaténation naïve n'est pas forcément une option, surtout si $X_{jeu_1} \times Y_{jeu_1}$ est différent de $X_{jeu_2} \times Y_{jeu_2}$.

On définit le cas particulier de l'évaluation multidomaine comme le cas où soit

$X_{entraînement}$ et X_{test} sont différents, soit $Y_{entraînement}$ et Y_{test} sont différents, ou les deux. La suite de cette section traitera des deux derniers cas, lorsque les jeux de labels sont différents. Visuellement, la figure 1.4 illustre les notions de monodomaine et multidomaine.

3.1.2 Pallier les différences de jeux de labels

Au vu de la diversité des scènes et des tailles respectives des jeux de données (voir le tableau 2.10), il est souvent souhaitable d'utiliser simultanément une variété de ces jeux de données pour évaluer un modèle. Ces évaluations multidomaines

permettent une évaluation plus exhaustive des méthodes, soit pour améliorer la pertinence de la mesure des performances en augmentant le nombre de séquences regardées, soit pour mesurer leur robustesse si cette variété de jeux de données est mise en place uniquement pour l'évaluation ; ces éléments seront détaillés dans la sous-section 4.1.2.

Cependant, en plus de la diversité des situations, le tableau 2.10 met aussi en évidence un élément bloquant pour l'évaluation des méthodes de segmentation. Chacun des jeux de données propose son propre jeu de label, une illustration des jeux de label de SemanticKITTI et nuScenes étant présentée sur la figure 3.1. Rappelons les outils de mesure classique de la segmentation sémantique pour mettre en avant en quoi cela est un problème.

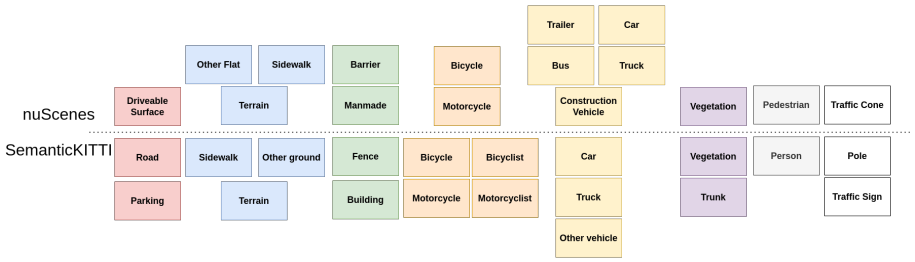


FIGURE 3.1 – Points communs et différences entre les jeux de labels de SemanticKITTI et nuScenes.

L'outil de mesure standard est l'*intersection over union (IoU)* défini comme suit dans le cas binaire, en introduisant TP (*True Positive*), FP (*False Positive*) et FN (*False Negative*) :

$$IoU = \frac{target \cap prediction}{target \cup prediction} = \frac{TP}{TP + FP + FN} \tag{3.1}$$

En pratique, dans le cadre multiclasse, cette mesure est appliquée sur chaque label indépendamment et est généralement moyennée et résulte en une mesure, le *mean intersection over union (mIoU)*. Cette métrique est alors utilisée pour comparer les méthodes. Bien que cette mesure ne pose aucun problème dans le cadre monodomaine, lorsque $Y_{entraînement}$ et Y_{test} sont différents cela amène des complications.

En effet, en pratique, une méthode de segmentation sémantique a la possibilité de prédire $N_{architecture}$ différents labels, cette valeur étant encodée en dur dans l'architecture au moment de l'entraînement. En pratique, $Y_{entraînement} = \{0, N_{entraînement}\}$ et donc $N_{architecture} = N_{entraînement}$. Cependant, lorsque $Y_{entraînement}$ et Y_{test} sont différents, la méthode n'a pas la possibilité de prédire des valeurs $y \in Y_{test}$ si $y \notin Y_{entraînement}$ donc a fortiori pour ces classes, la valeur de FP est nécessairement 0 lors de l'évaluation. Néanmoins, en pratique il est aussi observé que $Y_{entraînement}$ et Y_{test} ne sont pas disjoints.

Du fait de ce double constat, deux différentes stratégies d'évaluation sont mises en place. La première, comme appliquée dans [85] ou dans [86] est de ne s'intéresser

qu'à $Y_{entraînement} \cap Y_{test}$, et de ne pas considérer les autres prédictions et éléments de la vérité terrain, ou de les considérer comme systématiquement fausses (pour les prédictions). Cela résulte malheureusement en une analyse partielle de la compréhension de la scène d'arrivée. Le nombre de labels résiduels de la stratégie par intersection peut être vu dans le tableau 3.1, on assiste à une forte perte d'information dans de nombreux cas.

Dataset	# labels	# labels ∩ SK	# labels ∩ NS
SemanticKITTI [3]	19	N/A	10
nuScenes [79]	16	10	N/A
Pandaset [82]	37	8	8
Waymo [80]	22	15	11
KITTI-360 [78]	18	15	13
SemanticPOSS [81]	13	11	6

TABLE 3.1 – Nombre de labels de chaque jeu de données et la taille de leurs intersections avec SemantiKITTI et nuScenes.

La deuxième stratégie, comme employée dans [87] ou dans [88] consiste à définir un nouveau jeu de labels $Y_{intermédiaire}$ tel qu'il existe des fonctions f et g telles que $f(Y_{entraînement}) = Y_{intermédiaire}$ et $g(Y_{test}) = Y_{intermédiaire}$ et on peut alors mesurer les performances sur ce nouveau jeu de labels. L'hypothèse forte de l'existence de ces deux fonctions f et g vient de l'observation que ces jeux de données représentent tous des scènes de conduite, et qu'on peut alors supposer que les éléments présents dans ces scènes soient similaires d'un jeu de données à l'autre et que les différences entre $Y_{entraînement}$ et Y_{test} proviennent de choix d'annotations. Ainsi, en proposant des annotations moins précises, on peut retrouver des éléments communs dans ces deux scènes qui seront systématiquement identifiés comme les véhicules ou les piétons. Un exemple de f, g et $Y_{intermédiaire}$ entre nuScenes et SemanticKITTI est donné sur la figure 3.2.

Bien que cette méthode retire l'écueil de la précédente qui exclut certains éléments de la scène dans l'évaluation, elle est loin d'être parfaite. Il peut exister plusieurs $Y_{intermédiaire}$ possibles, et donc le résultat du mIoU est dépendant du choix fait par les différents auteurs. De plus, la mesure finale est elle aussi dégradée par rapport à une évaluation monodomaine, car les labels intermédiaires sont moins précis que ceux d'entraînement, on mesure donc avec moins de précision l'efficacité de la méthode. Finalement, cette stratégie, mais aussi la précédente, est sensible au *label shift*.

3.1.3 Être vigilant au *label shift*

Le terme *label shift* est proposé dans ce travail pour couvrir l'écart entre les annotations de deux jeux de données, même lorsqu'un label semble désigner le même

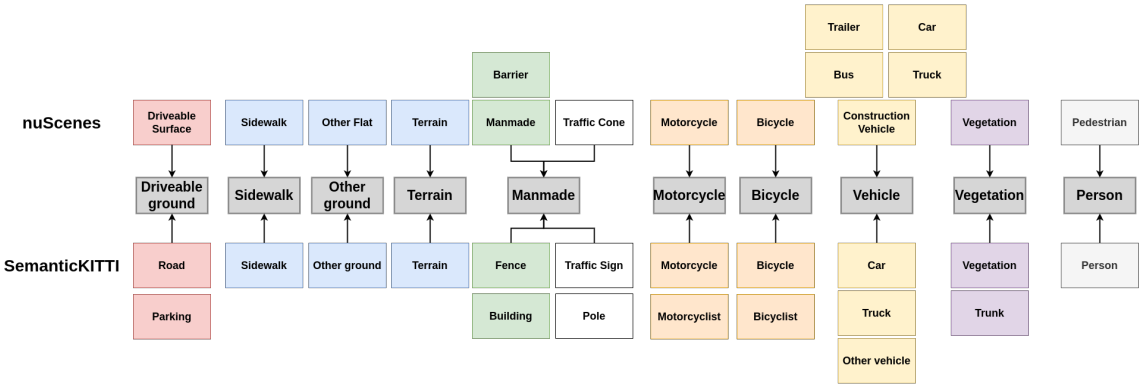


FIGURE 3.2 – Jeux de label intermédiaire entre SemanticKITTI et nuScenes extrait de [88].

élément. Ce décalage est lié à la métadonnée et pas au contenu intrinsèque de la scène représentée. Presque tous les jeux de données disponibles sont annotés par des organismes différents et ceux-ci ont fait des choix d'annotations pour décrire la scène dépendant de l'utilisation qu'ils en imaginaient. Par exemple, nuScenes est initialement un jeu de données destiné à la détection d'objets, et de ce fait distingue de nombreux différents acteurs de la route, mais très peu d'éléments statiques différents. Au contraire, SemanticPOSS identifie tous les véhicules à 4 roues comme des voitures, mais propose une annotation plus variée des éléments de la végétation. Chaque jeu de données a donc une finesse d'annotations propre.

Cependant, malgré cette diversité dans les jeux de données on voit deux schémas communs dans les annotations. D'abord, aucun jeu de données n'a une finesse exhaustive et chaque label regroupe plusieurs éléments atomiques différents. Par exemple, aucun jeu de données ne s'intéresse à la marque des voitures pour la compréhension de la scène, alors que c'est une donnée souvent accessible à un observateur humain ; cela se comprend par la difficulté de retrouver cette information dans la donnée 3D, mais aussi par le coût lié à utiliser une annotation trop fine. Ensuite, certains éléments sont identifiés par tous les jeux de données, comme par exemple les voitures, la route et les piétons.

Supposons qu'il existe un ensemble d'éléments atomiques $A = \{a_i\}_{i\in\{0,N\}}$ qui permet d'exhaustivement décrire tous types de scène de conduite autonome. On peut voir un label comme une union de ces éléments atomiques constituant un sous-ensemble de A. En résumé, le *label shift* correspond à l'existence d'éléments atomiques annotés par deux jeux de données différents avec des labels différents désignant des sous-ensembles de A différents

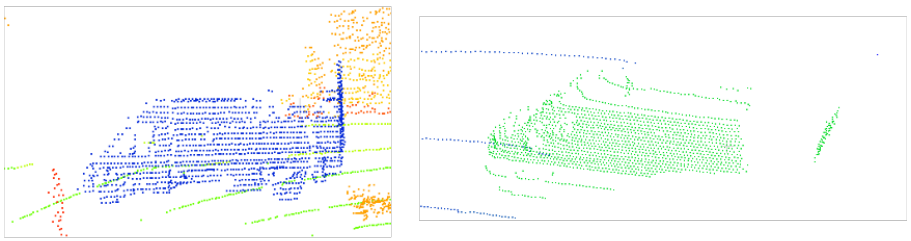
Plus précisément, pour deux jeux de données D_1 et D_2 dont les annotations sont Y^1 et Y^2 composées de respectivement n^1 et n^2 labels décrivant l'intégralité de la scène. On peut noter $Y^1 = \{l_i^1\}_{i\in\{1,n^1\}} = \bigcup_{i\in\{1,N\}} a_i$ et $Y^2 = \{l_i^2\}_{i\in\{1,n^2\}} = \bigcup_{i\in\{1,N\}} a_i$, où les l_i^1 et les l_j^2 décrivent des sous-ensembles de A de tailles respectives

N_i^1 et N_j^2.

On suppose donc que les deux jeux de labels regroupent l'intégralité des éléments atomiques de la scène. Lorsqu'il n'existe aucune correspondance deux à deux des labels, c'est-à-dire que $\forall (i,j) \in \{1, n^1\} \times \{1, n^2\}, l_i^1 = \bigcup_{i \in \{1, N_i^1\}} a_i \neq l_j^2 = \bigcup_{i \in \{1, N_i^2\}} a_i$ et qu'il existe au moins un couple (i_0, j_0) tel que $l_{i_0}^1 \cap l_{j_0}^2 \neq \emptyset$, alors il y a présence de *label shift*.

Dans le cadre où il existe certaines correspondances de labels entre les jeux de données, on peut trouver des sous-parties de Y^1 et Y^2 exposant la même propriété, sinon cela signifie que les jeux de labels sont les mêmes et par définition aucun *label shift* n'est présent.

Par exemple, *sidewalk* et *road* sont deux labels définis à la fois dans nuScenes et SemanticKITTI. Ayant les mêmes noms, il pourrait être attendu que ces labels couvrent les mêmes éléments au sein des scènes de nuScenes et de SemanticKITTI, respectivement les trottoirs et la route. Cependant, on observe qu'aucun de ces jeux de données ne propose de label pour les pistes cyclables, et l'analyse précise des instructions aux annotateurs de ces jeux de données fait apparaître que dans un cas, les pistes cyclables sont rangées dans le label *sidewalk* et dans l'autre cas dans le label *road*. On a donc une situation où ensemble *sidewalk* et *road* décrivent les mêmes éléments à travers deux jeux de données, mais individuellement ils présentent des distinctions. Cela peut être problématique, car les trottoirs et les routes jouent des rôles très différents dans une scène de véhicule autonome et il est intéressant que le modèle de segmentation sache les distinguer, on ne souhaite donc pas les regrouper ensemble lors de l'évaluation. Or un modèle extrêmement bon pour identifier les pistes cyclables dans les deux jeux de données donnera une mauvaise prédiction dans un des deux cas. La figure 3.3 illustre un autre cas où un élément atomique, ici ce que l'on peut considérer être un van, est annoté différemment dans deux jeux de données, soit comme *car* soit comme *truck*.



Car extrait de SemanticKITTI *Truck* extrait de nuScenes

FIGURE 3.3 – Exemple de *label shift*, où un van est labélisé comme *car* ou *truck*, entre SemanticKITTI et Pandaset.

On peut considérer ce cas comme marginal, étant donné que les pistes cyclables couvrent peu de surface dans nos jeux de données, cependant il est intéressant d'être conscient des biais d'annotations pour véritablement mesurer les performances d'une

architecture de segmentation et pas une distance entre deux jeux de labels.

3.2 Notre jeu de données : ParisLuco3D

Ce jeu de données a été intégralement réalisé en collaboration avec Louis Soum-Fontez, co-doctorant au sein du CAOR, et Jean-Emmanuel Deschaud, co-encadrant de cette thèse. Particulièrement, les tâches ont été réparties de la manière suivante : Jean-Emmanuel Deschaud pour l'acquisition des données et la partie localisation, Louis Soum-Fontez pour l'annotation de détection, Jules Sanchez pour l'annotation de segmentation sémantique.

3.2.1 Contexte

Comme cela a été illustré dans la section précédente, il est actuellement complexe de parvenir à effectuer de l'évaluation multidomaine satisfaisante. De cette observation, il nous a semblé nécessaire de mettre au point un jeu de données répondant à ce problème.

Pour le moment, les seuls jeux de données avec lesquels on peut en faire sont les jeux synthétiques, comme c'est le cas dans [89], qui peuvent générer les labels facilement et s'adapter au jeu de données d'entrée. Pour faire cela, nt d'une représentation des objets contenus dans les scènes de conduite bien plus fine que celle qui est demandée par les jeux de labels de segmentation sémantique. Grâce à cette représentation interne, les simulateurs de données connaissent les éléments atomiques permettant de décrire les scènes et peuvent ainsi éviter le label shift.

Dans la philosophie de construction du jeu de labels de ParisLuco3D, nous avons suivi la même idée que celle des simulateurs. L'idée est donc de disséquer la définition des labels des jeux de données de référence, et de réaliser une annotation plus fine de notre jeu de données, pour faire en sorte que nous puissions reformer les différents jeux de données sans ambiguïté en regroupant les annotations proposées.

Au-delà de la finesse de l'annotation, le jeu de données ParisLuco3D présente deux autres spécificités qui le différencie des jeux de données existants. C'est le premier jeu de données acquis et annoté en format véhicule autonome dans un centre dense et urbain européen, précisément Paris. Ce type de scène présente de nombreuses difficultés notamment dues au nombre d'usagers de la route, leur respect relatif des règles de la sécurité routière, ainsi que la présence de terrasses. De plus, il a été acquis avec un HDL-32E à une fréquence de 10Hz, et il présente donc une résolution angulaire différente des autres jeux de données.

Ce jeu de données à vocation à uniquement être utilisé pour l'évaluation de méthodes, et non pour leur entraînement.

3.2.2 Acquisition

Comme mentionné précédemment, ParisLuco3D a été acquis à Paris avec un HDL-32E, fixé sur le toit du véhicule comme les autres jeux de données. L'acquisition a eu lieu précisément dans le 6E arrondissement, autour du jardin du Luxembourg. Le détail de la trajectoire suivie peut être vu dans la figure 3.4. Bien que le tracé soit relativement court, le véhicule a le temps de passer à travers beaucoup de types de scènes différents.

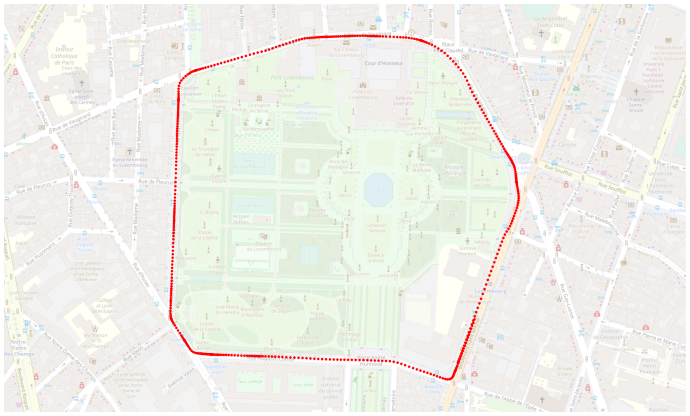


FIGURE 3.4 – Visualisation GPS de la séquence de ParisLuco3D.

On passe notamment par le boulevard Saint-Michel, axe très emprunté par les bus et les piétons, avec notamment deux zones de forte concentration de piétons qui sont le passage piéton rue Gay Lussac et le passage piétons à l'intersection du boulevard Saint-Michel et de la rue de l'Abbé de l'Épée. On a aussi sur ce boulevard une très longue voie de bus, atypique des jeux de données de véhicule autonome. La figure 3.5 illustre ces deux cas.

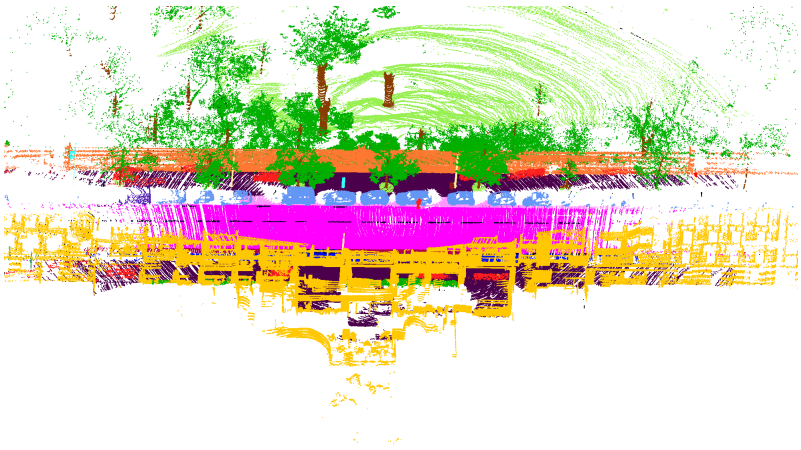


FIGURE 3.5 – Extrait du jeu de données ParisLuco3D.

Ensuite, on a aussi l'occasion de passer par la rue Auguste Comte, qui a la particularité d'être en pente et donc de présenter une route non parallèle au plan xy. On observe aussi des zones de garage à vélo, de nombreux travaux et un rond-point dans ce jeu de données, beaucoup de cas exceptionnels quand on se compare aux jeux existants.

Finalement, on peut aussi noter que le jeu de données, malgré sa modeste taille, présente plusieurs types de météos. En effet, durant l'acquisition il s'est mis à pleuvoir, on a donc à la fois des moments ensoleillés et des moments pluvieux, ce qui entraîne deux phénomènes. Les piétons portent des parapluies, ce qui change de manière non négligeable le profil de leur acquisition, et certaines zones de routes subissent du bruit d'acquisition à cause des flaques au sol.

L'accumulation de ces spécificités rend ce jeu de données très complet et complexe, parfait pour l'évaluation multidomaine.

Le jeu de données est composé de 7500 scans, annotés à 75m, mais dont on ne garantit la qualité de l'annotation que jusqu'à 50m. Le détail des annotations et du processus d'annotation peut être retrouvé dans la sous-section suivante. Un résumé des informations concernant ParisLuco3D peut être trouvé dans le tableau 3.2.

nom	# scans	# séquences	# labels	scène	pays	capteur
ParisLuco3D	7501	1	45	urbain	France	HDL-32E

TABLE 3.2 – Résumé de ParisLuco3D.

3.2.3 Annotations

Ce jeu de données possède trois types d'annotations, mais un seul sera traité en détail dans ce manuscrit : celles de segmentation sémantique. Pour ce qui est des autres annotations, des boîtes de détection sont disponibles pour les usages de la route, ainsi qu'un identifiant de panoptique, permettant de faire du *tracking*. Ces annotations n'ont pas été réalisées par moi, et ne sont seront pas réutilisées dans ce travail, leur détail peut être retrouvé dans la publication associée.

L'observation des annotations des jeux de données existants a fait ressortir des erreurs classiques d'annotations, notamment des confusions entre le sol et les objets mobiles (voir figure 3.6). Afin d'éviter de reproduire ce type d'erreurs, un protocole d'annotations garantissant leur qualité a été mis en place et est le suivant :

— Le premier annotateur annote l'intégralité de ParisLuco3D sémantiquement, sur les nuages accumulés.
— Le second annotateur annote la détection d'objets de ParisLuco3D, nuage par nuage. Simultanément, il reporte les erreurs d'annotations sémantiques.
— Le premier annotateur corrige la sémantique.
— Le troisième annotateur vérifie les annotations.

Ce protocole par double relecture, et par double modalité (nuage par nuage, par nuage accumulé) permet de s'assurer de la finesse de l'annotation des objets statiques et dynamiques.

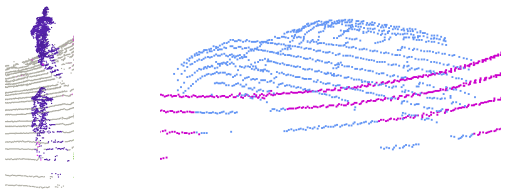


FIGURE 3.6 – Exemple d'erreurs d'annotations, confusion entre la route et des objets dynamiques, dans SemanticKITTI et SemanticPOSS.

Pour ce qui est des annotations sémantiques, 45 labels ont été choisis. Ces annotations ont été choisies suivant plusieurs lignes directrices. D'une part, les jeux de labels de nuScenes et SemanticKITTI ont été décortiqués pour comprendre les éléments atomiques de représentation de la scène contenus dans leurs labels. Ces jeux de données ont été choisis comme référence du fait de leur prédominance dans la littérature. Un scan annoté peut être observé sur la figure 3.7.

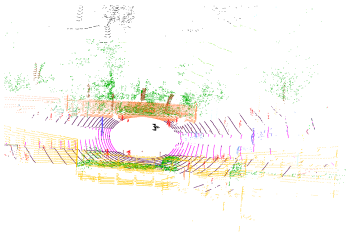


FIGURE 3.7 – Scan extrait de ParisLuco3D, colorisé selon les labels sémantiques.

L'objectif de ParisLuco3D étant de pouvoir regrouper ses labels facilement pour retrouver les annotations de nuScenes et SemanticKITTI, les annotations devaient au moins couvrir les éléments atomiques pris en compte par ces deux jeux de données. De plus, les éléments typiques des scènes ultra-urbaines européennes ont aussi été pris en compte. En effet, certains éléments retrouvés dans ParisLuco3D n'auraient pas pu être observés dans les autres jeux de données, mais sont quand même pris en compte dans ce jeu de données.

La distribution et la liste des labels de ParisLuco3D peuvent être retrouvées sur la figure 3.8. De plus, les regroupements nécessaires pour retrouver les jeux de labels de nuScenes et SemanticKITTI peuvent être vus sur la figure 3.9.

L'annotation sémantique a été réalisée avec le logiciel *point labeler*[1] qui est le même logiciel que celui utilisé pour annoter SemanticKITTI.

1. `https://github.com/jbehley/point_labeler`

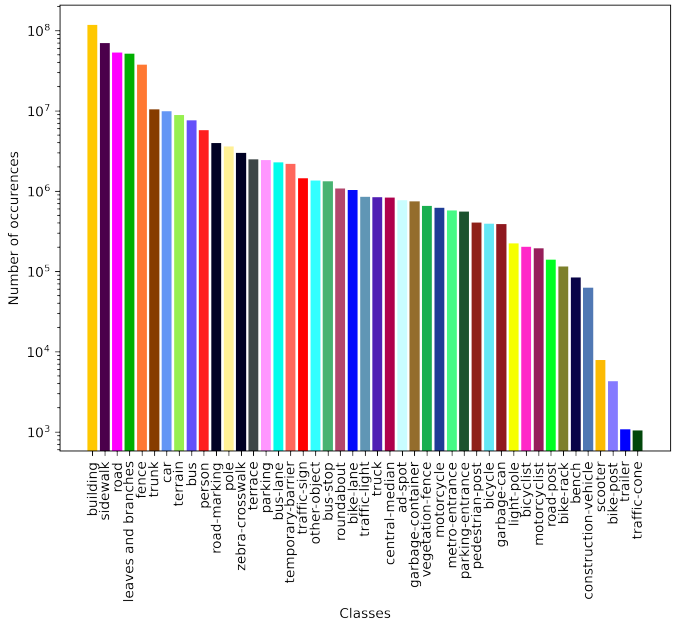


FIGURE 3.8 – Distribution des labels dans ParisLuco3D.

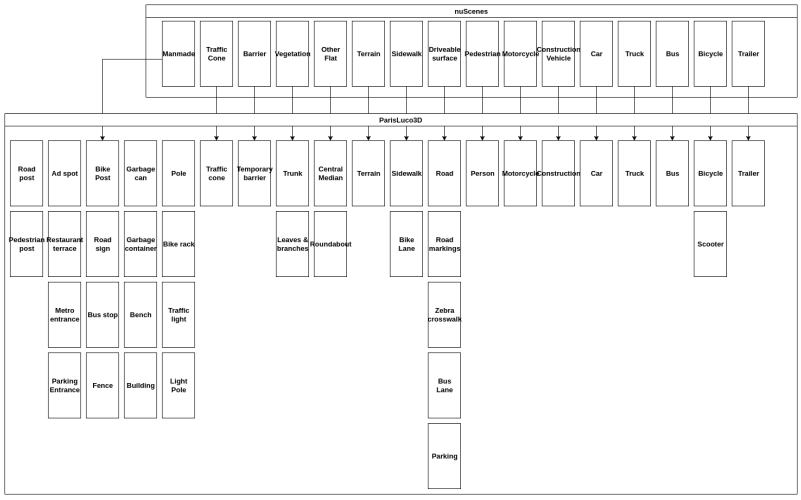


FIGURE 3.9 – Regroupement des labels de ParisLuco3D pour retrouver le jeu de labels de nuScenes.

3.2.4 Détails

Pour conclure sur la présentation de ce jeu de données, intéressons-nous à quelques éléments quantitatifs et qualitatifs, comparativement aux autres jeux de données. Tout d'abord dans le tableau 3.3, nous comparons le nombre total d'instances dans chaque jeu de données, et nous rappelons de plus le nombre de scans dans chacun de ces jeux de données pour recontextualiser l'information. On observe une densité extrêmement grande de piétons par rapport aux autres jeux de données. Il est important de noter pour Waymo, nuScenes et Panda64 que ce sont des jeux de données comportant des séquences très courtes, ce qui favorise une inflation du nombre d'instances, contrairement aux jeux de données avec des séquences plus longues comme SemanticKITTI.

Jeu de données	# véhicules	# piétons	# cyclistes	# scans
SemanticKITTI	2536	201	369	23000
nuScenes	35390	12143	1500	34149
Waymo	60000	23000	620	29667
Panda64	10907	2000	305	6000
PL3D	460	1861	362	7501

TABLE 3.3 – Comparaison du nombre d'usagers de la route entre les différents jeux de données.

Pour ce qui est des informations d'acquisition, dans le tableau 3.4, nous comparons la distance moyenne parcourue par séquence, ainsi que la durée temporelle moyenne de chaque séquence. On observe la particularité de ParisLuco3D d'être une séquence très longue, plus proche d'un trajet réel d'un véhicule autonome que les séquences très fractionnées d'autres jeux de données.

Jeu de données	Distance parcourue (m)	Durée (s)	Résolution angulaire (°) Horizotale/verticale
SemanticKITTI [3]	2015	209	0,08/0,4
KITTI-360 [78]	5793	1618	0,08/0,4
nuScenes [79]	100	20	0,33/1,33
Waymo [80]	103	20	0,16/0,31
SemanticPOSS [81]	230	50	0,2/de 0,33 à 1
Panda64 [82]	82	8	0,2/de 0,17 à 5
ParisLuco3D	2135	750	0,16/1,33

TABLE 3.4 – Comparaison des informations géométriques des séquences entre les différents jeux de données.

Finalement, quantitativement, il peut être observé que ParisLuco3D présente un

nombre de points appartenant à des bus anormalement élevé par rapport aux autres jeux de données. Cela est illustré sur la figure 3.10.

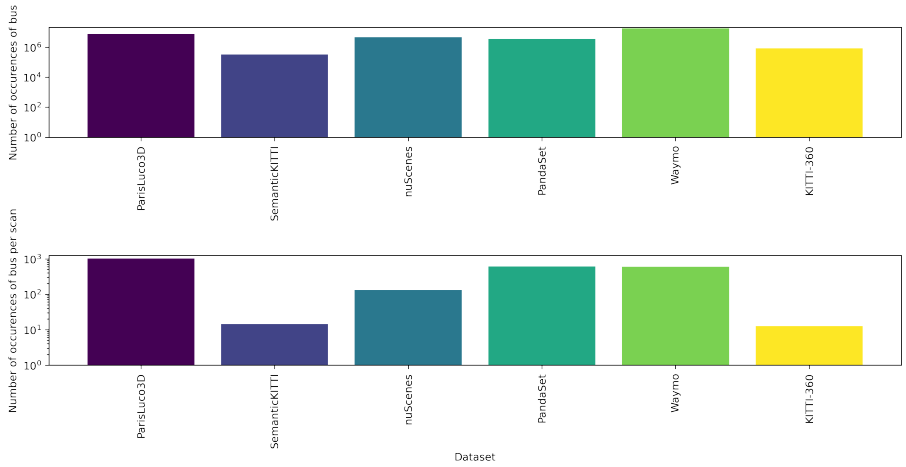


FIGURE 3.10 – Nombres de points annotés comme bus dans chaque jeu de données.

Pour finir, dans l'annexe A, des exemples de tous les labels de ParisLuco3D sont illustrés.

Chapitre 4

Généralisation de domaine monosource : utiliser la géométrie pour faire de l'alignement de domaine

Résumé

Comme présenté dans l'introduction, la tâche de segmentation sémantique 3D monodomaine a atteint des résultats quantitatifs plus que satisfaisants sur les *benchmarks* de référence nuScenes et SemanticKITTI. Dans ce chapitre, l'étude des architectures de segmentation sémantique est étendue au cas multidomaine. L'analyse des résultats nous démontrera des lacunes dans ce cas-là, surtout quand le nombre de scènes et de scénarii de route accessibles à l'entraînement est limité. On s'intéressera donc en détail aux méthodes de segmentation sémantique par la tâche de généralisation de domaine, qui sera introduite et définie dans une première partie. Ensuite, nous présenterons les résultats des méthodes existantes dans le cadre de la généralisation de domaine monosource. Finalement, une piste de réflexion et d'amélioration des performances de généralisation sera proposée par la présentation de 3DLabelProp qui sera étudiée en détail.

Sommaire

4.1 La généralisation de domaine : introduction formelle

4.1.1 Introduction générale

Pour rappel, la généralisation de domaine, dans le cadre de l'apprentissage profond, est la capacité d'un modèle à conserver ses performances lorsque le domaine d'inférence est différent des domaines vus pendant l'entraînement. Contrairement à l'adaptation de domaine, la généralisation de domaine suppose l'impossibilité d'avoir un accès préalable aux données d'inférence, et donc les performances de généralisation sont scellées dès la fin de l'entraînement. Un schéma explicatif (voir figure 1.4), a été proposé dans l'introduction.

Un des cas de généralisation les plus classiques et les plus étudiés lors de la naissance des modèles d'apprentissage pour la 2D est l'invariance aux transformations géométriques, comme la rotation ou la translation. Les autres cas classiques de sources de changements de domaine en 2D couvrent aussi les changements de colorimétrie et d'illumination, l'impact des saisons sur la scène, le flou, l'angle de vue ou bien les éléments de la scène. L'ensemble des sources de variations de domaine seront appelées les *domain shifts* dans le reste de ce travail. Une illustration de plusieurs de ces cas peut être vue sur la figure 4.1.

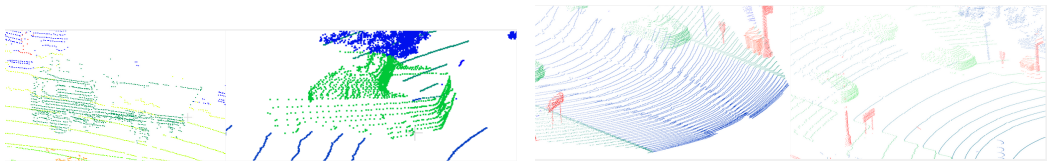
FIGURE 4.1 – Illustration de différents *domain shifts* en 2D [90].

Dans le cadre de la segmentation sémantique 3D pour le véhicule autonome, on distingue un ensemble de *domain shifts* différent de celui de la 2D. Notamment, il existe certaines invariances natives du capteur d'acquisition, notamment l'insensibilité au changement d'illumination de la scène, et donc de fait l'insensibilité au moment de la journée. Ces *domain shifts* peuvent être séparés en trois grandes catégories :

— *sensor shift* : variation de l'ensemble des facteurs liés au capteur qui peuvent avoir un impact sur la topologie du nuage de points acquis. Cela correspond donc d'une part aux spécifications du capteur comme sa technologie, sa résolution verticale et horizontale, son nombre de fibres, et d'autre part à ses caractéristiques externes notamment son positionnement.

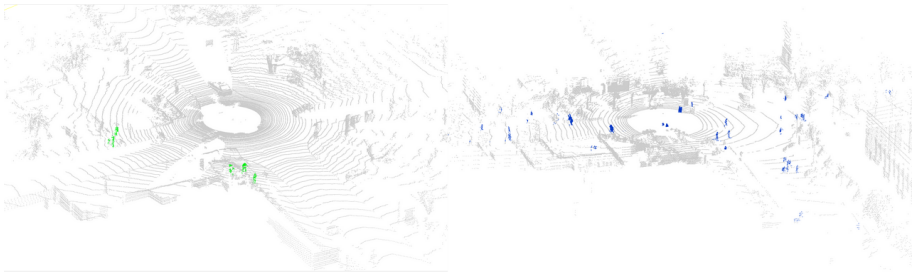
— *scene shift* : la composition de la scène acquise dont son agencement, la quantité d'éléments qui la composent, les facteurs climatiques et météorologiques ainsi que les règles de circulation.
— *appearance shift* : variation de l'aspect visuel de chaque élément de la scène comme la végétation, les véhicules et la route.

Ces différents *domain shifts* spécifiques à la segmentation sémantique 3D pour le véhicule autonome sont illustrés sur la figure 4.2. L'*appearance shift* est illustré avec deux éléments *truck* extraits de SemanticKITTI et Pandaset. Le *sensor shift* est illustré grâce à Panda64 et PandaFF, en regardant la même route échantillonnée par les deux capteurs présents pour l'acquisition de Pandaset. Finalement, le *scene shift* est illustré en comparant deux nuages de points extraits de SemanticKITTI et SemanticPOSS, et en mettant en valeur les piétons dans les deux cas. Pour SemanticPOSS, on observe une densité de piétons plus importante et répartie à des endroits différents de la route relativement à SemanticKITTI.



(a) *appearance shift.*

(b) *sensor shift.*



(c) *scene shift.*

FIGURE 4.2 – Illustration de différents *domain shifts* en 3D.

4.1.2 La généralisation de domaine dans la littérature

La généralisation de domaine est largement traitée en apprentissage machine, et dans de nombreux domaines de l'apprentissage profond qui ne sont pas connexes au nôtre, notamment en *NLP*. Des vues d'ensemble de ce sujet peuvent être trouvées dans [91, 92]. Bien que les présentations générales trop lointaines des problématiques de la vision ne nous intéressent pas, la typologie des méthodes de généralisation de domaines a de l'intérêt.

On peut séparer ce champ de recherche en deux catégories, selon la variété de don-

nées accessibles à l'entraînement : la généralisation monodomaine et la généralisation multidomaine. Une fois ce choix effectué, il existe de nombreuses méthodes différentes pour améliorer les performances de généralisation dont le meta-apprentissage, l'apprentissage multitâche, l'augmentation de données, le design d'architecture neuronale et l'alignement de domaines. Toutes ces méthodes ont principalement été étudiées et développées dans le cadre de la vision 2D. Illustrons ces différents cas.

Le meta-apprentissage correspond à l'ensemble des techniques qui s'intéressent à changer la boucle d'apprentissage et à en optimiser les paramètres, même au cours d'un entraînement. L'idée est d'exposer le modèle à du *domain shift* lors de l'entraînement en divisant les données d'entrée pour chaque boucle d'apprentissage en deux parties, le *meta-train* et le *meta-test*, et d'utiliser les données de *meta-train* pour optimiser les performances sur les données de *meta-test*. MAML [93] et MLDG [94] transforment la fonction de perte, du premier ordre, en une du second ordre pour prendre en compte l'impact de l'optimisation du réseau par un batch de *meta-train* sur les performances de *meta-test*. Un schéma illustratif de ces méthodes peut être trouvé sur la figure 4.3. L'*episodic training* [95] apprend à plusieurs réseaux à être performants sur un domaine spécifique et à compenser les performances des autres réseaux qui ne sont pas efficaces sur son domaine. Ces méthodes supposent donc l'accès à une variété de domaines de façon à ce qu'il existe un *domain shift* entre le *meta-train* et le *meta-test*, et sont donc généralement appliqués dans le cadre de la généralisation multidomaine.

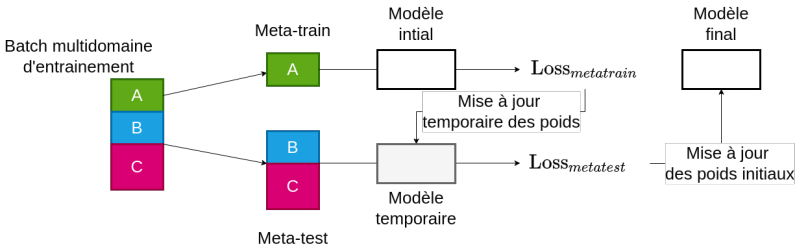


FIGURE 4.3 – Schéma explicatif de MLDG [94].

L'apprentissage multitâche correspond à l'apprentissage simultané, ou consécutif, de plusieurs tâches distinctes par un réseau de neurones. L'objectif de ces méthodes est d'apprendre des représentations internes au réseau de neurones qui soient plus générales, car permettant de réaliser plusieurs tâches, et donc de rendre les réseaux plus robustes au changement de domaine, comme proposé par [96]. Pour aller plus loin, [97] s'entraîne à replacer les éléments d'une image au bon endroit pour apprendre des représentations géométriques plus intéressantes et pertinentes.

Les méthodes d'augmentation de données ont pour but de diversifier les données d'entraînement en appliquant des *domain shift* contrôlés aux données d'entrées. Les méthodes classiques comme les rotations ou l'application de bruit ont montré qu'elles pouvaient créer des invariances et de la robustesse à certaines transformations, cer-

taines étant illustrées dans la figure 4.4, mais des méthodes plus poussées peuvent aider les capacités de généralisation. Par exemple, [98] applique des opérations de convolution aléatoire aux images pour les déformer et ainsi forcer le réseau à être résilient au changement de texture. Les méthodes basées sur les *adversarial data augmentations* [99] ajoutent aux exemples d'entraînement des exemples fictifs créant des difficultés au réseau entraîné afin d'améliorer ses capacités de généralisation.

FIGURE 4.4 – Exemples d'augmentations de données 2D [100].

Comme cela a été montré dans le chapitre précédent, certaines des clefs de design des architectures de convolution datent de la fin des années quatre-vingt-dix et ont assez peu évolué depuis, notamment la couche de convolution. Ces dernières années, certains travaux ont étudié ces couches et ont cherché à en modifier certaines pour en améliorer les performances de généralisation. Une large partie des travaux se sont intéressés à modifier les couches de normalisation [101, 102, 103], avec notamment IBN-Net [103]. L'idée d'IBN-Net est de combiner les couches d'*Instance Normalization* qui permet d'apprendre des invariances d'apparence à celles de *Batch Normalization* qui permet de conserver les informations contenues dans les images. Une application ingénieuse des deux permet d'atteindre les deux effets à la fois et de robustifier les modèles aux changements d'apparence. Une des implémentations peut être vue sur la figure 4.5.

Pour finir, le dernier courant de généralisation de domaine est l'alignement de domaines. L'idée est de réussir à rapprocher les domaines existants entre eux, pour que le modèle puisse apprendre des représentations insensibles aux changements de domaine. Si l'on visualise les domaines inconnus comme provenant de mélange de domaines connus, créant une nouvelle distribution, alors être robuste aux mélanges des domaines connus permet d'améliorer les performances de généralisation. Ceci est

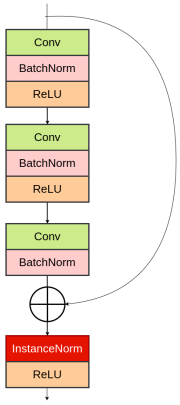


FIGURE 4.5 – Schéma d' IBN-Net-b, une des implémentations de [103].

l'idée de [104], et aussi indirectement celle de [105] qui cherche à minimiser l'écart de représentation entre chaque paire de domaine, pour réduire celui entre chaque paire de mélanges. [106] cherche à coupler les fonctions de perte de classification et *contrastive* pour pousser les éléments appartenant à la même classe mais à des domaines différents à être proches dans l'espace des caractéristiques. Ainsin on peut supposer la représentation interne indépendante du domaine, mais dépendante de la classe et donc améliorant les performances de généralisation.

Cette section a pour objectif de donner les clefs principales pour construire des méthodes de généralisation de domaine, mais elle n'a pas vocation à faire un panorama total des méthodes. Celles-ci étant conçues pour la 2D, elles ne se transfèrent pas systématiquement à la 3D, nous le montrerons dans les sections suivantes. Pour un approfondissement des méthodes et un listing plus exhaustif de celles-ci, on peut se référer aux travaux mentionnés au début de cette section [91, 92].

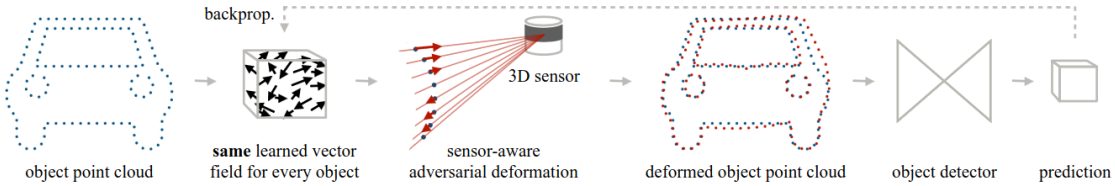
4.1.3 La généralisation de domaine pour la segmentation sémantique 3D

La section précédente présentait les méthodes habituelles pour faire de la généralisation de domaine, mais elles étaient cantonnées à la vision 2D. La vision 3D étant un champ de recherche plus récent, peu de travaux se sont intéressés à faire de la généralisation, ou même à seulement adapter les méthodes de la 2D à la 3D.

Une méthode s'intéresse spécifiquement à la détection d'objets 3D, appelée 3D-VField [107] et est en cours d'extension à la segmentation sémantique 3D. DGLSS [87] et LIDOG [108] traitent la généralisation monosource pour la segmentation sémantique 3D. Finalement, [109] s'intéresse spécifiquement à la robustesse des méthodes 3D en conditions d'environnement dégradé.

3D-VField [107] est une méthode d'augmentation de données pour la généralisation de domaine par attaque *adversarial*. Les méthodes *adversarial* se basent sur l'utilisation d'exemples créés pour mettre le réseau en défaut ; en introduisant ces

exemples dans les boucles d'entraînement, on robustifie la méthode. Ici, l'idée est de trouver un champ de déformation *adversarial* à appliquer aux objets de la scène pour les rendre plus compliqués à détecter. Ces transformations sont tout de même contraintes pour s'assurer que le nuage de points déformé continue de ressembler à l'objet qu'il représente. Pour cela, le champ de déformation est cohérent par rapport à la direction des rayons d'acquisition émis par le capteur LiDAR. Cette méthode est initialement confinée à la tâche de détection d'objet, car elle nécessite de connaître les botes englobantes autour de ceux-ci pour contraindre la déformation, cependant elle peut être adaptée sur les annotations de segmentation panoptique quand elles sont disponibles. Le procédé d'augmentation de données est illustré par la figure 4.6.



Overview of the proposed 3D-VField. We first learn a vector field adversarially to plausibly deform objects, taking constraints into account. The modified scenes are later used as augmentations to improve the generalization to unseen object shapes.

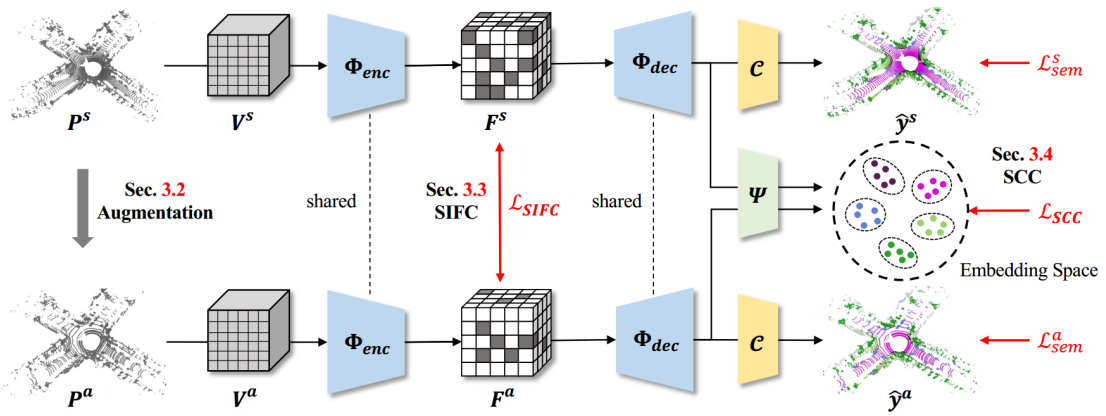
FIGURE 4.6 – Image extraite de l'article original expliquant 3D-Vfield [107].

DGLSS [87] peut être classé dans les méthodes d'alignement de domaines, car cette méthode cherche à forcer le réseau de neurones à apprendre des caractéristiques indépendantes de l'éparsité de la scène. Pour parvenir à cela, lors de l'entraînement, chaque nuage de points étudié est augmenté par du *ray dropping* et le réseau de neurones doit réussir à sémantiser le nuage original et le nuage ainsi dégradé. En plus de ce double apprentissage, deux fonctions de perte auxiliaires sont rajoutées pour que les caractéristiques extraites par l'*encoder* soient similaires pour les deux niveaux de densité. De plus, la seconde fonction de perte vient forcer la proximité des représentations en sortie de *decoder*. Un schéma du fonctionnement de cette méthode est présent sur la figure 4.7.

Ce travail est le premier en généralisation de domaine pour la segmentation sémantique 3D pour le véhicule autonome. En plus de cela, il étudie aussi quelques méthodes de généralisation de données de la 2D (MLDG et IBN-Net) et démontre leur inefficacité pour la 3D.

LIDOG [108] s'intéresse au même problème que DGLSS. Pour augmenter les performances de généralisation, en sortie du *decoder* du réseau se trouvent deux têtes de segmentation. Une, habituelle, travaille sur les voxels, et l'autre sémantise une projection *bird's eye view* des caractéristiques extraites du *decoder*.

Bien que [109] propose une méthode de généralisation, le cœur de leur travail est l'analyse en profondeur du comportement de nombreuses méthodes de segmentation sémantique face à différents cas de dégradation des conditions d'acquisition, soit dûs



Overall DGLSS framework. From the single source domain P^s, we augment a new domain with different sparsity P^a to learn sparsity-invariant representations. The two domains are encoded by Φ_{enc}, and the encoded internal features F^s and F^a are constrained with the proposed *sparsity invariant feature consistency* (SIFC). The decoded features from Φ_{dec} are fed to the metric learner Ψ to construct a feature embedding space. The embedding space is constrained by the proposed *semantic correlation consistency* (SCC). The semantic class predictions $\hat{y}^s$ and $\hat{y}^a$ are supervised by the semantic segmentation ground truth for both source and augmented domains.

FIGURE 4.7 – Image extraite du papier original expliquant DGLSS [87].

à des paramètres météorologiques soit dûs à des paramètres capteurs. Pour cela, il crée des jeux de données alternatifs à SemanticKITTI appelés SemanticKITTI-C. Ils représentent les mêmes scènes que SemanticKITTI dans des cas de forte neige, de fort brouillard, d'existence d'*outliers* d'acquisition, de distorsion locale, d'un capteur 32 fibres ou d'un capteur 16 fibres d'acquisition. Cela correspond à 16 jeux de données auxiliaires. 11 méthodes sont comparées sur ces jeux de données et arrivent à la conclusion que KPConv et SRU-Net sont les deux architectures nativement les plus robustes.

Alors que DGLSS [87] s'intéresse à la généralisation de domaine globale, on peut considérer que [109] s'intéresse à la généralisation de domaine visant spécifiquement la qualité de l'acquisition, ce sont ainsi des travaux complémentaires plutôt que concurrents.

4.1.4 L'adaptation de domaine non supervisée pour la segmentation sémantique 3D

Bien que la généralisation de domaine soit relativement inexplorée dans le domaine de la 3D, l'adaptation de domaine est un sujet beaucoup plus étudié. Dans le cadre de l'adaptation de domaine, on peut supposer disponibles, en amont de l'inférence, quelques exemples des données attendues. Dans le cadre plus restrictif de l'adaptation de domaine non supervisée, qui nous intéressera plus particulièrement dans cette section, les données brutes sont disponibles, mais sans annotations associées.

Pour l'adaptation de domaine, on distingue deux cas : synthétique vers réel et

réel vers réel. Dans le premier cas, on suppose que l'entraînement initial des modèles d'apprentissage se fait uniquement avec des données synthétiques. À l'inverse, dans le second cas, des données réelles sont aussi utilisées pour l'entraînement.

L'adaptation de domaine n'étant pas le cœur de ce travail, dans la suite de cette section, certaines des méthodes phares seront illustrées, mais une revue exhaustive ne sera pas proposée.

Différentes stratégies sont employées pour réaliser l'adaptation de domaine, en fonction du cas dans lequel les méthodes se trouvent. Dans le cadre du synthétique vers réel, on peut supposer la quantité de données d'apprentissage très grande et éventuellement couvrant un grand nombre de domaines. Cependant, les simulateurs actuels ne modélisent pas de manière efficace les statistiques de bruit et de déformation des capteurs. Les premières méthodes d'adaptation de domaine s'attaquent à ce problème en transférant le style des nuages de points réels vers ceux simulés pour améliorer l'entraînement [110, 32, 111]. Les méthodes plus récentes, comme Cosmix [89], plutôt que de transférer directement le style avec des méthodes de type GAN complexes à mettre en place, s'intéressent à mixer des patchs réels et synthétiques pour pseudo-annoter les patchs réels et ensuite s'entraîner avec. Ces méthodes sont dites de *pseudo labeling*.

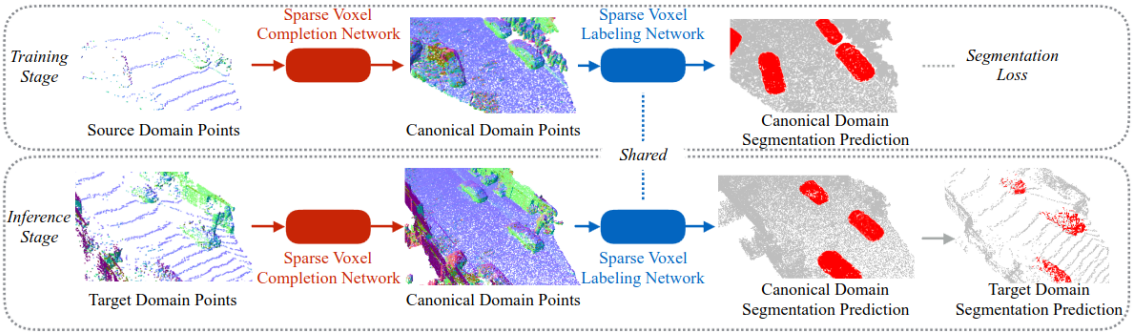


FIGURE 4.8 – Image extraite du papier original expliquant Complete & Label [85].

Pour ce qui est de l'adaptation réel vers réel, on peut supposer les statistiques d'acquisition déjà connues, mais il existe une forte sensibilité à la résolution du capteur dans les performances des méthodes de segmentation. Cela a été mis en avant par [112] qui propose de rééchantillonner les nuages de points nouvellement acquis pour les aligner avec ceux de l'entraînement et obtenir une résolution similaire. Complete & Label [85] propose de travailler sur une représentation intermédiaire, continue, obtenue par un réseau de complétion. La complétion peut être apprise de manière non supervisée sur les nouvelles données et on suppose que les nuages complétés d'entraînement et d'inférence sont suffisamment semblables pour conserver les performances. Une illustration de cette méthode se trouve dans la figure 4.8. Saluda [113] se base sur une idée similaire, celle de l'existence d'une représentation commune indépendante du capteur à travers les nuages complétés, mais l'inclut dans la boucle

d'entraînement pour forcer le réseau à extraire des caractéristiques indépendantes de la résolution d'entraînement.

4.2 Panorama des performances actuelles

Bien que [109] propose un large panorama de la mesure de la robustesse des méthodes de segmentation sémantique, celui-ci est cantonné à la mesure de performance sur des jeux de données synthétiques relativement proches du jeu de données d'entraînement. Dans les travaux de DGLSS [87], certes l'augmentation des performances de généralisation par cette méthode est démontrée, cependant elle n'est appliquée qu'à une seule architecture de segmentation. En somme, au vu des travaux actuels, il est difficile d'estimer les performances de généralisation natives des différentes méthodes de segmentation sémantiques composant l'état de l'art. Indirectement, en mettant bout à bout les *baselines* des différents articles traitant de l'adaptation de domaine, on peut extraire quelques premiers résultats de généralisation des méthodes classiques, cependant ce constat est incomplet et parfois réalisé sur des jeux de labels différents.

Dans la suite de cette section, un panorama le plus exhaustif possible sera mis en place pour comprendre les performances et les limitations des méthodes actuelles dans plusieurs cas de généralisation.

4.2.1 Comprendre les expériences

L'objectif de ce panorama est de comprendre les forces et faiblesses des différentes méthodes de l'état de l'art en segmentation sémantique 3D. Pour cela, nous souhaitons étudier chacun des trois *domain shifts* de la 3D, à savoir le *scene shift*, le *sensor shift* et l'*appearance shift*, indépendamment et ensemble. Il est donc important de porter beaucoup d'attention à la création de deux ensembles : un groupe de méthodes de sémantisation représentant exhaustivement l'état de l'art, et un ensemble d'expériences permettant d'étudier les différents *domain shifts*.

Choix des méthodes

Pour la sélection des méthodes, plusieurs critères sont entrés en jeu : la disponibilité du code, la disponibilité des poids des réseaux préentraînés, la représentation de chaque paradigme de segmentation et les performances sur les *benchmarks* de référence (nuScenes et SemanticKITTI). Les modèles retenus sont : Cylinder3D [65], KPConv [37], SRU-Net [9], SPVCNN [69], Helix4D [75] et CENet [55]. Ces méthodes couvrent la plupart des paradigmes (basée point, basée voxels, basée projection, fusion des représentations, basée séquence) et sont reconnues comme étant très performantes.

Une nuance importante concernant Cylinder3D [65] vis-à-vis des autres méthodes citées ici est la présence d'une étape de prétraitement des points avant la voxelisation. En effet, un réseau PointNet est entraîné pour prendre en entrée les points composant un voxel et en ressortir un vecteur de caractéristiques qui est utilisé en entrée du réseau convolutionnel. Les autres méthodes n'utilisent pas de tel réseau avant la voxelisation ou la projection.

Choix des expériences

Pour mesurer efficacement les performances de généralisation lorsque le domaine est soumis aux différents *domain shifts*, il est important de prendre le temps de comprendre les *shifts* entre chaque jeu de données, pour essayer d'isoler les *domain shifts* et comprendre ce qui pose des difficultés aux différentes méthodes. De plus, en raison de leur taille, et pour être en accord avec les expériences et les études réalisées dans la littérature, SemanticKITTI et nuScenes seront les jeux de données d'entraînement.

Un récapitulatif des expériences peut être trouvé dans le tableau 4.1.

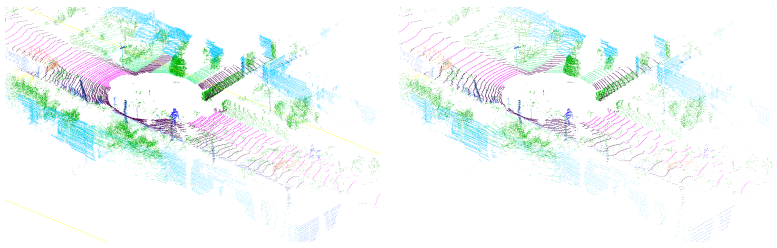
Jeu de données source	Jeu de données cible	Objectif
SemanticKITTI	SemanticKITTI-32	Vérifier la robustesse au *sensor shift*
SemanticKITTI	Panda64 et PandaFF	Vérifier la robustesse au *sensor shift*
SemanticKITTI	SemanticPOSS, Waymo et Panda64	Vérifier la robustesse au *scene shift*
SemanticKITTI	nuScenes	Vérifier la robustesse aux *domain shifts*
SemanticKITTI	ParisLuco3D	Vérifier la robustesse aux *domain shifts* sans concession sur le jeu de labels cible
nuScenes	SemanticKITTI et SemanticKITTI-32	Vérifier la robustesse au *sensor shift*
nuScenes	Panda64 et PandaFF	Vérifier la robustesse au *sensor shift*
nuScenes	SemanticPOSS, Waymo et Panda64	Vérifier la robustesse aux *domain shifts*
nuScenes	ParisLuco3D	Vérifier la robustesse au *scene et appearance shift* sans concession sur le jeu de labels cible

TABLE 4.1 – Récapitulatif des expériences de généralisation.

Sept jeux de données cibles sont identifiés : SemanticKITTI (SK), SemanticPOSS (SP), nuScenes (NS), Waymo (W), Panda64 (P64), PandaFF (PFF), ParisLuco3D (PL3D). De plus, un jeu de données, obtenu par sous-échantillonnage, est ajouté : SemanticKITTI-32, une version dégradée de SemanticKITTI où les points acquis par un laser sur deux sont retirés. Un exemple de ce jeu de données est présenté sur la figure 4.9. Avec deux jeux de données sources, nous arrivons à 14 expériences, chacune ayant un objectif différent.

Utiliser deux jeux de données permet de s'assurer que les résultats observés ne sont pas conditionnés à un capteur précis. De plus, par leur différence sur le plan de la scène et de l'acquistion, ces deux jeux de données permettent de couvrir un plus large panel de *domain shifts*.

Les jeux de données sources ont des rôles différents. SemanticPOSS a la particularité d'avoir été relevé dans un campus et donc de contenir une grande quan-



(a) SemanticKITTI. (b) SemanticKITTI-32.

FIGURE 4.9 – Nuage de points extrait de SemanticKITTI-32, comparé au même nuage dans SemanticKITTI.

tité de piétons et de vélos, avec des comportements plus erratiques que dans les autres jeux de données. On retrouve notamment bien plus de piétons sur la route que dans les autres jeux de données. Panda64 et PandaFF représentent les mêmes scènes, mais sous deux capteurs différents, représentant un cas unique de *sensor shift*. SemanticKITTI-32 a été construit dans la même optique pour mesurer la dépendance à la résolution du capteur relativement à celui utilisé à l'entraînement. Waymo permet une évaluation sur une quantité de données plus importantes que les jeux de données précédents, avec une finesse d'annotation plus intéressante ; il est de plus acquis avec un capteur différent de ceux de SemanticKITTI et nuScenes. Finalement, ParisLuco3D permet de mesurer les performances de généralisation sans concession sur la finesse des labels par sa conception et permet donc les meilleures conclusions vis-à-vis des questions de généralisation globale.

Comme mentionné dans le chapitre 3, il est important d'être attentif au *label shift*, et nous définissons donc précisément les jeux de labels communs lorsque cela est nécessaire. Ils peuvent être retrouvés dans la section suivante.

4.2.2 Spécifications des protocoles expérimentaux

Pour réaliser le panorama de généralisation, les six méthodes de segmentation sémantique ont été réentraînées sur deux jeux de données source : SemanticKITTI et nuScenes. Pour conserver les performances mises en avant par les auteurs, le même choix de paramètres a été réalisé. Lorsqu'aucun jeu de paramètres n'était proposé pour les entraînements avec nuScenes, les mêmes paramètres que pour SemanticKITTI ont été choisis. Les augmentations de données choisies sont les classiques, c'est-à-dire celles mises en avant par [9] : rotation autour de z, facteur d'échelle, bruit gaussien local.

Ces modèles ont été réentraînés sans l'utilisation de la caractéristique de réflectivité. On rappelle que la réflectivité est une valeur fournie par le capteur pour chaque point d'un nuage de points, qui prend en compte le matériau constitutif de l'objet échantillonné et l'angle d'incidence du rayon. Les calculs menant à une valeur de

réflectivité appartiennent aux fabricants de capteurs et changent d'une marque à l'autre, et même d'un modèle à l'autre. Pour ces raisons, cette valeur a simplement été retirée et est remplacée par l'occupation lors de l'entraînement. Pour appuyer ces propos, une expérience quantitative est proposée dans la sous-section 4.2.3.

Dans le chapitre précédent, la complexité de l'évaluation multidomaine a été mise en évidence. Pour parvenir à l'évaluation, nous définissons un jeu de labels pour chaque paire de jeux de données considérés ensemble. Cela correspond à 9 jeux de labels :

- $\mathcal{L}_{SK\cap SP}$: person, rider, bike, car, ground, trunk, vegetation, traffic sign, pole, building, fence
- $\mathcal{L}_{SK\cap PS}$: 2-wheeled, pedestrian, driveable ground, sidewalk, other ground, manmade, vegetation, 4-wheeled
- $\mathcal{L}_{SK\cap NS}$: motorcycle, bicycle, person, driveable ground, sidewalk, other ground, manmade, vegetation, vehicle, terrain
- $\mathcal{L}_{SK\cap W}$: car, bicycle, motorcycle, truck, vegetation, sidewalk, road, person, bicyclist, motorcyclist, trunk, other-vehicle, sign, pole, building, other-ground
- $\mathcal{L}_{NS\cap SP}$: person, bike, car, ground, vegetation, manmade
- $\mathcal{L}_{NS\cap PS}$: 2-wheeled, pedestrian, driveable ground, sidewalk, other ground, manmade, vegetation, 4-wheeled
- $\mathcal{L}_{NS\cap W}$: car, truck, bus, other vehicle, motorcycle, bicycle, pedestrian, traffic cone, manmade, vegetation, driveable road, sidewalk, other ground
- $\mathcal{L}_{SK\cap PL3D}$: car, bicycle, motorcycle, truck, other-vehicle, person, road, parking, sidewalk, other-ground, building, fence, vegetation, trunk, terrain, pole, traffic-sign
- $\mathcal{L}_{NS\cap PL3D}$: barrier, bicycle, bus, car, construction vehicle, motorcycle, pedestrian, traffic cone, trailer, truck, driveable surface, other flat, sidewalk, terrain, manmade, vegetation

On peut remarquer que $\mathcal{L}_{NS\cap PL3D}$ est équivalent à $\mathcal{L}_{NS}$ et que $\mathcal{L}_{SK\cap PL3D}$ est équivalent à $\mathcal{L}_{SK}$ en regroupant les cyclistes et leurs engins.

Bien que l'existence d'autant de jeux de labels différents rende la comparaison directe entre les résultats plus compliquée, il semblait important de les définir individuellement pour calculer avec le plus de finesse possible la qualité de la généralisation. Opter pour un jeu de labels global, applicable à chaque expérience, a pour effet de faire disparaître des catégories mises en avant par certains jeux de données.

Pour calculer les performances de segmentation sémantique, comme vu dans le chapitre précédent, la métrique de choix est l'IoU par classe, et sa version moyennée le $mIoU$. Le $mIoU$ est donc dépendant du jeu de labels choisi. Pour clarifier les notations, dans le cadre de l'évaluation multidomaine, nous noterons les $mIoU$: $\text{mIoU}^{test}_{\text{jeu de labels}}$. Par exemple, lorsqu'on calcule les résultats de généralisation sur SemanticPOSS sur le jeu de labels commun qu'il a avec SemanticKITTI $\mathcal{L}_{SK\cap SP}$ avec comme jeu d'entraînement SemanticKITTI (SK), on calcule le $mIoU^{SP}_{\mathcal{L}_{SK\cap SP}}$.

4.2.3 Étude de l'influence de la réflectivité

Dans le tableau 4.2, les différents modèles sont entraînés avec SemanticKITTI avec ou sans la réflectivité puis testés sur SemanticPOSS. SemanticPOSS est choisi pour l'analyse de la réflectivité, car les résolutions angulaires de son capteur d'acquisition sont similaires à celles de SemanticKITTI, et de plus, les réflectivités suivent des distributions similaires.

Au-delà de l'augmentation des performances de généralisation, on observe aussi une dégradation des performances monodomaine. Le cœur de ce travail étant la généralisation, en retirant la réflectivité nous pouvons donner une borne supérieure des performances de chacune des méthodes. En pratique, il serait nécessaire de considérer les performances dans les deux cas.

Méthode	Réflectivité	$\text{mIoU}^{SK}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{SP}_{\mathcal{L}_{SK\cap SP}}$
CENet [55]	✓	61,4	27,5
	×	58,8	**27,9**
Helix4D [75]	✓	63,1	35,0
	×	60,0	**36,0**
KPConv [37]	✓	59,9	33,1
	×	58,3	**39,1**
SRU-Net [9]	✓	61,9	45,2
	×	58,6	**45,3**
SPVCNN [69]	✓	63,8	36,9
	×	62,3	**45,4**
C3D [65]	✓	66,9	33,8
	×	60,7	**41,9**

TABLE 4.2 – Étude de l'effet de la réflectivité.

4.2.4 Étude des *domain shifts*

Dans cette section, nous montrons les résultats de généralisation pour tous les jeux de données cibles hormis ParisLuco3D. Dans le tableau 4.3, SemanticKITTI est le jeu de données source, dans le tableau 4.4 il s'agit de nuScenes. La conclusion générale est l'incapacité des méthodes de référence de nativement réaliser de la généralisation de domaine. En effet, il est important de prendre en considération que hormis les résultats lorsque le jeu de données source et cible sont les mêmes, les valeurs de mIoU sont calculés sur des jeux de labels simplifiés. Et malgré la simplification de la tâche, les résultats quantitatifs sont faibles donc insuffisants pour l'application de perception 3D pour la conduite autonome. Le détail des résultats par classe des différents cas de généralisation peut être trouvé dans l'annexe B.

Méthode	Paradigme d'entrée	$\text{mIoU}^{SK}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{SK32}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{P64}_{\mathcal{L}_{SK\cap PS}}$	$\text{mIoU}^{PFF}_{\mathcal{L}_{SK\cap PS}}$	$\text{mIoU}^{SP}_{\mathcal{L}_{SK\cap SP}}$	$\text{mIoU}^{W}_{\mathcal{L}_{SK\cap W}}$	$\text{mIoU}^{NS}_{\mathcal{L}_{SK\cap NS}}$
CENet [55]	Basée projection	58,8	39,1	13,3	4,9	27,9	7,4	5,0
Helix4D [75]	Basée séquence & 4D	60,0	53,2	27,7	14,2	36,0	N/A [1]	34,3
KPConv [37]	Basée point	58,3	52,7	32,7	21,1	39,1	17,5	**46,7**
SRU-Net [9]	basée voxels	58,6	54,0	**44,2**	**22,2**	45,3	33,1	42,7
SPVCNN [69]	basée voxels et point	**62,3**	**57,4**	40,2	19,4	**45,4**	29,7	45,1
C3D [65]	basée voxels cylindrique	60,7	53,1	18,4	6,5	41,9	18,9	32,7

TABLE 4.3 – Panorama des performances des méthodes de référence entraînées avec SemanticKITTI sous *domain shifts*.

Méthode	Paradigme d'entrée	$\text{mIoU}^{NS}_{\mathcal{L}_{NS}}$	$\text{mIoU}^{SK}_{\mathcal{L}_{NS\cap SK}}$	$\text{mIoU}^{SK32}_{\mathcal{L}_{NS\cap SK}}$	$\text{mIoU}^{P64}_{\mathcal{L}_{NS\cap PS}}$	$\text{mIoU}^{PFF}_{\mathcal{L}_{NS\cap PS}}$	$\text{mIoU}^{SP}_{\mathcal{L}_{NS\cap SP}}$	$\text{mIoU}^{W}_{\mathcal{L}_{NS\cap W}}$
CENet [55]	Basée projection	69,1	10,0	49,6	9,8	6,0	3,3	3,5
Helix4D [75]	Basée séquence & 4D	69,3	40,0	44,1	13,6	7,6	45,2	N/A
KPConv [37]	Basée point	63,1	44,9	50,6	25,0	**16,9**	60,7	15,2
SRU-Net [9]	basée voxels	66,3	46,3	52,4	33,1	9,8	61,5	23,6
SPVCNN [69]	basée voxels et point	67,2	**49,4**	**53,2**	**43,7**	11,1	**64,8**	37,2
C3D [65]	basée voxels cylindrique	**70,2**	31,7	46,1	15,8	4,7	42,8	12,7

TABLE 4.4 – Panorama des performances des méthodes de référence entraînées avec nuScenes sous *domain shifts*.

Une fois ce constat réalisé, nous pouvons tout de même nous intéresser aux performances relatives des méthodes pour déceler les tendances et les sensibilités de celles-ci.

A travers les deux tableaux, nous observons une tendance principale qui est la capacité des méthodes basées sur les représentations voxeliques à correctement généraliser lorsque la résolution du capteur est proche (SK → P64, SK → SP). Cylinder3D échappe à ce constat à cause du PointNet appliqué à chaque voxel, poussant le réseau à surapprendre sur les données d'entraînement et donc à fortement diminuer les performances de généralisation.

On note la forte sensibilité à la résolution de la totalité des méthodes, et l'importance de la similarité entre les topologies d'acquisition du jeu de données source et cible (SK → SK32 ainsi que la différence entre NS → SK et NS → SK32).

Sans surprise, les méthodes basées projection ont de faible capacités de généralisation, ce qui est exacerbé lorsque la résolution du capteur change (SK → SK32).

Bien que KPConv semble montrer des difficultés à changer de résolution (SK → SK32), il démontre une certaine résilience au changement de scène, spécifiquement pour les piétons qu'il parvient à reconnaître dans les situations les plus complexes (SK → NS). De plus KPConv, SRU-Net et SPVCNN sont les seuls à parvenir à extraire quelques informations lors d'un changement de capteur (différence entre SK → P64 et SK → PFF).

4.2.5 Résultats sur ParisLuco3D

Les analyses précédentes ont permis d'extraire les tendances principales des différentes architectures vis-à-vis de la généralisation de domaine. Cependant, en raison des concessions faites lors de la définition des jeux de labels utilisés pour l'évaluation, l'analyse s'arrête majoritairement à des observations macroscopiques, d'autant plus dans le cadre de jeux de labels très simples (NS → SP). L'absence de *label shift* entre SemanticKITTI et ParisLuco3D ainsi qu'entre nuScenes et ParisLuco3D permet une analyse fine des résultats par classe.

Dans cette partie, on présente d'abord les résultats de SemanticKITTI vers ParisLuco3D (tableau 4.5). Ce cas peut être considéré comme le plus complexe pour plusieurs raisons. D'abord, le *scene shift* est très fort, entre une zone urbaine dense (ParisLuco3D) et une zone périurbaine (SemanticKITTI), et cela se voit particulièrement dans la quantité des différents objets apparaissant dans la scène (voir chapitre 2). De plus, le capteur est différent. Finalement, le jeu de labels de SemanticKITTI met l'accent sur les objets statiques qui sont très sensibles à l'*appearance shift*.

Globalement, les méthodes basées sur la voxelisation sont les meilleures pour les véhicules et les différents types de sols, cependant on peut noter que KPConv est de loin la méthode la plus efficace pour l'extraction de piétons, tendance qui avait déjà été relevée dans le cas SemanticKITTI vers nuScenes, montrant la résilience à la résolution du capteur pour la segmentation de piétons.

Contrairement à SemanticKITTI, nuScenes est acquis dans une zone urbaine et a le même capteur que ParisLuco3D, réduisant considérablement les *domain shifts*. NuScenes met l'accent sur les véhicules dans la labélisation, regroupant la quasi-totalité de l'arrière-plan en un seul label. Il est cependant important de noter que deux des labels (*cone* et *trailer*) sont peu présents dans ParisLuco3D et ont donc une forte influence sur le mIoU. Bien que la réflectivité renvoyée par les capteurs d'acquisition de nuScenes et de ParisLuco3D soit la même, elle n'a pas été utilisée en tant que caractéristique d'entrée dans la continuité des expériences réalisées précédemment.

Il est important de noter que ce cas d'étude est le seul sans *sensor shift*, ce qui permet à CENet d'être bien plus performant que dans les autres cas de généralisation. A capteur égal, les méthodes basées sur la projection *range* peuvent être envisagées, mais elles sont très sensibles au *sensor shift*, CENet se distinguant notamment sur

ses capacités de compréhension des types de sols. Inversement, les méthodes basées sur la voxelisation sont très pertinentes sur les véhicules.

	Car	Bicycle	Motorcycle	Truck	Other-vehicle	Person	Road	Parking	sidewalk	Other-ground	building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic-sign	mmIoU$_{\mathcal{L}_{SK} \cap \mathcal{L}_{PL3D}}$
CENet [55]	2,2	0	0	0,4	0	0	1,6	0	6,8	0	10,1	13,1	10,6	0	1,1	2,6	1,3	2,9
Helix4D [75]	18,2	0,2	2,1	0,3	4,7	3,0	55,1	2,7	42,7	**2,1**	40,5	22,1	51,0	27,3	7,7	27,9	**35,9**	20,2
KPConv [37]	39,8	7,4	9,1	0,3	5,1	**30,6**	8,1	0,1	41,5	0,7	58,5	11,7	66,9	**49,3**	14,0	25,6	6,8	22,1
SRU-Net [9]	**69,5**	**9,3**	**15,4**	**4,1**	**32,1**	20,5	**71,4**	1,0	66,7	1,5	67,3	18,1	71,4	44,2	15,7	**40,2**	19,0	**33,3**
SPVCNN [69]	66,7	7,0	14,0	4,0	18,9	21,8	66,6	0,2	**67,0**	0,1	66,3	13,0	**71,6**	43,2	10,8	38,3	25,4	31,5
Cylinder3D [65]	46,4	4,6	5,8	0,3	15,2	11,3	58,9	**3,9**	57,2	1,7	65,8	36,6	54,5	24,4	10,8	31,7	3,8	25,5

TABLE 4.5 – Benchmark de généralisation de domaine monosource de SemanticKITTI vers ParisLuco3D.

Méthode	barrier	bicycle	bus	car	cnstrctn-vhcl	motorcycle	pedestrian	traffic-cone	trailer	truck	drvbl-grnd	other-flat	sidewalk	terrain	manmade	vegetation	mIoU$_{\mathcal{L}_{NS} \cap \mathcal{L}_{PL3D}}$
CENet [55]	4,1	**4,7**	35,7	61,6	1,6	22,7	51,6	0	0	6,1	**77,4**	**22,5**	**56,7**	13,3	81,1	66,6	31,6
Helix4D [75]	1,0	0,4	9,2	29,2	0,1	2,4	9,3	0	0	0,1	57,4	5,8	40,8	13,0	71,5	65,3	19,2
KPConv [37]	**4,2**	0,1	2,1	8,3	0,2	1,9	11,4	0	0	1,7	8,3	0	1,7	3,7	73,2	65,7	11,4
SRU-Net [9]	1,8	0,4	48,4	**78,1**	3,9	10,6	51,7	0,3	0	**20,4**	72,7	2,5	47,4	**14,3**	83,5	**83,2**	**32,5**
SPVCNN [69]	1,7	0,8	**49,5**	66,0	**5,4**	**27,1**	**55,9**	0,3	0	15,4	69,3	1,4	42,8	11,8	**84,8**	82,1	32,1
Cylinder3D [65]	0,3	0	5,0	31,5	0,4	0,1	17,4	**1,2**	0	14,0	13,3	0,1	25,5	5,8	77,3	82,4	17,1

TABLE 4.6 – Benchmark de généralisation de domaine monosource de nuScenes vers ParisLuco3D.

4.2.6 Retour sur la réflectivité

Dans la sous-section 4.2.3, nous avons démontré l'intérêt de laisser de côté la caractéristique de réflectivité pour les performances de généralisation. Néanmoins, au vu du panorama qui vient d'être dressé, la caractéristique de réflectivité pour la segmentation monodomaine est très importante. Pour tenter de réunir les performances avec et sans réflectivité, nous proposons une approche d'apprentissage avec *dropout* de la réflectivité. Cette approche consiste à ne pas utiliser la réflectivité dans 50% des nuages utilisés lors de l'entraînement, et l'utiliser pour les autres nuages. Nous utilisons le terme de *dropout* car ce retrait de la réflectivité est réalisé aléatoirement lors de la sélection d'un nuage de points pour être ajouté au batch d'entraînement.

Au moment de l'inférence, la réflectivité est gardée si le capteur d'acquisition est le même que celui utilisé pour le jeu de données de l'entraînement, et elle est retirée

si le capteur est différent. Les résultats et leur comparaison par rapport au cas sans réflectivité sont présentés dans le tableau 4.7. On observe un gain significatif dans le cadre monodomaine et une conservation globale des performances de généralisation avec une forte augmentation dans le cas SK → NS. En contrepartie, on observe une forte baisse de performances dans le cas SK → PFF, ce qui atteste d'une perte de compréhension de la géométrie.

Globalement, il s'agit d'une stratégie envisageable, mais pas strictement meilleure que celle utilisée qui est le retrait de la caractéristique de réflectivité.

Méthode	$\text{mIoU}^{SK}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{SK32}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{P64}_{\mathcal{L}_{SK\cap PS}}$	$\text{mIoU}^{PFF}_{\mathcal{L}_{SK\cap PS}}$	$\text{mIoU}^{SP}_{\mathcal{L}_{SK\cap SP}}$	$\text{mIoU}^{W}_{\mathcal{L}_{SK\cap W}}$	$\text{mIoU}^{NS}_{\mathcal{L}_{SK\cap NS}}$	$\text{mIoU}^{PL3D}_{\mathcal{L}_{SK\cap PL3D}}$
SRU-Net [9]	58,6	54,0	44,2	**22,2**	45,3	**33,1**	42,7	33,1
SRU-Net avec r[9]	**61,9**	57,0	**44,7**	16,5	45,2	26,0	39,6	30,7
SRU-Net avec *dropout* [9]	61,4	**57,4**	43,3	17,2	**46,4**	32,2	**45,4**	**33,9**

TABLE 4.7 – Comparaison des performances de généralisation sans réflectivité et avec la méthode de *dropout*.

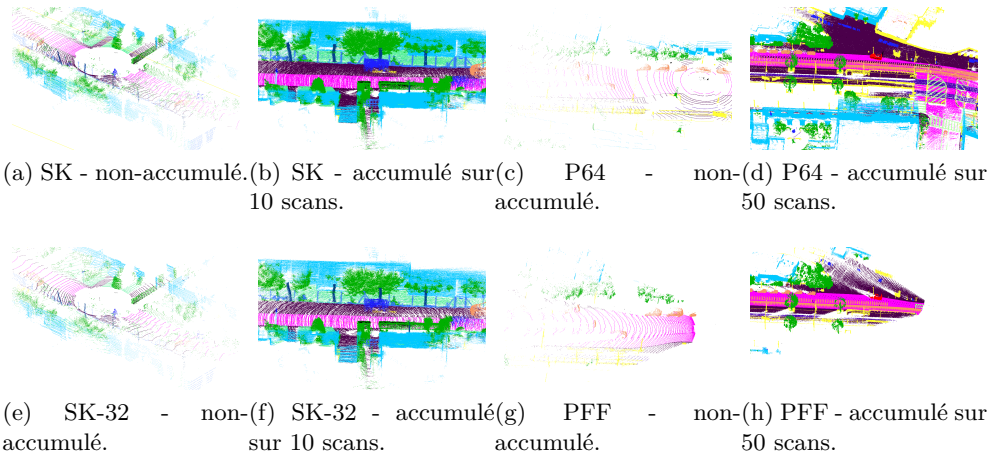
4.3 Robustesse au *sensor shift* - 3DLabelProp

4.3.1 Observations précédentes et intuitions

Comme vu dans la section précédente, la première cause de baisse de performance pour la généralisation de domaine est la sensibilité au *sensor shift*. Le changement de résolution cause une baisse de performance conséquente, et le changement de technologie d'acquisition rend les scènes incompréhensibles. Les stratégies existantes de généralisation de domaine s'intéressent d'ailleurs particulièrement à insensibiliser les méthodes au *sensor shift*.

Se basant sur les travaux de Complete & Label [85], la piste de l'existence d'un domaine canonique sera suivie. Le domaine canonique permettrait de représenter une scène indépendamment du capteur d'acquistion ; le *sensor shift* y serait donc inexistant. Pour cela, au lieu de travailler sur chaque nuage de points acquis indépendamment, comme le fait la majorité de l'état de l'art, on souhaite exploiter la séquentialité des jeux de données et travailler sur des nuages de points **pseudo-denses** obtenus par l'accumulation de cette séquence. Un exemple d'une telle accumulation peut être vu sur la figure 4.10, qui illustre aussi le rapprochement de domaine pour des résolutions de capteurs différentes.

On suppose que travailler dans le domaine pseudo-dense réduit le *sensor shift* et permet donc de meilleurs résultats de généralisation. Le contrecoup de travailler



(a) SK - non-accumulé. (b) SK - accumulé sur 10 scans. (c) P64 - non-accumulé. (d) P64 - accumulé sur 50 scans.

(e) SK-32 - non-accumulé. (f) SK-32 - accumulé sur 10 scans. (g) PFF - non-accumulé. (h) PFF - accumulé sur 50 scans.

FIGURE 4.10 – Comparaison entre SemanticKITTI, SemanticKITTI-32, Panda64 et PandaFF, accumulés ou non.

sur des nuages pseudo-denses est l'augmentation du nombre de points dans le nuage et donc l'augmentation du temps de calcul nécessaire pour le traiter. Il sera donc nécessaire de mettre au point des stratégies d'accélération de la segmentation, pour cela on s'intéressera aux stratégies utilisées en vision 2D et plus précisément dans le traitement de flux caméra.

4.3.2 Segmentation sémantique de flux caméra

Contrairement à la segmentation sémantique 3D pour le véhicule autonome, la segmentation 2D atteint des seuils qualitatifs bien plus hauts (jusqu'à 86% mIoU sur les 19 labels de Cityscape, jeu de données de référence pour la tâche). Ces méthodes incluent PSPNet [114], SegFix [115] ou Hierarchical multiscale attention network [116]. Bien que la qualité de la segmentation soit suffisante, la vitesse de traitement est trop lente pour traiter un flux vidéo en temps réel (60Hz minimum).

Pour améliorer cette vitesse d'exécution, des méthodes comme ICNet [117] se concentrent sur l'accélération du traitement d'une seule image. D'autres méthodes prennent en compte la structure du flux vidéo et la persistance de l'information d'une image à l'autre pour améliorer la vitesse de traitement globale.

Pour le court panorama suivant, seules les méthodes n'utilisant pas les images dans le futur par rapport à l'image considérée seront étudiées, pour se placer dans le cas du traitement d'un flux vidéo pour le véhicule autonome.

Une des premières méthodes prenant en compte le flux vidéo est Clockwork Convnet [118] et se base sur l'idée qu'entre deux images consécutives au sein d'un flux vidéo, les cartes de vecteurs caractéristiques dans le réseau de neurones changent à une vitesse différente selon leur profondeur dans le réseau. D'après cette observation,

entre deux images consécutives, certaines parties du réseau n'ont pas besoin d'être recalculées ce qui entraîne une réduction du temps de traitement.

Cependant, l'approche principale est l'utilisation du flot optique, comme dans DFF [119], pour propager les cartes de vecteurs caractéristiques d'une image à l'autre. La méthode est présentée sur la figure 4.11. L'idée sous-jacente est que lors de la segmentation d'images, on peut visualiser une architecture de segmentation en deux sous-réseaux : un premier lent, profond et large qui est l'*encoder* qui permet d'extraire les vecteurs de caractéristiques, et un second rapide et léger qui s'occupe de réaliser la tâche, ici la segmentation. S'appuyant sur cette idée, et dans la continuité de Clockwork Convnet, on souhaite minimiser le nombre d'appels au premier sous-réseau qui est lent, et on va utiliser le flot optique pour transférer les caractéristiques d'une image à l'autre. Ces caractéristiques translatées peuvent ensuite être transmises au deuxième sous-réseau. Cette méthode s'appuie sur le fait que le calcul de flot optique est extrêmement rapide.

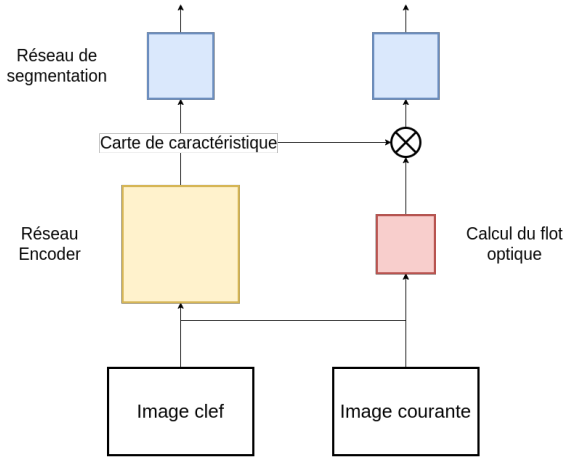


FIGURE 4.11 – Illustration de DFF [119].

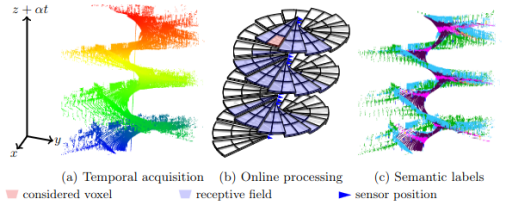
On peut noter deux améliorations depuis DFF. DSVNet [120] traite les quartiers d'image indépendamment et apprend un réseau de décision pour savoir quand le premier sous-réseau doit être appliqué sur chaque quartier. Le Accel [121] ajoute un *update network* pour aider le flot optique à repositionner les caractéristiques dans les images suivantes.

4.3.3 Méthodes basées sur les séquences

Les données LiDAR pour le véhicule autonome sont analogues à un flux vidéo. Elles se succèdent à une vitesse d'acquisition relativement élevée et forment une séquence. Cette information séquentielle est majoritairement ignorée par les méthodes de segmentation sémantique LiDAR.

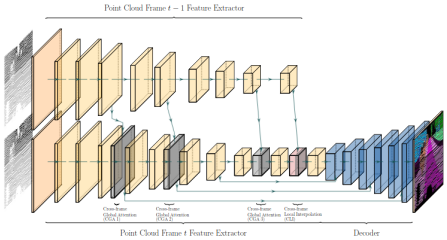
Cependant, certaines méthodes incorporent la temporalité dans le traitement des caractéristiques. ASAP-Net [122] sépare explicitement les interactions temporelles et spatiales en introduisant de l'attention temporelle grâce aux corrélations spatio-temporelles entre des frames consécutives. SpSequenceNet [123] concatène des cartes de vecteurs caractéristiques des nuages de points précédents avec celui considéré. [124] utilise un RNN (Recurrent Neural Network) pour garder en mémoire des informations sur les caractéristiques des nuages précédents pour les intégrer dans le traitement du nuage actuel.

Une autre vague de méthodes utilise la séquentialité en amont du traitement des caractéristiques, dans la géométrie du nuage de points. Ces stratégies ont notamment été employées dans la segmentation d'objets dynamiques avec [125] qui utilise des convolutions 4D après avoir remis les nuages des instants précédents dans le référentiel du nuage courant. D'autres méthodes comme MeteorNet [126] et PSTNet [127] s'intéressent à la création de modules d'extraction de caractéristiques 4D en redéfinissant des convolutions par points adaptées. Helix4D [75] visualise la séquence comme un hypercylindre et traite les nuages de points dans ce nouvel espace de représentation. Finalement, [128] et [129] voient la séquentialité comme une information purement géométrique, remettent la séquence dans un référentiel commun et traitent le nuage de points partiellement accumulé grâce à des méthodes 3D. Une illustration de différentes stratégies se trouve dans la figure 4.12.



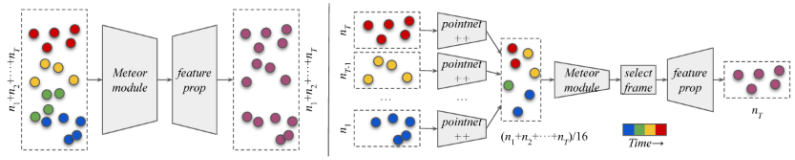
Online LiDAR Segmentation. The 3D point sequences of rotating LiDAR data of our proposed dataset HelixNet follow a complex helix-like structure in space and time, represented in a by using the vertical axis for both time and elevation. We propose an efficient spatio-temporal transformer to process angular slices of data centered on the sensor's position. The slices are partitioned into voxels, each attending other voxels from past slices to build a large spatio-temporal receptive field b. Our proposed model can segment the LiDAR point stream c with state-of-the-art accuracy and in real-time.

(a) Helix4D [75].



Sparse sequence network structure. The input data is the point cloud frames P_{t-1} and P_t. The output is the semantic label for P_t. In addition, we use colors to represent the different functions. The yellow blocks are the basic neural network blocks, which is a 3D residual network. Grey blocks are the Cross-frame Global Attention (CGA) modules, which is designed to fuse the comprehensive information from the last frame. The red blocks is the Cross-frame Local Interpolation (CLI) module, which is proposed to combine the local information from previous frames and current frames. The Blue blocks are the decoder modules for segmentation outputs respectively.

(b) SpSequenceNet [123].



Architecture design choices for MeteorNet. Left: Early fusion with per-point output for all frames. Right: Late fusion with per-point output for the last frame.

(c) MeteorNet [126].

FIGURE 4.12 – Visualisation de quelques méthodes de traitement de séquence de nuages de points.

4.3.4 KPConv et SRU-Net

Au vu des résultats de généralisation de la section précédente, une architecture de segmentation sémantique sort du lot : SRU-Net. Il s'agit de l'une des architectures ayant nativement les meilleures facultés de généralisation (voir tableau 4.3). De plus, comme présenté dans le chapitre 2, SRU-Net sert de *backbone* à de nombreuses autres architectures de segmentation et est l'architecture de référence. Son fonctionnement a été présenté dans la sous-section 2.1.2.

Pour tester notre intuition initiale, qui est qu'utiliser des nuages de points pseudo-denses améliorerait les performances de généralisation, un SRU-Net utilisant les nuages pseudo-denses est entraîné. En complément, un KPConv exploitant aussi les nuages pseudos-denses est entraîné. Ce choix est fait car KPConv est une des meilleures architectures de segmentation des nuages denses, et une architecture de généralisation de scans LiDAR moyenne, ce qui permettra d'étudier plus précisément l'effet des nuages pseudo-denses. Le protocole consiste à créer un nuage pseudo-dense en utilisant les 10 acquisitions précédentes à celle considérée recalées grâce à une méthode de SLAM, précisément CT-ICP [84]. La comparaison entre les résultats lorsque les modèles sont entraînés avec un seul nuage de points et les modèles entraînés avec 10 sont disponibles dans le tableau 4.8. Les résultats de KPConv pseudo-dense ne sont pas disponibles pour Waymo à cause du format des données, pour lesquelles nous disposons des scans dans le référentiel global, mais pas des positions individuelles. Les résultats de SRU-Net pseudo-dense ne sont pas disponibles pour Panda64 à cause du coût mémoire prohibitif qui empêche l'inférence.

Méthode	$\text{mIoU}^{SK}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{SK32}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{P64}_{\mathcal{L}_{SK\cap PS}}$	$\text{mIoU}^{PFF}_{\mathcal{L}_{SK\cap PS}}$	$\text{mIoU}^{SP}_{\mathcal{L}_{SK\cap SP}}$	$\text{mIoU}^{W}_{\mathcal{L}_{SK\cap W}}$	$\text{mIoU}^{NS}_{\mathcal{L}_{SK\cap NS}}$	$\text{mIoU}^{PL3D}_{\mathcal{L}_{SK\cap PL3D}}$
KPConv [37]	58,3	52,7	32,7	21,1	39,1	17,5	46,7	22,1
KPConv pseudo-dense [37]	53,6	53,2	47,8	45,0	47,5	N/A	46,1	30,5
SRU-Net [9]	58,6	54,0	44,2	22,2	45,3	33,1	42,7	33,3
SRU-Net pseudo-dense [9]	56,7	57,3	N/A	61,7	46,9	25,6	49,8	38,0

TABLE 4.8 – Impact de l'accumulation sur les performances de généralisation de SRU-Net et KPConv.

On observe une nette amélioration des performances de généralisation dans la plupart des cas, et cela pour les deux modèles. Cependant, cette amélioration vient au détriment de deux éléments, les résultats monodomaines et la vitesse d'exécution. Notamment, ce sont les premières méthodes à être capables d'extraire de l'information de PandaFF, alors que les méthodes n'utilisant qu'un seul nuage de points en sont incapables.

Les informations de vitesse d'exécution peuvent être trouvées dans le tableau 4.9 ; on y voit que les performances en termes de vitesse d'inférence passent de bonnes, ou correctes, à désastreuses. Il est donc nécessaire pour tirer parti de la faculté des nuages pseudo-denses à permettre de mieux généraliser de le faire de manière optimisée. La situation est donc similaire à celle de la segmentation de flux vidéo. Les capacités de segmentation multidomaines sont satisfaisantes, mais la vitesse d'exécution ainsi que les performances monodomaine sont insuffisantes.

Méthode	Vitesse d'inférence SemanticKITTI (Hz)	Vitesse d'inférence nuScenes (Hz)
KPConv [37]	0,6	1,3
KPConv pseudo-dense [37]	0,1	0,3
SRU-Net [9]	6,7	7,7
SRU-Net pseudo-dense [9]	1,1	4

TABLE 4.9 – Vitesse d'exécution de KPConv et SRU-Net.

En plus d'une augmentation du temps de traitement, les méthodes qui utilisent des nuages pseudo-denses sont aussi dépendantes de la qualité de l'estimation des trajectoires. Dans le tableau 4.10, un SRU-Net pseudo-dense est entraîné, et évalué, avec des positions artificiellement bruitées pour simuler une mauvaise estimation de trajectoire. On voit une perte systématique de performances, accentuée pour les jeux de données acquis avec un capteur à faible résolution (nuScenes, ParisLuco3D et SemanticKITTI32). Dans la suite de ce travail, nous utiliserons des trajectoires estimées avec précision.

Méthode	$\text{mIoU}^{\text{SK}}_{\mathcal{L}^{\text{SK}}}$	$\text{mIoU}^{\text{SK32}}_{\mathcal{L}^{\text{SK}}}$	$\text{mIoU}^{\text{PFF}}_{\mathcal{L}^{\text{SK}\cap\text{PS}}}$	$\text{mIoU}^{\text{SP}}_{\mathcal{L}^{\text{SK}\cap\text{SP}}}$	$\text{mIoU}^{\text{NS}}_{\mathcal{L}^{\text{SK}\cap\text{NS}}}$	$\text{mIoU}^{\text{PL3D}}_{\mathcal{L}^{\text{SK}\cap\text{PL3D}}}$
SRU-Net pseudo-dense [9]	56,7	57,3	61,7	46,9	49,8	38,0
SRU-Net pseudo-dense perturbé [9]	45,1	41,5	58,5	39,9	36,3	30,1

TABLE 4.10 – Impact de l'introduction de perturbations dans les poses SLAM sur les performances de généralisation de SRU-Net pseudo-dense.

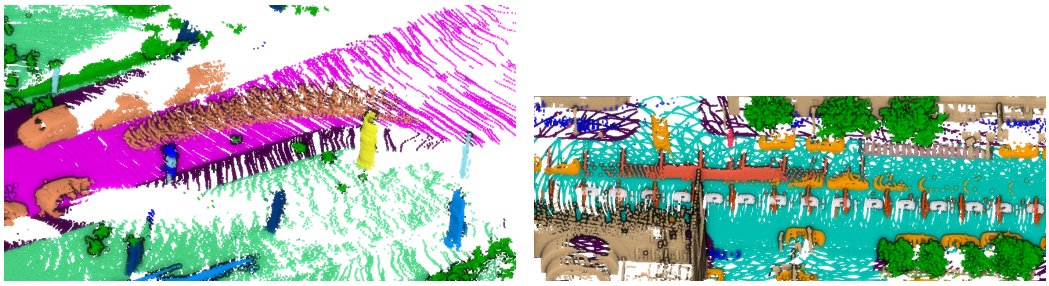
4.3.5 3DLabelProp

A partir de ces différents constats, je propose ici une nouvelle méthode de généralisation monosource pour la segmentation sémantique de données LiDAR pour

le véhicule autonome, appelée **3DLabelProp**. 3DLabelProp est classée dans les méthodes d'alignement de domaine, car en s'inspirant des méthodes de segmentation pseudo-dense, elle cherche à représenter les données dans un domaine commun pour réaliser la segmentation. De plus, 3DlabelProp est inspiré de la segmentation de flux vidéo et distingue des zones simples à segmenter et des zones compliquées. Les zones simples peuvent être segmentées grâce à la géométrie de la scène, les méthodes géométriques étant très rapides, et les zones complexes sont isolées dans de petits sous-nuages de points pour accélérer l'exécution des méthodes d'apprentissage profond.

Globalement, 3DLabelProp n'est pas conçue pour atteindre les contraintes de vitesse d'exécution de la perception pour le véhicule autonome. 3DLabelProp a pour objectif de maximiser les performances de généralisation, même si cela se fait au détriment de la vitesse. Nous considérons tout de même 3DLabelProp comme digne d'être réalisée et étudiée pour comprendre les interactions entre la représentation de la donnée d'entrée et les performances de généralisation, ainsi que pour donner des bornes atteignables de performances de généralisation pour les futurs travaux.

Intuitivement, les zones simples correspondent aux objets statiques. Comme ces objets ne bougent pas dans le référentiel global, il y a de grandes chances que deux scans consécutifs de la scène échantillonnent des points voisins sur ces objets. On peut donc utiliser les inférences passées pour prédire les points nouvellement acquis par propagation des plus proches voisins. Les zones complexes, elles, correspondent aux objets mobiles et aux objets nouvellement échantillonnés. Ces derniers n'ont pas de voisins dans les acquisitions passées, et les objets dynamiques n'en ont théoriquement pas. S'ils en ont, à cause des phénomènes de traînées, il n'y a pas de garantie que des points voisins dans l'espace 3D appartiennent au même objet. On illustre le problème des traînées sur la figure 4.13. À droite, on observe la trainée laissée par un véhicule au cours de 5 scans consécutifs, à gauche, on observe la superposition de traînées de nombreux véhicules sur une plage de temps plus longue.



(a) traînées laissées par une voiture. (b) Trainée laissée par une voiture et un bus.

FIGURE 4.13 – Illustration du phénomène de traînées.

On décompose 3DLabelProp en 6 étapes :

1. Recalage du nuage nouvellement acquis dans un nuage de points de référence,

2. Propagation des labels préalablement segmentés aux objets statiques de la scène nouvellement échantillonnée,
3. Isolement des zones complexes en N sous-nuages de points,
4. Enrichissement des sous-nuages de points avec les points avoisinants dans le nuage de points de référence,
5. Segmentation des sous-nuages enrichis grâce à des méthodes d'apprentissage profond,
6. Fusion des prédictions.

L'étape (1) n'est pas traitée dans ce travail et est supposée réalisée avec un SLAM LiDAR comme CT-ICP [84]. Le nuage de points de référence correspond aux acquisitions passées et mises dans le même référentiel, il s'agit d'un nuage pseudo-dense. De plus, il y a aussi une étape de réduction de l'empreinte mémoire, le nuage de points de référence est sous-échantillonné sur une grille d'une résolution de 5 cm, et les points au-delà d'une certaine distance (ici 75 m) sont retirés. Cette étape est incluse dans le recalage.

L'intégralité du code est disponible en ligne sur
`https://github.com/JulesSanchez/3DLabelProp`.

Détaillons les étapes suivantes.

A. Propagation des labels

Tout d'abord, il est nécessaire de séparer le jeu de labels, $L = (l_i)_{i\in\{0,K\}}$, en deux parties correspondant aux labels des objets statiques, $S = (s_i)_{i\in\{K_d,K\}}$, et ceux correspondant aux objets dynamiques $D = (d_i)_{i\in\{0,K_d\}}$. Il n'y a pas d'intersection entre ces sous-jeux de labels et $L = D \cup S$. Les objets statiques sont définis comme les objets ne pouvant pas bouger, ce qui comprend notamment le sol et les bâtiments. Les objets dynamiques sont les éléments en mouvement, ou pouvant le devenir, ce qui correspond majoritairement aux piétons et aux véhicules.

Par définition, les objets sémantisés comme appartenant à un label statique ne bougent pas dans le référentiel monde d'une acquisition à l'autre. En définissant un voisinage suffisamment petit, on peut donc supposer que deux points voisins appartiennent au même objet. En pratique pour un rayon de 10 cm, la classe majoritaire dans le voisinage d'un point nouvellement acquis dans plus de 95% des cas sa classe réelle.

Pour réaliser la propagation, deux étapes sont nécessaires : extraire le voisinage de chaque point, et assigner un label et un score de confiance à chaque point selon ses voisins.

Le voisinage est extrait par un algorithme de recherche par rayon, accéléré par la voxelisation du nuage de points qui permet d'extraire un pré-voisinage grossier de chaque point. Ensuite une étape de vote a lieu, les votes étant pondérés par la distance du voisin au point considéré et par le score de confiance du voisin. Si la

pondération tombe en dessous d'une certaine valeur, le voisin est ignoré, car il est considéré comme non fiable.

Si le label assigné est un label dynamique, alors aucun label n'est assigné, car il est supposé impossible de propager un label dynamique. Sinon c'est le résultat du vote qui est assigné, avec un score de confiance qui est la moyenne pondérée par la distance des scores de confiance des voisins ayant voté pour ce label. Une illustration de la méthode est proposée sur la figure 4.14.

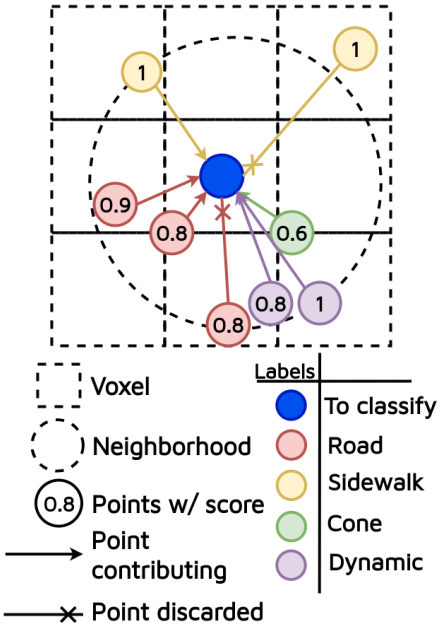


FIGURE 4.14 – Module de propagation de label.

A une étape quelconque, on peut noter $\mathcal{P}_{ref} \in \mathbb{R}^{N\times 3}$ le nuage de point de référence, $Y_{ref} \in \{0, K\}^N$ les labels sémantisés associés et $\mathcal{C}_{ref} = (c_{ref,i} \times e_{y_i})_{i\in\{1,N\}}$ le score de confiance pour chaque point, qui est un vecteur indicateur d'un label (e_{y_i}) multiplé par le score de confiance pour le label associé $c_{ref,i}$. De la même façon, on note $\mathcal{P}_{curr} \in \mathbb{R}^{M\times 3}$, Y_{curr} et $\mathcal{C}_{curr}$ les valeurs correspondantes au nuage de points nouvellement acquis. Initialement, tous les labels sont initialisés à -1 et leur score de confiance associé à 0, on peut alors effectuer la propagation :

$$\forall p_{curr,i}, i \in \{1, M\};$$

$$c_{curr,i} = \sum_{p_{ref,n}\in\mathcal{N}(p_{curr,i})} w(i,n)\mathbb{1}_{w(i,n)>0,5}; \tag{4.1}$$

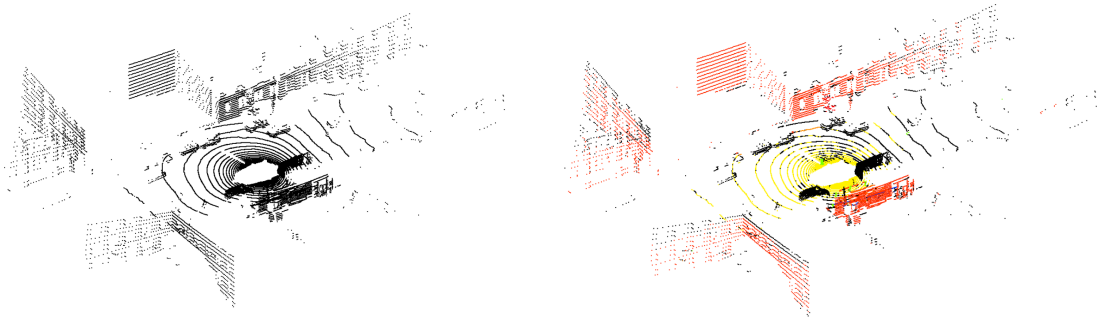
avec $w(i,n) = d(p_{ref,n}, p_{curr,i}) \times c_{ref,i}$ et $d(p,q) = e^{-\frac{||p-q||^2}{\sigma^2}}$, où σ est le paramètre de bande passante qui permet de définir la portée de la recherche de voisin. $\mathcal{N}(p_{curr,i})$ est le voisinage de $p_{curr,i}$.

On a alors :

$$y_{curr,i} = \begin{cases} \arg\max(c_{curr,i}), & \text{si} \quad \arg\max(c_{curr,i}) > K_d \\ -1, & \text{sinon} \end{cases} \tag{4.2}$$

avec $\arg\max(c_{curr,i}) > K_d$, signifiant que c'est un label statique (K_d étant le nombre de labels dynamiques).

L'effet de la propagation est illustré en pratique sur un nuage de points de SemanticKITTI sur la figure 4.15. Pour l'illustration, les labels appliqués sur le nuage de référence sont les labels de la vérité terrain.



(a) Nuage en entrée de la propagation.

(b) Nuage en sortie de la propagation.

FIGURE 4.15 – Effet de la propagation sur un nuage de points avec en couleur les points sémantisés et en noir les points résiduels.

B. Création des sous-nuages de points

Le problème majeur soulevé auparavant est la difficulté de traiter en un temps convenable un nuage de points pseudo-dense ; or cette pseudo-densité est justement recherchée pour enrichir les voisinages locaux et harmoniser les représentations des objets, quel que soit le capteur d'acquisition.

Pour réduire le problème, la stratégie est d'extraire des sous-nuages de points de taille réduite du nuage originel, et de les traiter en parallèle. Certaines méthodes utilisent déjà une stratégie similaire comme KPConv. Dans ce cas, les sous-nuages sont obtenus par extraction de sphères au sein du nuage, cependant il est nécessaire de créer une très grande quantité de sphères pour couvrir intégralement le nuage et cela résulte en une grande quantité de superposition. Cette superposition est d'ailleurs cherchée par les auteurs pour rendre leur segmentation plus robuste. Dans la méthode 3DLabelProp, la vitesse d'exécution est favorisée et le choix d'utiliser un algorithme de *clustering* est fait pour couvrir plus efficacement le nuage.

Etant donné qu'il n'est pas nécessaire de couvrir l'intégralité du nuage de points, mais uniquement les parties restantes après la phase de propagation, l'algorithme de

clustering peut être appliqué seulement sur les $p_{curr,i}$, où $y_{curr,i} = -1$. Cet ensemble de points est bien plus petit que le nuage initial, et cela permet d'appliquer des algorithmes de clustering sophistiqués à une très grande vitesse. Le choix se porte sur l'algorithme des K-means. Un exemple de points résiduels suite à la phase de propagation et un des sous-nuages extraits se trouvent sur la figure 4.16.

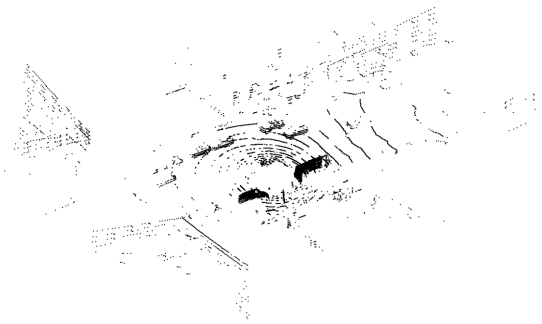


FIGURE 4.16 – Exemple de sous-nuage épars, résidu de la propagation restant à traiter.

C. Enrichissement des sous-nuages de points

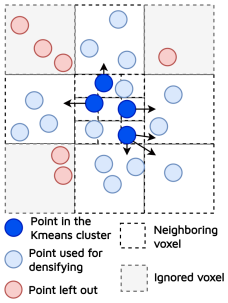
A la suite de l'étape d'extraction des sous-nuages, résidus de l'étape de propagation, ceux-ci sont presque dépourvus de contexte. Pour en améliorer le contenu, ils sont enrichis grâce aux points appartenant au nuage de référence. Pour cela, l'espace est voxélisé et les points du nuage de référence (c'est-à-dire le nuage formé par l'accumulation des nuages précédents) se situant dans les voxels occupés par les points du sous-nuage considéré sont ajoutés. De plus, les points se situant dans des voxels avoisinants sont aussi ajoutés.

Cependant, la totalité des voxels avoisinants n'est pas prise en compte, afin de réduire le plus possible la taille des sous-nuages. Pour ce faire, les voxels occupés par le sous-nuage considéré sont recoupés en 27 (3x3x3) et pour chaque morceau de voxel, un voisinage spécifique est associé. Seuls les voisins assignés aux morceaux de sous-voxels occupés sont utilisés pour l'enrichissement. Le processus est illustré en 2D sur la figure 4.17.

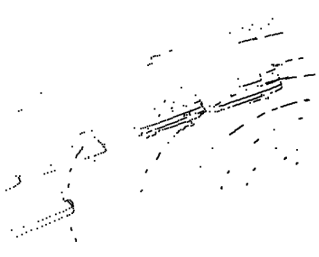
D. Segmentation des sous-nuages de points enrichis

Une fois les sous-nuages enrichis obtenus, ils sont traités par un algorithme d'apprentissage profond. Cet algorithme doit, de préférence, tirer profit des voisinages très denses de ces sous-nuages. Comme les sous-nuages ne respectent plus les topologies d'acquisition habituelles des LiDAR rotatifs, les méthodes basées sur les projections sphériques ne peuvent pas être employées.

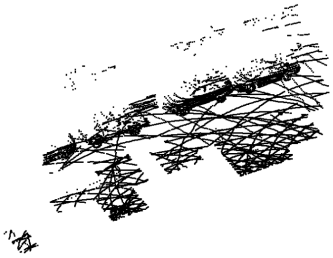
Dans ce travail, KPConv est l'algorithme choisi. D'une part, cette méthode a déjà des performances correctes de généralisation, et d'autre part, il s'agit d'une des



(a) Module.



(b) Sous-nuage épars.



(c) Sous-nuage dense.

FIGURE 4.17 – Module d'enrichissement des sous-nuages.

méthodes de référence pour le traitement des nuages de points denses. KPConv est ré-entraîné spécifiquement avec les sous-nuages et est ensuite appliqué tel quel au moment de l'inférence.

E. Fusion des prédictions

Certains points du nuage segmenté final peuvent avoir plusieurs prédictions. Déjà, en raison de la phase d'enrichissement des sous-nuages, certains points peuvent appartenir à plusieurs sous-nuages à la fois. De plus, certains de ces points ont aussi déjà eu une prédiction initiale par propagation. Pour combiner ces différentes prédictions, une moyenne des scores de confiance prédits par classe par chacun des modules est faite, et l'argmax est pris comme prédiction finale.

Un schéma récapitulatif du fonctionnement de 3DLabelProp est proposé sur la figure 4.18. Les points en couleurs correspondent aux labels prédits, le nuage de points bleu et rouge correspond à la visualisation des erreurs (en rouge) commises.

4.3.6 Facultés de généralisation

L'intégralité des expériences de la section 4.2 est reproduite sur 3DLabelProp. Dans les tableaux de résultats suivants, seuls les résultats de KPConv sont reportés avec 3DLabelProp pour améliorer la lisibilité. De plus avec SemanticKITTI en

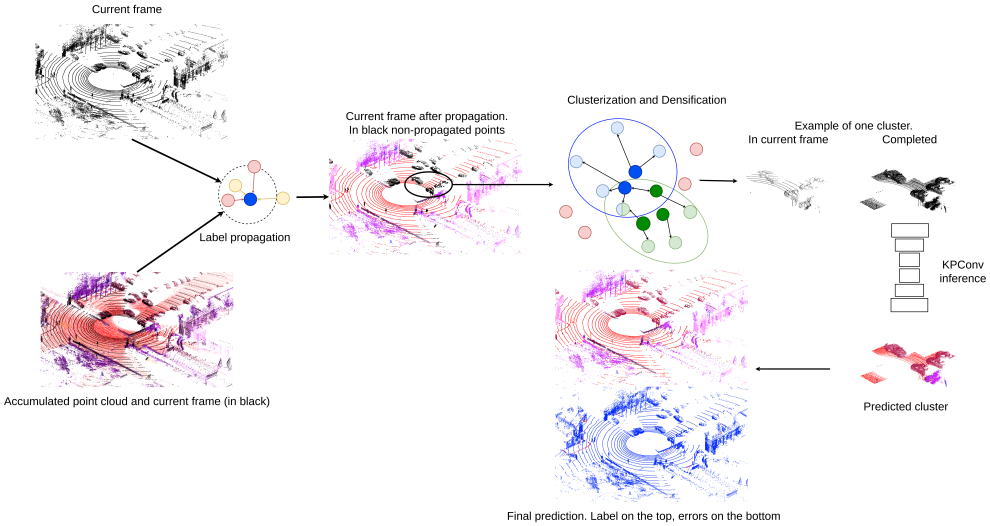


FIGURE 4.18 – Schéma du fonctionnement de 3DLabelProp, sur un nuage de points de SemanticPOSS.

source, on ajoute KPConv pseudo-dense, considéré comme la *baseline* de notre méthode. Une ligne fictive "BestOf" est ajoutée, qui représente les meilleures performances atteintes dans le benchmark de généralisation (tableau 4.3 et tableau 4.4), indépendamment de la méthode. Lors des comparaisons sur ParisLuco3D, "BestOf" correspond au meilleur résultat par label.

Globalement, comme les autres méthodes pseudo-denses, 3DLabelProp est une méthode de segmentation très résiliente aux *sensor shifts*, mais qui reste pertinente dans les autres cas de *domain shifts*. De plus, contrairement aux méthodes d'exploitation naïve des nuages pseudo-denses, 3DLabelProp arrive à des résultats de segmentation monodomaine satisfaisants, ce qui en fait la meilleure méthode pour généraliser.

Pour le cas avec SemanticKITTI en source, présenté dans le tableau 4.11, on observe que 3DLabelProp est l'architecture la plus performante en généralisation pour la vaste majorité des cas. De plus, dans le cas monodomaine (SK→SK), 3DLabelProp est très compétitif, à moins de 0,5% de mIoU des autres méthodes.

Pour l'expérimentation de résilience au *sensor shift* (SK→SK32), 3DLabelProp démontre les meilleurs résultats en termes de mIoU (+4,1%), mais aussi en termes d'écart aux performances monodomaine (-0,2%). De plus, 3DLabelProp démontre le plus de proximité entre les performances sur Panda64 (+13,1%) et PandaFF (+37,1%), en plus d'atteindre les meilleures performances pour ces jeux de données. On voit que 3DLabelProp, comme on l'avait vu pour les autres méthodes pseudo-denses (tableau 4.8), parvient à extraire de l'information de PandaFF. 3DLabelProp atteint aussi les meilleures performances de généralisation sur SemanticPOSS (+1,8%) et Waymo (+6,3%). L'unique cas de généralisation où 3DLabelProp est

battu est sur nuScenes (-1,1%) par KPConv (sur nuages épars). L'analyse des résultats par classe explique que cela provient de la grande capacité de KPConv à segmenter les piétons dans le cas multidomaine.

Méthode	$\text{mIoU}^{SK}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{SK32}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{P64}_{\mathcal{L}_{SK\cap PS}}$	$\text{mIoU}^{PFF}_{\mathcal{L}_{SK\cap PS}}$	$\text{mIoU}^{SP}_{\mathcal{L}_{SK\cap SP}}$	$\text{mIoU}^{W}_{\mathcal{L}_{SK\cap W}}$	$\text{mIoU}^{NS}_{\mathcal{L}_{SK\cap NS}}$
KPConv [37]	58,3	52,7	32,7	21,1	39,1	17,5	46,7
KPConv pseudo-dense [37]	53,6	53,2	47,8	45,0	**47,5**	N/A	46,1
BestOf	**62,3**	57,4	44,2	22,2	45,4	33,1	**46,7**
3DLabelProp	61,9	**61,7**	**57,3**	**59,3**	47,2	**39,4**	45,6

TABLE 4.11 – Performances de généralisation de 3DLabelProp entraîné sur SemanticKITTI.

Pour le cas avec nuScenes en source, tableau 4.12, on observe un schéma de résultats similaire au cas avec SemanticKITTI en source. 3DLabelProp domine sur la majorité des cas de généralisation, ici à l'exception, marginalement, de SemanticPOSS (-0,2%). On observe encore une fois la sensibilité minimale de 3DLabelProp au *sensor shift* dans la proximité des performances sur SemanticKITTI et SemanticKITTI-32 et entre Panda64 et PandaFF. Il est notable que dans le cas monodomaine pour nuScenes, 3DLabelProp a les meilleurs résultats (+1,3%).

Méthode	$\text{mIoU}^{NS}_{\mathcal{L}_{NS}}$	$\text{mIoU}^{SK}_{\mathcal{L}_{NS\cap SK}}$	$\text{mIoU}^{SK32}_{\mathcal{L}_{NS\cap SK}}$	$\text{mIoU}^{P64}_{\mathcal{L}_{NS\cap PS}}$	$\text{mIoU}^{PFF}_{\mathcal{L}_{NS\cap PS}}$	$\text{mIoU}^{SP}_{\mathcal{L}_{NS\cap SP}}$	$\text{mIoU}^{W}_{\mathcal{L}_{NS\cap W}}$
KPConv [37]	63,1	44,9	50,6	25,0	16,9	60,7	15,2
BestOf	70,2	49,4	53,2	43,7	16,9	**64,8**	37,2
3DLabelProp	**71,5**	**59,8**	**62,1**	**66,2**	**70,0**	64,6	**47,1**

TABLE 4.12 – Performances de généralisation de 3DLabelProp entraîné sur nuScenes.

Avec ParisLuco3D en cible, tableau 4.13 et tableau 4.14, 3DLabelProp surpasse toutes les méthodes classiques, de +2,6% avec SemanticKITTI en source et de +11,0% avec nuScenes en source. ParisLuco3D étant acquis avec le même capteur que nuScenes, la réflectivité pourrait être employée pour améliorer les performances de généralisation. Cela n'est pas possible avec notre méthode 3DLabelProp : la ré-

flectivité est dépendante de l'angle d'incidence avec les objets, et donc dans un nuage pseudo-accumulé la réflectivité est très bruitée et inexploitable. Cependant, elle est utilisable pour les méthodes travaillant sur un seul nuage de points.

Dans le cas d'utilisation de la réflectivité, SPVCNN obtient de meilleurs résultats que ceux du tableau 4.6 (NS→PL3D) et atteint 38,7% mIoU mais reste derrière 3DLabelProp (43,5%) qui n'utilise pas la réflectivité. Dans les deux cas, on observe notamment un grand gain de performances sur les véhicules deux roues ainsi que sur le terrain. Sur nuScenes le gain de performance a lieu sur la quasi-totalité des catégories. Dans le cas de jeu de labels exhaustif offert par ParisLuco3D, on peut dénoter que 3DLabelProp a une compréhension beaucoup plus fine de la géométrie de la scène.

	Car	Bicycle	Motorcycle	Truck	Other-vehicle	Person	Road	Parking	sidewalk	Other-ground	building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic-sign	$\text{mIoU}^{PL3D}_{LSK \cap PL3D}$
KPConv [37]	39,8	7,4	9,1	0,3	5,1	30,6	8,1	0,1	41,5	0,7	58,5	11,7	66,9	49,3	14,0	25,6	6,8	22,1
KPConv pseudo-dense [37]	61,5	10,8	**25,8**	0,4	25,5	28,2	62,8	1,6	50,4	0,6	64,6	8,9	68,1	37,6	17,0	35,9	19,7	30,5
BestOf	**69,5**	9,3	15,4	**4,1**	**32,1**	**30,6**	**71,4**	**3,9**	**67,0**	**2,1**	67,3	**36,6**	**71,6**	49,3	15,7	**40,2**	35,9	33,3
3DLabelProp	64,5	**14,1**	25,7	3,1	27,1	29,6	70,3	2,7	57,9	0,2	**72,1**	17,9	70,6	**50,5**	**27,5**	38,8	**37,1**	**35,9**

TABLE 4.13 – Résultat de généralisation de 3DLabelProp de SemanticKITTI vers ParisLuco3D.

Méthode	barrier	bicycle	bus	car	cnstrctn-vhcl	motorcycle	pedestrian	traffic-cone	trailer	truck	drvbl-grnd	other-flat	sidewalk	terrain	manmade	vegetation	$\text{mIoU}^{PL3D}_{LNS \cap PL3D}$
KPConv [37]	4,2	0,1	2,1	8,3	0,2	1,9	11,4	0	0	1,7	8,3	0	1,7	3,7	73,2	65,7	11,4
SPVCNN [69] avec réfléctivité	6,5	4,5	**64,5**	80,5	4,1	36,2	63,2	0,9	0	15,0	71,1	16,5	61,7	22,8	89,2	82,7	38,7
BestOf	4,2	4,7	49,5	78,1	5,4	27,1	55,9	**1,2**	0	**20,4**	77,4	**22,5**	56,7	14,3	84,8	83,2	32,5
3DLabelProp	**9,6**	**24,1**	50,0	**80,6**	**14,3**	**65,2**	**78,1**	0,7	0,0	7,8	**79,2**	6,8	**68,3**	**29,2**	**93,2**	**89,4**	**43,5**

TABLE 4.14 – Résultat de généralisation de 3DLabelProp de nuScenes vers ParisLuco3D.

4.3.7 Comparaison avec les autres méthodes de généralisation

Pour nous comparer le mieux possible aux autres travaux de l'état de l'art, nous nous comparons ici à trois méthodes : DGLSS [87], LIDOG [108] et Complete & Label (C&L) [85], respectivement dans le tableau 4.15, le tableau 4.17 et le tableau 4.16. DGLSS est la première autre méthode de généralisation de domaine pour la segmentation sémantique LiDAR pour le véhicule autonome, et travaille sur un jeu de labels exhaustif et explicité. LIDOG est la seule autre méthode de géné-

ralisation de domaine pour la segmentation sémantique 3D, le jeu de labels est bien plus simpliste, mais par exhaustivité des comparaisons, les résultats sont inclus ici. C&L est la méthode la plus proche de la nôtre, en se basant, elle aussi, sur une représentation intermédiaire du nuage de points dense. Les résultats de C&L utilisés ici sont ceux d'adaptation non supervisée (c'est-à-dire où on suppose l'accès à des données du domaine cible non annotées), car leurs expériences de généralisation n'ont lieu que sur deux classes.

Méthode	SK	NS	W
DGLSS [87]	59,6	**44,8**	40,7
3DLabelProp	**74,7**	44,2	**43,6**

TABLE 4.15 – Comparaison avec DGLSS, sur leur jeu de labels, entraînement avec SemanticKITTI.

	SK → NS			NS → SK		
Méthode	SK	NS	% drop	NS	SK	% drop
C&L [85]	50,2	31,6	**-37%**	54,4	33,7	-38%
3DLabelProp	**69,0**	**42,7**	-38%	**66,5**	**50.5**	**-24%**

TABLE 4.16 – Comparaison avec C&L, sur leur jeu de labels.

	SK → NS			NS → SK		
Méthode	SK	NS	% drop	NS	SK	% drop
LIDOG [108]	61,5	34,9	-43%	48,5	41,2	-15%
3DLabelProp	**83,0**	**58,8**	**-29%**	**82,4**	**73,9**	**-10%**

TABLE 4.17 – Comparaison avec LIDOG, sur leur jeu de labels.

Nous voyons que nous battons systématiquement C&L, même en perte de performance relative par rapport à la segmentation monodomaine, prouvant la pertinence de produire des nuages denses par géométrie plutôt que par apprentissage. Pour DGLSS, nous avons un modèle monodomaine bien plus efficace (+15,1%) pour des performances légèrement inférieures sur nuScenes (-0,6%), mais supérieures sur Waymo (+2,9%), rendant notre approche très compétitive. Pour LIDOG, notre méthode est bien supérieure à celle proposée ; cela dit, le jeu de labels ne présente aucun label considéré comme complexe, et la plupart des catégories sont statiques, facilitant la tâche pour une méthode pseudo-dense. De plus, les deux labels dynamiques (**person** et **vehicle**) ne présentent aucune ambiguïté l'un avec l'autre.

4.3.8 Analyse du temps d'inférence

Il a été observé que 3DLabelProp était bien plus efficace que les méthodes n'utilisant qu'un seul scan comme entrée, et qu'elle est aussi plus efficace dans le cadre monodomaine que les approches naïves d'accumulation des données. Cependant, un des critères d'exclusion des méthodes pseudo-denses est leur coût mémoire et temporel prohibitif. Dans le tableau 4.18, la vitesse d'exécution de 3DLabelProp est comparée à KPConv simple et pseudo-dense. Nous remarquons immédiatement que les méthodes basées sur KPConv, y compris 3DLabelProp, ne sont pas compétitives avec SRU-Net (tableau 4.9) pour le temps d'inférence et sont dans tous les cas loin des performances attendues (10 Hz). Cependant, on remarque que 3DLalelProp est un meilleur choix que KPConv pseudo-dense sur cette question. Du point de vue de la mémoire, les méthodes basées sur KPConv génèrent des sous-nuages dont la taille peut être adaptée et ne posent donc pas de soucis, contrairement à SRU-Net pseudo-dense.

Méthode	Vitesse d'inférence SemanticKITTI (Hz)	Vitesse d'inférence nuScenes (Hz)
KPConv [37]	0,6	1,3
KPConv pseudo-dense [37]	0,1	0,3
3DLabelProp	0,17	1,2

TABLE 4.18 – Vitesse d'exécution de 3DLabelProp.

4.3.9 Analyse de l'impact des différents modules de la méthode

Dans le cadre des modules de propagation et de clusterisation de 3DLabelProp, 5 paramètres ont été introduits : v_s, la taille des voxels pour le sous-échantillonnage ; d_{prop}, la distance de propagation, lié au paramètre de bande passante σ ; N_c, le nombre de sous-nuages ; V_c, la taille des voxels pour l'enrichissement des sous-nuages ; et N_f le nombre de scans utilisés dans le nuage accumulé. La relation entre d_{prop} et σ est : $\sigma = \frac{d_{prop}}{\sqrt{ln(2)}}$.

Pour la totalité des entraînements et des inférences, les mêmes jeux de paramètres sont utilisés : $v_s = 0.05$m, $d_{prop}= 0.30$m, $V_c = 2$m, $N_c = 20$ et N_f=20.

Pour mieux comprendre l'impact des différents modules de 3DLabelProp, les différents paramètres géométriques sont étudiés pour comprendre l'impact de chacun. Dans le tableau 4.19, l'influence de trois paramètres est étudiée : d_{prop}, N_c et N_f. v_s et V_c ne sont pas étudiés en détail, car v_s suit le standard de la littérature et V_c est lié à la consommation mémoire. L'étude proposée est effectuée avec nuScenes en source, car les entraînements et les inférences sont plus rapides dans ce cas.

La première observation est que même avec des paramètres mal sélectionnés (d_{prop}=0.60m et N_f=5), 3DLabelProp atteint des performances du niveau de l'état

d_{prop}	N_c	N_f	mIoU$^{NS}_{\mathcal{L}_{NS}}$	mIoU$^{SK}_{\mathcal{L}_{NS\cap SK}}$	Vitesse d'exécution (Hz)
0,30m	20	20	71,5	59,8	1,2
0,10m	20	20	72,4	60,2	0,6
0,60m	20	20	69,1	57,2	1,5
0,30m	5	20	71,3	61,5	1,3
0,30m	40	20	68,8	59,7	0,9
0,30m	20	5	67,4	60,0	1,5
0,30m	20	10	70,9	60,2	1,5
0,30m	20	40	67,9	59,5	1,0

TABLE 4.19 – Etude des paramètres géométriques de 3DLabelProp entraîné sur nuScenes (NS).

de l'art dans le cas de la généralisation de nuScenes à SemanticKITTI, démontrant la robustesse de l'approche malgré la pluralité de paramètres.

L'impact principal de ces paramètres est sur le temps de calcul (FPS) avec un facteur 3 entre les choix de paramètres les plus rapides et les plus lents. d_{prop}=0.30m est sélectionné comme bon compromis entre rapidité et qualité des résultats.

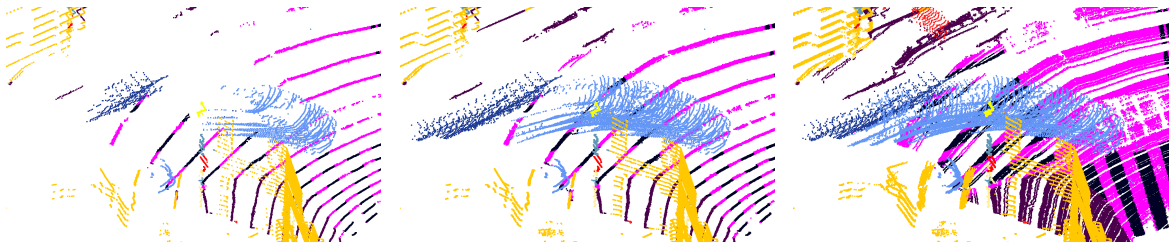
Le paramètre N_c, le nombre de sous-nuages, est plus lié à la consommation mémoire. Plus il y a de sous-nuages, qui sont donc supposément plus petits, plus les calculs se font rapidement. Cependant, passé un certain niveau, augmenter le nombre de sous-nuages augmente la superposition de ces sous-nuages et la consommation de mémoire remonte. Bien que cette dernière ne soit pas reportée ici, utiliser N_c=5 provoque des soucis mémoire avec SemanticKITTI, ce qui explique le choix de N_c=20.

Bien qu'intuitivement on pourrait s'attendre à ce que plus N_f soit grand, meilleures soient les performances grâce à l'enrichissement des voisinages et à l'obtention d'une géométrie plus fine des objets, on voit qu'en pratique cela n'est pas le cas. Cela s'explique par l'existence des phénomènes de traînées, illustré dans la figure 4.19. On voit que lorsque la fenêtre temporelle d'acquisition augmente, les traînées se complexifient et il devient plus difficile d'extraire l'information du scan courant pour les objets mobiles. Il est donc intéressant de garder une fenêtre peu élevée, d'où le choix de N_f=20.

4.3.10 Conclusions

Les données 3D sont naturellement soumises à de nombreuses sources de *domain shift*, dues en partie à la grande variété de capteurs d'acquisition sur le marché, et la variété de leurs installations sur les véhicules. Malgré cela, pour le moment peu de travaux s'y sont intéressés.

Les méthodes actuelles se basant uniquement sur un seul nuage de points en tant qu'entrée présentent de fortes lacunes en généralisation de domaines. De plus, on observe que de bonnes performances monodomaines peuvent cacher un surapprentissage sur celui-ci et une incapacité à généraliser à de nouvelles scènes comme



(a) N_f=5. (b) N_f=20. (c) N_f=40.

FIGURE 4.19 – Phénomène de traînées dans ParisLuco3D pour une fenêtre d'accumulation de taille 5, 20 et 40 scans.

le montre Cylinder3D. Cependant, on remarque que certaines architectures ont des capacités naturelles pour mieux généraliser, comme SRU-Net.

Les nuages de points pseudo-denses présentent une piste d'amélioration des performances de généralisation monosource en étant résilientes aux *sensor shifts*. Leur écueil principal est leur temps d'inférence grandement augmenté et loin des prérequis des véhicules autonomes.

Dans ce chapitre, une méthode basée sur les nuages pseudo-denses a été présentée ; elle a de bonnes capacités de généralisation et démontre de fortes capacités de compréhension de la géométrie de la scène. Cette méthode, 3DLabelProp, est elle aussi loin des performances temps réel, mais est une illustration des capacités de généralisation que pourraient atteindre des méthodes s'intéressant à de nouvelles modalités d'entrée des réseaux de segmentation 3D.

Chapitre 5

Généralisation de domaine multisource : méthode & application

Résumé

Nous avons vu des stratégies de généralisation de domaine monosource lorsque uniquement un seul jeu de données, et par extension un seul domaine, est disponible lors de l'entraînement. Ce cas est assez réducteur, car en pratique plusieurs jeux de données sont disponibles pour entraîner les réseaux. Dans ce contexte, il est intéressant d'effectuer de la généralisation de domaine multisource. Ce chapitre se concentrera sur cette tâche, en se demandant comment la réaliser pour la segmentation sémantique LiDAR, où les jeux de données source ont des jeux de labels différents. Puis, on s'intéressera à son impact sur les performances de généralisation de plusieurs architectures ainsi que plusieurs stratégies de généralisation de domaine de référence. Finalement, un cas pratique d'utilisation de la généralisation de domaine pour l'amélioration de performance monodomaine sera présenté. Pour réaliser ces deux objectifs nous introduisions la stratégie COLA, les *coarse labels*.

Sommaire

5.1 Comment utiliser plusieurs jeux de données simultanément

5.1.1 Travaux connexes en 2D

Pour faire de la généralisation de domaine multisource, il est nécessaire d'être capable d'utiliser plusieurs jeux de données simultanément. Lorsque tous les jeux de données ont le même jeu de labels, cela ne pose aucun soucis. La création d'un nouveau jeu de données par concaténation des jeux de données sources est alors une méthode donnant de solides résultats. Comme vu dans la section 4.1, il existe aussi des moyens plus sophistiqués d'utiliser ces jeux de données ensemble, notamment avec le meta-apprentissage.

La situation est plus complexe lorsque les jeux de labels sont différents. Certaines approches font le choix de n'utiliser que l'intersection entre les divers jeux de labels [130, 131], ou l'union [132]. D'autres méthodes, appelées *multi-head (MH)*, partagent un *encoder*, mais proposent d'avoir un *decoder* (aussi appelé tête de classification) pour chacun des jeux de données d'entrée [133, 134]. Une illustration de ces différentes approches est disponible sur la figure 5.1. Ces méthodes, lorsqu'elles sont utilisées pour faire de la généralisation de domaine, font la supposition que soit le jeu de labels cible est contenu dans l'intersection (ou l'union) des jeux de données sources, soit que le jeu de labels correspond à une des têtes de segmentation.

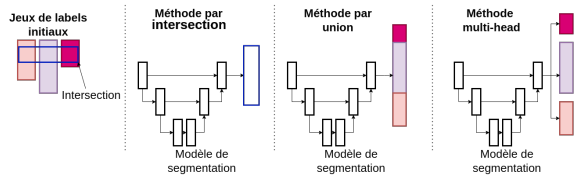


FIGURE 5.1 – Illustration des stratégies de segmentation multidomaine par intersection des labels, union des labels et par segmentation *multi-head*.

Des travaux observent que généralement, même si les jeux de labels sont différents, s'ils représentent la même scène il existe des connexions ontologiques entre eux. Ces relations entre labels sont alors explicitement exploitées [135] [136] [137] lors de l'entraînement. Parmi les stratégies existantes, [137] et [136] modélisent un arbre de relation (voir la figure 5.2), où plus on avance dans l'arbre plus l'annotation est fine, et donc des annotations plus ou moins grossières peuvent être utilisées lors de l'inférence.

Finalement, certaines méthodes font tout simplement le choix de relabéliser les données, soit par réassignation des labels [138] soit par réannotations des données [139].

Les méthodes d'adaptation de domaine *category-shift multi-source*, d'après la taxonomie de [140], s'intéressent aussi à un problème similaire, celui d'augmenter les performances de segmentation sur un jeu de données cible présentant un jeu de

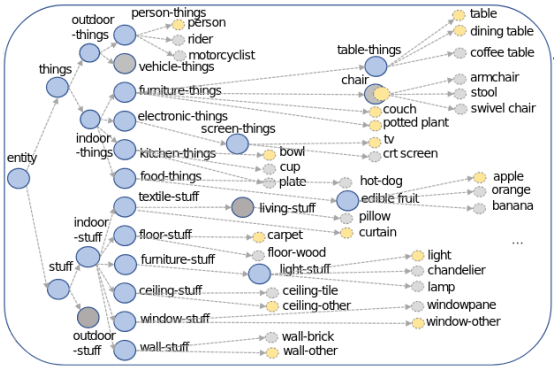


FIGURE 5.2 – Exemple d'arbre de labels, extrait de [137].

labels différent de ceux d'entraînement. Parmi eux, Deep Cocktail Network [141] et mDALU [140] sont des méthodes de référence. mDaALU explicite un espace de label unifié qui est utilisé pour pseudo-labelliser les images et augmenter le nombre d'exemples de chaque jeu de labels.

En 3D, [142] fait le constat de la diversité des jeux de données et de la complexité de combiner leurs jeux de labels. Pour parvenir à utiliser plusieurs jeux de données à la fois, ils utilisent un modèle de langage préentraîné pour comprendre et modéliser la proximité entre les labels des différents jeux de données. Les prédictions se font aussi dans cet espace continu qui représente l'espace des labels. [142] se concentre cependant uniquement sur la compréhension de scène intérieure.

5.1.2 Difficultés d'utilisation de multiples jeux de données

De la même façon que pour les problématiques observées précédemment, les jeux de données pour la segmentation sémantique LiDAR pour le véhicule autonome ne peuvent pas être utilisés simultanément de façon simple à cause de la diversité de leurs jeux de labels. On peut le voir dans le tableau 2.10 où on remarque que la totalité des jeux de données pris deux à deux ont des jeux de labels différents ce qui pose un problème lors de la généralisation de domaine multisource.

Une des stratégies privilégiées dans la généralisation de domaine monosource était l'utilisation de l'intersection des jeux de labels source et cible. Cette stratégie n'est pas applicable dans le cas de la généralisation multisource car, le nombre de jeux de labels à intersecter étant plus conséquent, les intersections sont souvent réduites et cela résulte en un grand nombre de labels mis de côté. De plus, les labels restants ont tendance à considérer des objets consensuels, dont la route et les voitures, mais perdre la finesse des informations de chaque jeu de labels. En outre, comme vu avec le *label shift*, des labels aux noms identiques dans deux jeux de données différents peuvent représenter des éléments différents de la scène. Une illustration de ce problème est présentée sur la figure 5.3

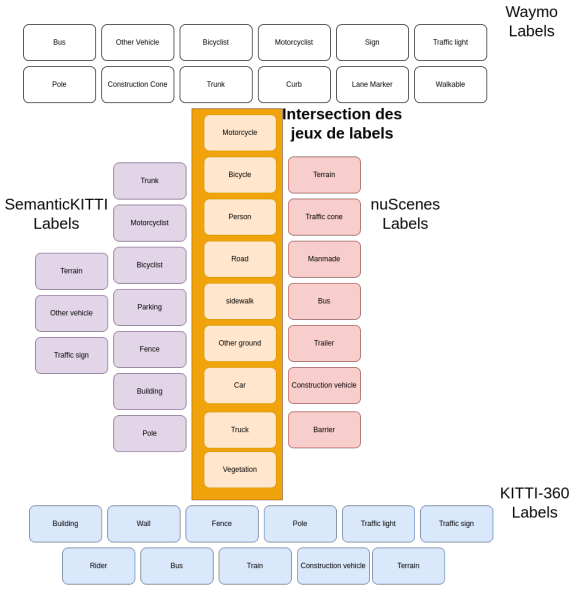


FIGURE 5.3 – Illustration du problème de l'intersection de nombreux jeux de labels.

De la même façon, utiliser l'union des labels résulte en un très grand nombre de labels, accroissant encore plus le déséquilibre dans la fréquence d'apparition entre les labels les plus présents et les moins présents et donc rendant plus difficile l'entraînement des réseaux de segmentation.

5.1.3 Introduction des *coarse labels*

Inspirée par les méthodes de modélisation par arbre des jeux de labels, notre méthodologie consiste à extraire l'arbre des labels à partir des jeux de données disponibles pour la tâche de segmentation sémantique de données LiDAR pour le véhicule autonome. L'arbre extrait se trouve sur la figure 5.4. On observe que naturellement les différents éléments de la scène se regroupent en plusieurs macro-catégories qui sont présentes dans tous les jeux de données.

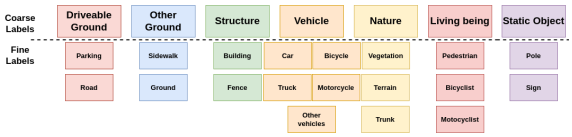


FIGURE 5.4 – Illustration de la hiérarchie de labels extraite pour SemanticKITTI.

À partir de ces labels extraits, on peut réassigner la totalité des jeux de labels vers ces macro-catégories qui constituent un jeu de labels applicable à tous les jeux de données pour le véhicule autonome. Ces labels extraits sont appelés les *coarse labels*. Ils ont l'avantage d'être suffisamment simples pour être appliqués à tous les jeux de

données considérés, mais conservent une représentation relativement intéressante de la scène. Sept labels sont extraits : *Vehicle*, *Driveable Ground*, *Other Ground*, *Person*, *Object*, *Structure* et *Vegetation*. Les différentes catégories ont un titre assez clair, la distinction *Structure*/*Object* provient du jeu de labels de nuScenes pour la détection d'objets. Les *Objects* correspondent à des éléments identifiables un à un comme les poteaux et les panneaux, alors que les *Structures* sont les éléments continus de la scène comme les barrières et les bâtiments. Une illustration du label *Vehicle* et de ses correspondances dans les jeux de données se trouve sur la figure 5.5. Ce jeux de labels est légèrement différent de celui utilisé dans [143] car la notion de *dynamic* et *static object* a été considérée trop ambiguë.

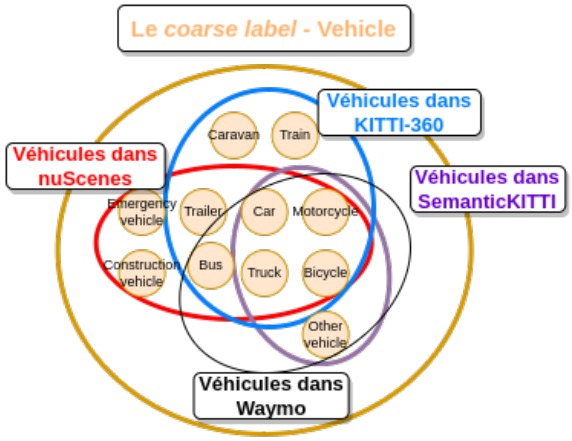


FIGURE 5.5 – Le *coarse label Vehicle*.

Ce nouveau jeu de labels peut être appliqué à tous les jeux de données. On peut donc faire de l'entraînement multisource et de l'évaluation multidomaine en ayant avoir un jeu de labels pour comparer les architectures de segmentation sur tous les jeux de données.

5.2 Benchmark

5.2.1 Expériences

Afin de déterminer l'effet de l'utilisation d'un entraînement multisource, trois architectures de segmentation sémantique 3D, à savoir SRU-Net, Cylinder3D et KPConv, sont utilisées et entraînées grâce aux *coarse labels*, avec quatre jeux de données d'entraînement : KITTI-360 (K360), SemanticKITTI (SK), nuScenes (NS) et Waymo (W). Ces jeux de données sont choisis car ce sont les plus grands parmi ceux disponibles. Lorsqu'ils sont utilisés ensemble avec les *coarse labels* le jeu de données résultant est appelé COLA. Ensuite, quatre jeux de données sont utilisés pour l'évaluation, qui a aussi lieu sur les *coarse labels* : ParisLuco3D (PL3D), Se-

manticPOSS (SP), Panda64 (P64) et PandaFF (PFF). Ces quatre jeux de données représentent des scènes, résolutions de capteurs et topologies d'acquisition différents. Pour faciliter l'utilisation de ces données, dans chaque cas la caractéristique de réflectivité est laissée de côté.

Le choix des architectures se base sur les capacités de généralisation monosource ainsi que sur les capacités de segmentation monodomaine. Ces trois architectures sont systématiquement comparées à leur équivalent entraîné avec SemanticKITTI, le jeu de données de référence, habituellement utilisé. Lors de cet entraînement monosource, les *coarse labels* sont tout de même utilisés.

Par la suite, une étude complémentaire pour comprendre l'interaction et l'effet de la diversité des jeux de données disponibles lors de l'entraînement sera réalisée. Pour cela, plusieurs combinaisons des quatre jeux de données seront effectuées pour constater leur effet sur les capacités de généralisation. Ces expériences sont entièrement réalisées avec SRU-Net. Nous comparerons aussi l'effet des *coarse labels* par rapport à de l'entraînement *multi-head*, mentionné dans la section précédente.

De plus, de la même façon que SemanticKITTI-32 introduit dans le chapitre précédent, SemanticKITTI-16 est présenté ici. Ce jeu de données est utilisé pour évaluer l'intérêt d'avoir une variété de résolution de capteurs au sein des jeux de données.

5.2.2 Résultats initiaux de généralisation de domaine multisource

En complément des valeurs moyennes présentées dans le tableau tableau 5.1, les résultats par classe du benchmark réalisé peuvent se trouver dans l'annexe C. Les expériences sont toutes réalisées à quantité d'epochs égale, choisie par observation des plafonnements de performance à 5 epochs.

La première observation est l'amélioration systématique des performances de généralisation lorsque les modèles sont exposés à l'entraînement multisource. Il est important de noter que l'exposition à un jeu de données multidomaine à deux effets : l'augmentation de la diversité des exemples en multipliant les domaines, et l'augmentation de la taille du jeu d'entraînement.

Alors que nous souhaitons démontrer que la diversité du jeu d'entraînement augmente les performances de généralisation, il est nécessaire de tenir compte du fait que les modèles d'apprentissage profond profitent également grandement de l'expansion du jeu d'entraînement. Cet effet sera étudié en détail dans la section suivante ; pour le reste de cette discussion, nous pouvons supposer que c'est bien la diversité des jeux de données qui aide les performances de généralisation.

Les gains de performances sont présents pour tous les types de *domain shifts*, avec des gains conséquents vers ParisLuco3D et PandaFF démontrant la création d'une résilience au *sensor shift*, mais aussi au *scene et appearance shifts* avec le gain de performance vers Panda64 et SemanticPOSS.

Les performances de Cylinder3D sont particulièrement remarquables, surtout

au vu de ses mauvaises performances de généralisation dans le cadre monosource (voir chapitre 4). Ses bonnes performances multisource s'expliquent de la même façon que ses mauvaises performances monosource : c'est l'effet du PointNet de prétraitement. Alors qu'en monosource il surapprend sur le domaine des données auquel il est exposé, c'est aussi une architecture avec de très bonnes performances de généralisation [144, 145]. Il parvient à créer des caractéristiques indépendantes du domaine lors de l'entraînement multisource et laisse Cylinder3D exprimer ses hautes performances sur de nouveaux domaines. Les performances catastrophiques sur PandaFF s'expliquent par la voxelisation cylindrique de l'espace inadaptée aux capteurs non rotatifs.

On voit que lorsque KPConv est entraîné sur les *coarse labels* monosource, il surapprend beaucoup plus qu'avec une grande variété de labels comme c'est le cas habituellement, en raison de l'utilisation dans le vecteur caractéristique d'entrées de la coordonnée z des points. Utiliser plusieurs jeux de données à la fois gomme l'effet de cette caractéristique.

SRU-Net, en plus d'être une bonne architecture de généralisation monosource, profite pleinement d'un jeu de données varié et atteint des performances satisfaisantes sur la totalité des jeux de données cibles. Cependant, une diminution des performances sur SemanticKITTI est à noter.

Bien que le jeu de labels considéré soit particulièrement simple, on observe l'intérêt de réaliser des entraînements multisource plutôt que monosource pour réaliser de la généralisation de domaine.

Méthode	Jeux de données	SK	SP	PL3D	P64	PFF
KPConv [37]	SK	**78,6**	27,3	39,0	25,4	15,1
KPConv [37]	COLA	73,8	**52,3**	**50,0**	**56,0**	**51,7**
Cylinder3D [65]	SK	81,8	32,0	41,0	29,1	7,6
Cylinder3D [65]	COLA	**82,5**	**56,6**	**67,0**	**59,8**	**28,3**
SRU-Net [9]	SK	**81,9**	35,3	44,3	42,8	42,8
SRU-Net [9]	COLA	75,1	**57,7**	**59,6**	**62,5**	**57,8**

TABLE 5.1 – Benchmark de généralisation de domaine multisource selon l'architecture et les jeux de données d'entrée, résultats sur les *coarse labels*.

5.2.3 Étude de l'influence du jeu d'entraînement

Comme mentionné auparavant, il est important de différencier l'impact de la taille augmentée du jeu de données de l'impact de la variété du jeu de données multisource. De plus, on souhaite comprendre l'impact de l'introduction de chaque *domain shift* au sein du jeu d'entraînement. Le tableau 5.2 s'intéresse à l'effet de l'introduction de *sensor shift* au sein du jeu d'entraînement, et son impact sur les

facultés de généralisation de domaine.

On observe une légère perte de performance dans le cas monosource sur SemanticPOSS qui est le jeu de données présentant le plus de similitudes avec SemanticKITTI. Pour PandaSet et ParisLuco, présentant des topologies d'acquisition plus éparses que SemanticKITTI, les performances de généralisation augmentent fortement en introduisant uniquement un jeu de données à faible résolution dans le jeu de données d'entraînement, même si SemanticKITTI16 ne représente aucun des capteurs d'acquisition des jeux de données cibles.

On réalise une dernière expérience en réutilisant SemanticKITTI32 pour l'ajouter dans le jeu de données d'entraînement. Intégrer un jeu de données de résolution intermédiaire aide la généralisation sur le jeu de données présentant le plus de similitudes en termes de topologie d'acquisition, c'est-à-dire ParisLuco3D. Pareillement, on observe une augmentation de performance sur Panda64. Ces expériences démontrent l'intérêt d'intégrer de la diversité de résolution d'acquisition, même si les scènes sont les mêmes.

Jeux d'entraînement	SK	SK16	SK32	SP	PL3D	P64	PFF
SK	81,9	66,2	77,7	35,3	44,3	42,8	42,8
SK+SK16	79,5	73,9	77,8	33,5	52,3	46,6	41,7
SK+SK16+SK32	**82,5**	**77,5**	**80,8**	**36,9**	**55,2**	**50,7**	**50,7**

TABLE 5.2 – Étude de l'impact de la diversité de résolution de capteur dans le jeu d'entraînement sur SRU-Net.

En complément de cette expérience préliminaire, nous effectuons une analyse exhaustive de l'effet de la taille du jeu de données et de diversité sur les performances de généralisation dans le tableau 5.3. Les résultats sont plutôt attendus, et confirment nos hypothèses implicites. Ajouter nuScenes augmente fortement les performances de généralisation sur ParisLuco3D, car ils utilisent le même capteur d'acquisition. Utiliser Waymo améliore fortement les performances sur Panda64, pour les mêmes raisons. Utiliser KITTI-360 à la place de SemanticKITTI augmente les performances de généralisation, en raison de l'exposition à un jeu de données plus grand même si la scène reste la même. La meilleure option est de concaténer tous les jeux de données, ce qui permet de couvrir plus de scènes différentes et d'augmenter la taille du jeu de données.

On compare aussi les résultats avec une autre stratégie d'utilisation des jeux de données, la stratégie multi-head. Pour utiliser cette méthode en cas de généralisation de domaine, les quatre têtes de segmentation réalisent une inférence sur leurs jeux de labels respectifs. Les scores en sorties des têtes sont ensuite réassignés aux *coarse labels*. Les scores peuvent alors être additionnés et correspondent à une prédiction effectuée directement sur les *coarse labels*. Ceci a pour effet que lorsqu'une des têtes de segmentation est largement plus sûre d'elle que les autres, elle prend la décision

globale de sémantisation. Cela a un effet positif pour ParisLuco3D, où la tête de segmentation correspondant à nuScenes est extrêmement pertinente. Pour les autres jeux de données n'ayant pas de géométrie d'acquisition identique parmi les jeux d'entraînement, c'est la décision moyenne des quatre têtes qui est retenue et résulte en un score moins bon que dans le scénario monotête. On voit que cela reste une stratégie globalement acceptable même si perfectible. Comme les résultats moyens sont moins bons qu'en utilisant directement les *coarse labels*, cette méthode ne sera pas plus étudiée.

Jeux d'entraînement	SK	SP	PL3D	P64	PFF
SK	**81,9**	44,3	35,3	42,8	42,8
K360	61,8	42,5	50,1	44,2	42,7
K360 + SK	70,7	43,2	48,5	49,9	47,4
K360 + NS	61,0	46,3	53,5	48,7	39,4
K360 + W	63,1	54,3	52,4	59,5	52,2
K360 + SK + W + NS	75,1	**57,7**	59,6	**62,5**	**57,8**
K360 + SK + W + NS (MH)	59,9	54,2	**62,6**	49,4	30,3

TABLE 5.3 – Étude de l'impact du jeu d'entraînement pour la généralisation de domaine multisource sur SRU-Net.

5.3 La généralisation de domaine multisource comme préentraînement - COLA

5.3.1 Le préentraînement pour la segmentation sémantique 3D

Les préentraînements, et plus généralement les méthodes de *transfer learning* ont pour objectif de ré-employer des connaissances apprises en amont de la tâche que l'on souhaite réaliser pour en améliorer les performances. L'amélioration des performances est vaste et peut concerner plusieurs points comme la vitesse d'entraînement, la quantité de données requises pour l'entraînement ou plus directement les performances quantitatives après entraînement. On distingue généralement cette stratégie d'entraînement en deux parties, le **préentraînement** réalisé en amont et le ***finetuning*** qui correspond à s'entraîner sur la tâche réellement souhaitée.

Il existe plusieurs stratégies de préentraînement : soit réaliser une tâche auxiliaire sur des données différentes de celle du finetuning, soit préutiliser les données de finetuning. Les jeux de données de préentraînement sont généralement très grands afin d'apprendre des primitives générales aux réseaux préentraînés. De ce fait, comme en 3D il existe peu de données annotées, ils sont souvent réalisés avec des tâches non supervisées comme les méthodes contrastives.

Parmi ces méthodes on peut noter PointContrast [146], CSC [147], DepthContrast [148] et CrossPoint [149]. Les différences entre ces méthodes proviennent soit de la source de données utilisée pour le préentraînement, nuage de points ou image RGBD, soit de la façon de calculer la fonction de perte. PC-FractalDB [150] propose une alternative aux approches contrastives en utilisant l'apprentissage de la géométrie fractale comme préentraînement. Point-BERT [151] s'intéresse lui uniquement au préentraînement de transformers. Ces méthodes, bien qu'elles s'intéressent aux tâches de compréhension de formes et de scènes, traitent des nuages de points denses grâce à la plus grande disponibilité de ce type de nuages de points [152, 153, 154]

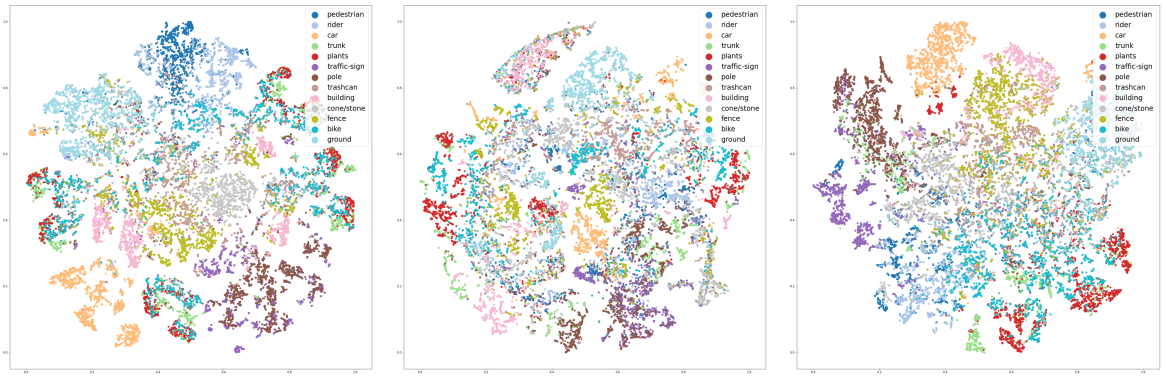
Quelques méthodes s'intéressent au préentraînement pour les nuages de points épars. GCC-3D [155] applique une méthode auto-supervisée en deux étapes : la nature séquentielle des données est utilisée pour créer des pseudo-instances utilisées pour apprendre des clusters sémantiques. ProposalContrast [156] adapte les approches contrastives aux nuages épars. Ces deux méthodes sont conçues pour la détection d'objets.

Finalement, seuls SegContrast [157] et TARL [158] s'intéressent à la segmentation sémantique 3D pour le véhicule autonome. SegContrast est largement inspiré de PointContrast, sauf que les paires de nuages de points utilisés ne sont pas obtenus par recalage mais par augmentation de données. TARL utilise un modèle de déplacement du véhicule lors de l'acquisition pour créer des paires d'objets d'un nuage à l'autre et cherche à minimiser la distance entre des représensations de ces objets pour deux nuages différents. La stratégie de préentraînement est non supervisée dans les deux cas. Ces deux stratégies utilisent le même jeu de données en préentraînement et en finetuning.

5.3.2 Motivation et intuition

L'objectif du préentraînement d'architecture neuronale est l'apprentissage de primitives géométriques réemployables au moment du finetuning pour aider le réseau à apprendre. Le cœur du concept, l'existence de ces primitives géométriques réemployables est similaire à la généralisation de domaine, qui cherche des caractéristiques indépendantes du domaine. Dans le cas spécifique où la tâche de préentraînement est la même que la tâche de finetuning, si après préentraînement le modèle généralise bien, on a une garantie de la pertinence des primitives apprises.

Pour vérifier cette hypothèse, le même protocole expérimental que SegContrast [157] sera suivi, avec SemanticPOSS en jeu de données de finetuning. L'étude sur SemanticKITTI ne sera pas proposée, car le jeu de préentraînement disponible serait alors trop faible pour être convaincant. Dans son analyse, SegContrast conclut que la méthode contrastive est plus pertinente qu'un entraînement supervisé (préentraîné avec SemanticKITTI et finetuné sur SemanticPOSS). Nous pousserons ici l'analyse plus loin en testant différents degrés de diversité dans le jeu de préentraînement et donc plusieurs résultats de généralisation.



(a) t-SNE de COLA. (b) t-SNE de SegContrast. (c) t-SNE d'un préentraînement supervisé sur SemanticKITTI.

FIGURE 5.6 – Analyse t-SNE des caractéristiques obtenues sur SemanticPOSS par COLA, SegContrast et un préentraînement supervisé avec uniquement SemanticKITTI sur SRU-Net.

Notre méthode de préentraînement supervisé, lorsque la totalité des jeux de données est utilisée avec les *coarse labels*, s'appelle COLA (COarse LAbels pre-training). Nous réalisons ce préentraînement sur deux architectures différentes : Cylinder3D et SRU-Net.

Une préanalyse qualitative confirme l'intérêt de réaliser un préentraînement supervisé. Pour cela, une analyse t-SNE du modèle préentraîné sur le jeu de données de finetuning est réalisée pour voir si les caractéristiques résultantes du préentraînement ont déjà une valeur sémantique. Cette analyse est sur la figure 5.6.

Un préentraînement sémantique devrait parvenir à préclusteriser les différentes classes dans l'espace de dimension réduit résultant de la t-SNE. On voit une forte confusion entre certaines classes de géométrie proche, ou de voisinage proche dans le cas du préentraînement avec SegContrast (confusion entre *Traffic sign* et *Pole* par exemple). A contrario, les préentraînements supervisés parviennent à isoler certaines classes avec pertinence, notamment *Car*, *Building* et *Fence*. On peut observer des zones de confusion entre *Rider* et *Pedestrian* qui sont dans le même coarse label. Les classes dont la géométrie fait la sémantique sont bien identifiées dans les deux cas, comme *Ground*. Globalement, les entraînements supervisés performent mieux que les entraînements contrastifs pour extraire a priori l'information sémantique.

5.3.3 Étude de l'impact des données d'entrée sur les performances du préentraînement

Les mêmes ensembles de jeux de données que ceux de la section précédente sont utilisés pour analyser l'effet de la généralisation de domaine sur la pertinence du préentraînement. Les valeurs présentes dans le tableau 5.4 et le tableau 5.5 corres-

pondent aux mIoU après préentraînement (mIoU_{DG}) sur la tâche de généralisation, et donc sur les *coarse labels*, et aux mIoU après *finetuning* (mIoU_{FT}) sur le jeu de labels de SemanticPOSS.

Méthode	mIoU_{DG}	mIoU_{FT}
Aucun préentraînement	N/A	64,2
SK	35,3	64,5
K360	42,5	64,8
K360 + SK	43,2	**65,1**
K360 + NS	46,3	64,8
K360 + W	54,3	64,9
COLA	**57,7**	64,9
K360 + SK + W + NS (MH)	54,2	64,6
SegContrast [157]	N/A	64,9

TABLE 5.4 – Étude de l'impact de la généralisation de domaine multisource sur les performances après finetuning sur SemanticPOSS avec SRU-Net.

Même si l'analyse préliminaire (figure 5.6) laissait voir une meilleure compréhension a priori du jeu de données SemanticPOSS à l'issue du préentraînement, en pratique à l'issue du finetuning la tendance peut s'inverser. Dans le cas de SRU-Net (tableau 5.4), la première remarque est qu'utiliser un préentraînement plus large qu'uniquement avec SemanticKITTI permet d'obtenir de meilleurs résultats et d'atteindre les mêmes performances que SegContrast, meilleure méthode de l'état de l'art pour le jeu de données SemanticPOSS. Dans le meilleur cas (SK+K360), SegContrast est même battu.

On voit qu'augmenter les performances de généralisation de domaine n'entraîne pas systématiquement un gain de performance après finetuning. Le meilleur cas correspond au jeu de données de préentraînement le plus proche géométriquement du jeu de données de finetuning.

Méthode	mIoU_{DG}	mIoU_{FT}
Aucun préentraînement	N/A	64,5
SK	32,0	64,3
COLA	**52,0**	**65,9**

TABLE 5.5 – Étude de l'impact de la généralisation de domaine multisource sur les performances après finetuning sur SemanticPOSS avec Cylinder3D.

Pour Cylinder3D on observe un gain de performance bien plus significatif (tableau 5.5) entre l'utilisation d'uniquement SemanticKITTI en préentraînement, qui n'apporte aucun gain de performance, et l'utilisation de COLA qui apporte le gain

le plus large de toutes les méthodes (+1.4%). Il avait été observé que Cylinder3D était une architecture à haut risque de surapprentissage, et utiliser COLA permet de fortement diminuer ce risque et donc crée un préentraînement pertinent.

Globalement, il est compliqué de conclure précisément sur le lien entre généralisation de domaine et performance après finetuning, cependant il est clair que COLA est une tâche de préentraînement efficace au vu des gains de performances similaires ou supérieurs à ceux l'état de l'art.

5.3.4 Résultat de finetuning

Au vu des résultats prometteurs de l'utilisation de la généralisation de domaine multisource comme tâche de préentraînement, les expériences précédemment réalisées sont étendues à trois autres cas. Dans le tableau 5.6 on s'intéresse au gain de performance selon la quantité d'accessibilité de la donnée au moment du *finetuning* sur SemanticPOSS. A chaque fois, deux cas sont considérés selon l'utilisation ou non de la réflectivité comme caractéristique d'entrée. Comme les méthodes de l'état de l'art TARL et SegContrast l'utilisent, nous présentons des résultats qui la prennent en compte.

On observe qu'à très faible quantité de données, et conformément aux résultats qualitatifs (figure 5.6), COLA permet une meilleure compréhension du jeu de données et surpasse l'état de l'art SegContrast. Dans les autres cas, COLA permet un gain de performance similaire à celui obtenu par SegContrast.

Méthode	0,1%	1 %	10%	50%	100%
Aucun préentraînement	33,1	43,1	57,3	63,3	64,2
SegContrast [157]	43,7	**55,2**	60,3	**64,6**	**64,9**
COLA	44,5	53,1	58,8	62,0	63,3
COLA avec réflectivité	**45,1**	54,5	**60,6**	63,6	**64,9**

TABLE 5.6 – Résultat de finetuning sur SemanticPOSS selon la proportion de données disponibles à l'entraînement, avec SRU-Net.

Ensuite, on s'intéresse à un autre cas d'utilisation d'une méthode de préentraînement en regardant les performances après *finetuning* sur des jeux de données vu lors du préentraînement qui sont SemanticKITTI et nuScenes. On simule deux cas d'application. Dans le tableau 5.7, on suppose la disponibilité de l'intégralité du jeu d'entraînement, et cela peut donc être vu comme une incorporation de la donnée multisource pour améliorer les performances de segmentation monodomaine. Dans le tableau 5.8, on suppose l'accès à une quantité limitée de données de *finetuning*, qui correspondrait à la disponibilité d'une grande quantité de données annotée grossièrement (par *coarse labels*) et d'une faible quantité annotée finement.

Dans le tableau 5.7, on observe que le gain de performance obtenu par utilisation

de la donnée monosource est non négligeable et arrive au même niveau que les techniques de préentraînement de l'état de l'art, voire à un niveau supérieur. De même, pour le cas à quantité de données limitée (tableau 5.8), l'information extraite par le préentraînement est fortement utile.

On voit que les modèles sans réflectivité sont très pertinents à faible quantité de données, mais que les modèles avec réflectivité atteignent de plus hautes valeurs après *finetuning*. Ces effets sont amplifiés par rapport à SemanticPOSS.

Méthode	SK	NS
Aucun préentraînement	59,6	66,0
SegContrast [157]	60,5	67,7
TARL [158]	61,5	68,3
COLA	61,8	69,0
COLA avec réflectivité	**64,9**	**70,0**

TABLE 5.7 – Résultat de finetuning sur SemanticKITTI et nuScenes, avec SRU-Net.

Méthode	0,1%	1 %	10%	50%	100%
Aucun préentraînement	25,6	41,7	53,9	58,3	59,6
SegContrast [157]	34,8	47,4	55,2	58,3	60,5
TARL [158]	**38,6**	51,4	**60,3**	61,4	61,5
COLA	37,8	**52,1**	58,5	**61,8**	61,8
COLA avec r	38,3	51,8	58,0	60,8	**64,9**

TABLE 5.8 – Résultat de finetuning sur SemanticKITTI selon la proportion de données disponibles à l'entraînement, avec SRU-Net.

5.3.5 Etude de l'importance du jeu de labels de préentraînement multisource

Pour finir l'étude du préentraînement multisource, nous nous intéressons à l'impact du choix de jeu de labels dans les performances après *finetuning* sur SemanticPOSS. On regarde les deux cas extrêmes où 0,1% des données sont disponibles et où l'intégralité des jeux de données sont disponibles. On compare 6 types de jeux de labels différents :

— COLA : notre méthode de *coarse labels* avec 7 labels,
— COLA-5 : une réduction à 5 labels (*Ground*, *Vehicle*, *Manmade*, *Pedestrian*, *Vegetation*) de COLA,
— COLA-9 : une extension à 9 labels de COLA (*Driveable Ground*, *Structure*, *4-Wheeled*, *Nature*, *Pedestrian*, *Object*, *Other Ground*, *Pole&Sign*, *2-Wheeled*),

— Intersection : garder les 9 labels identifiés dans la figure 5.3,
— Union : utiliser les 32 labels différents constituant les 4 jeux de données de préentraînement,
— MH : la méthode *multi-head*.

On observe dans le tableau 5.9 que plus les jeux de labels de préentraînement sont fins et proches de celui de *finetuning*, plus les performances à très faible quantité de données augmentent. Inversement, les performances à grande quantité de données diminuent, et ce sont les jeux de labels de préentraînement grossiers qui s'en sortent le mieux. Hormis pour la stratégie par Union, les performances sont supérieures avec le préentraînement que sans.

Il est important de noter que les stratégies MH, Union et Intersection sont dépendantes des jeux de données de préentraînement alors que les stratégies COLA ne le sont pas.

Méthode	0,1%	100%
Aucun préentraînement	33,1	64,2
COLA	44,5	64,9
COLA-5	43,7	**65,0**
COLA-9	45,2	64,8
Intersection	43,3	64,6
Union	**46,6**	63,6
MH	45,3	64,6

TABLE 5.9 – Impact du jeu de labels de préentraînement multisource, évalué sur SemanticPOSS, avec SRU-Net.

5.3.6 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la généralisation de domaine multisource et nous avons démontré son intérêt sur de multiples niveaux. Tout d'abord, il s'agit d'une stratégie très efficace de généralisation de domaine et cela améliore systématiquement les performances, peu importe le type d'architecture. De plus, elle permet à des architectures surapprenant en monosource de réussir à atteindre de bonnes performances de généralisation.

En outre, les entraînements multisource peuvent aussi être utilisés pour la segmentation monodomaine. L'incorporation d'informations multisource augmente les performances à faible et grande quantité de données disponibles pour le *finetuning*.

La totalité de ces expériences est réalisée grâce à des méthodes naïves de généralisation multidomaine, on peut donc supposer qu'utiliser des approches plus complètes et complexes inspirées de la littérature pourrait encore améliorer ces résultats.

Chapitre 6

Aller plus loin : *Moving Object Segmentation*

Résumé

Une tâche importante pour la conduite autonome est la localisation du véhicule. Avec des données LiDAR, une source importante d'erreur pour la localisation est la présence d'objets mobiles qui peut causer des problèmes lors des phases de *feature matching*. Pour pallier cela, des méthodes se sont intéressé à retirer les objets mobiles de la scène pour améliorer les performances de localisation, et ainsi la tâche de *Moving Object Segmentation* (MOS) s'est développée. Bien que ce travail se soit concentré sur la segmentation sémantique de données LiDAR, dans ce chapitre une ouverture sur les questions de généralisation pour la MOS est proposée. D'abord, nous parcourrons la littérature de la MOS et nous nous intéresserons aux performances de généralisation actuelles. Ensuite, 3DLabelProp sera adaptée pour faire de la MOS et nous comparerons cette méthode avec l'état de l'art, démontrant ainsi que 3DLabelProp ne se restreint pas à la segmentation sémantique.

Sommaire

6.1 Généralisation de domaine pour la MOS

6.1.1 La MOS pour le LiDAR

La tâche de MOS pour les données LiDAR s'est développée à partir de 2021 avec [159] qui met à disposition un benchmark en ligne de MOS sur les données de SemanticKITTI. Cependant, la tâche de MOS pouvait déjà être réalisée indirectement sur les données initiales de SemanticKITTI [3] qui distinguent les éléments *moving* des *stopped*. Cette dualité de la tâche de MOS donne lieu à deux stratégies pour en réaliser : voir la MOS soit comme un problème de classification binaire, soit comme un problème de segmentation sémantique. En raison de la nature de la tâche, contrairement à la segmentation sémantique traditionnelle, l'aspect séquentiel des données est généralement exploité par les méthodes de l'état de l'art. L'intérêt d'utiliser une séquence plutôt qu'un nuage seul est illustré sur la figure 6.1. La majorité

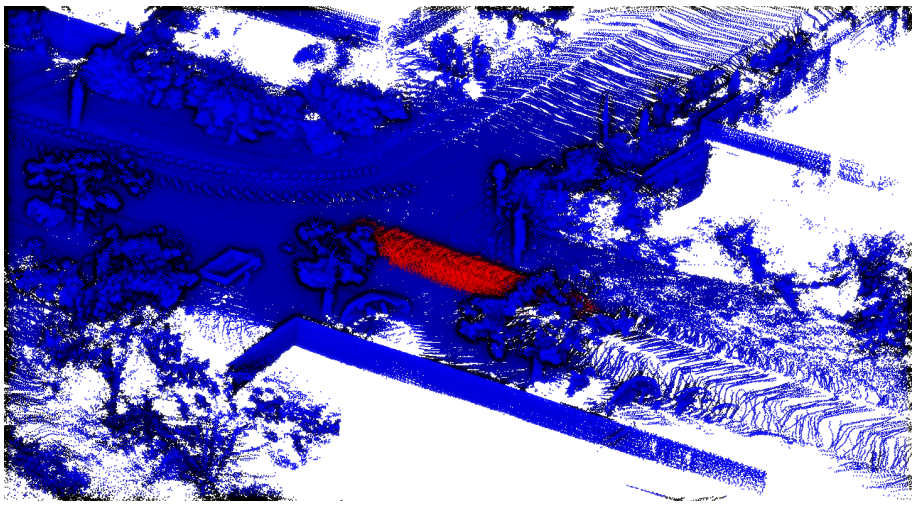


FIGURE 6.1 – Annotations de MOS, avec les éléments mobiles en rouge.

de la littérature s'intéresse à la MOS avec la première grille de lecture, et propose uniquement une classification binaire par point. Conjointement avec la sortie du benchmark, LMNet [159] est une des premières méthodes de MOS et introduit la notion d'images résiduelles dans le cadre des projections en *range image* des nuages de points. Les images résiduelles correspondent à la soustraction de deux *range images* consécutives et donc mettent en avant les objets mobiles qui se déplacent d'une image à l'autre. De telles images sont présentées sur la figure 6.2.

A partir de ce travail de référence, plusieurs autres approches ont émergés pour traiter la tâche de MOS. MotionSeg3D [160] s'inpire de SalsaNext [161] et LMNet en combinant la structure de SalsaNext et en injectant dans l'*encoder* des caractéristiques extraites d'images résiduelles. RVMOS [162] exploite aussi les *range images* et utilise conjointement l'information sémantique avec les images résiduelles pour

apprendre des caractéristiques plus pertinentes pour la MOS. Grâce à l'utilisation de *range images*, ces méthodes sont les plus rapides.

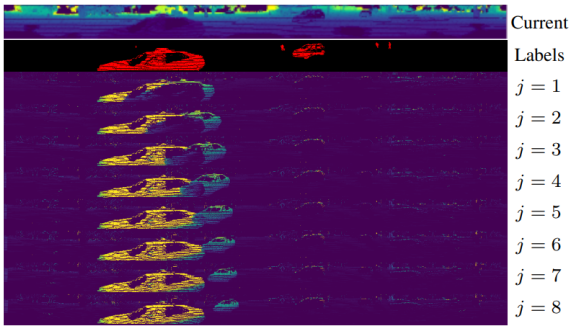


FIGURE 6.2 – Images résiduelles employées par LMNet [159].

3DSeq-MOS [163] et 4DMOS [164] travaillent directement sur les nuages de points. 3DSeq-MOS construit des nuages de points résiduels qui sont ajoutés au nuage de points courant pour réaliser la MOS. 4DMOS, similaire à 3DLabelProp, accumule ensemble tous les nuages de points de la séquence passée et les traite avec un 4D-SRU-Net.

Finalement, BeliefMOS [165] utilise conjointement le nuage de points nouvellement acquis et une carte géométrique de l'espace dans lequel sont segmentés les points mobiles et les points statiques. A partir de ces deux informations, un 4D-SRU-Net est utilisé pour prédire les points se déplaçant dans le nuage actuel. Un post-traitement bayésien est appliqué aux points nouvellement segmentés pour qu'ils soient conformes aux prédictions passés. Ce travail peut être vu comme une extension de 4DMOS.

MapMOS [165] s'intéresse à des considérations similaires aux nôtres et reporte les performances en généralisation de domaine de sa méthode sur deux autres jeux de données : nuScenes [79] et Apollo [166]. L'annotation de MOS pour Apollo a été réalisé par [167] et les auteurs de ce papier nous ont partagé ce jeu de données.

Les performances des différentes méthodes de MOS sur SemanticKITTI, validation et test sets, sont reportées dans le tableau 6.1. On remarque pour 4DMOS et MapMOS une forte différence entre les performances sur le jeu de validation et le jeu de test, ce qui met en avant l'importance de reporter les performances sur le test set.

6.1.2 Jeux de données

Pour le moment, il existe trois jeux de données permettant de faire de la MOS LiDAR : SemanticKITTI, Apollo et nuScenes. SemanticKITTI et nuScenes ont été présentés extensivement dans le chapitre 2. L'annotation utilisée est une annotation binaire pour *moving* et *static*. Traditionnellement, c'est SemanticKITTI qui est uti-

Méthode	SK val	SK test
LMNet [159]	63,8	62,5
4DMOS [164]	77,2	65,2
MapMOS [165]	**86,1**	66
MotionSeg3D [160]	71,4	70,2
3DSeq-MOS [163]	N/A	73,1
RVMOS [162]	71,2	**74,7**

TABLE 6.1 – Performance des méthodes principales de MOS sur SemanticKITTI.

lisé pour l'entraînement, car il s'agit du seul jeu de données proposant un benchmark en ligne pour la MOS.

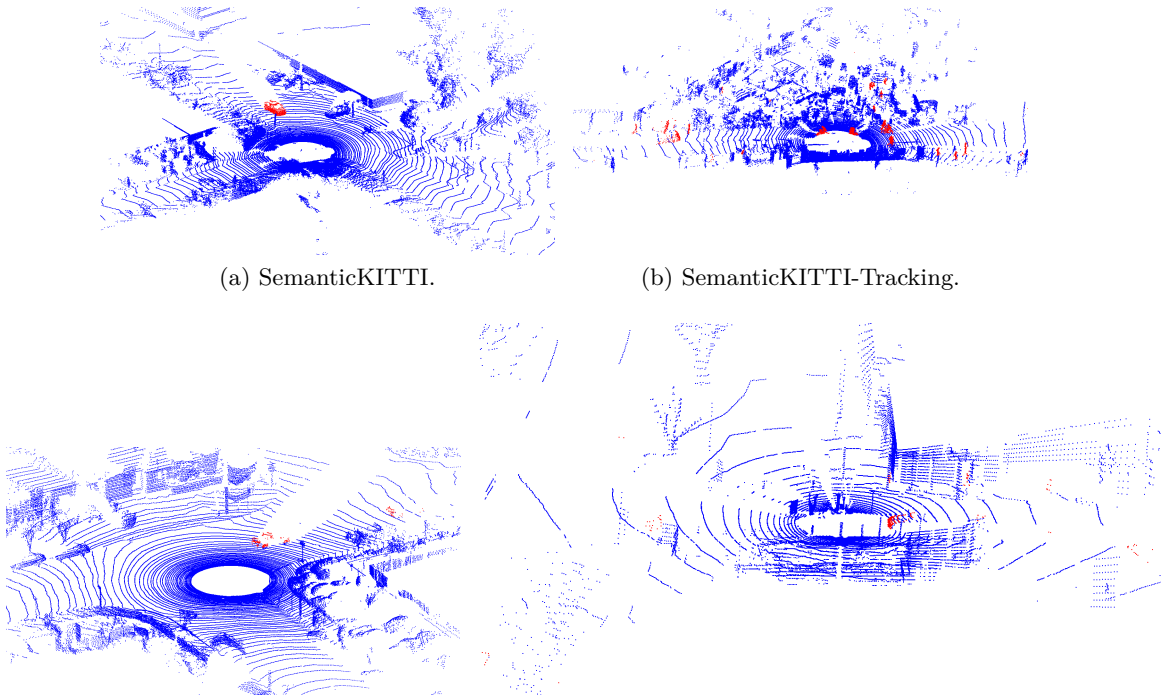
Apollo est initialement un jeu de données pour le SLAM et la détection d'objet. Comme mentionné précédemment, un sous-ensemble de ce jeu de données a été relabelisé pour la MOS. Ce jeu de données correspond à 6600 nuages de points annotés. Ils sont tous utilisés pour l'évaluation. Apollo est acquis avec un HDL-64, le même capteur que SemanticKITTI, résultant en un faible *sensor shift*.

Un jeu de données annexe est ajouté par [165], intitulé SemanticKITTI-Tracking. Il consiste en une séquence test de SemanticKITTI annotée pour la MOS et utilisée pour l'évaluation. La particularité de cette scène est la présence d'un très grand nombre d'objets dynamiques, contrairement aux autres séquences de SemanticKITTI, résultant en un *scene shift* par rapport aux données d'entraînement. Cette séquence est aussi utilisée uniquement pour l'évaluation. Les différents jeux de données sont présentés qualitativement sur la figure 6.3.

Un récapitulatif de la taille et de la spécificité des jeux de données de la MOS se trouve dans le tableau 6.2.

Nom	# nuages	# fibres	entraînement	Comparaison	Spécificité
SemanticKITTI [3] train	20000	64	✓	×	Faible densité d'objets mobiles
SemanticKITTI [3] val (SK val)	4000	64	✓	✓	N/A
SemanticKITTI [3] test (SK test)	20000	64	×	✓	N.A
SemanticKITTI-Tracking [3] (SKT)	1000	64	×	✓	Forte densité d'objets mobiles
Apollo [166] (AP)	6600	64	×	✓	N.A
nuScenes [79] train	18000	32	✓	×	Forte densité d'objets mobiles
nuScenes [79] val (NS)	6000	32	✓	✓	Forte densité d'objets mobiles

TABLE 6.2 – Récapitulatif des données utilisées pour la MOS.



(a) SemanticKITTI. (b) SemanticKITTI-Tracking.

(c) Apollo. (d) nuScenes.

FIGURE 6.3 – Illustration des différents jeux de données de MOS.

6.2 3DLabelProp pour la MOS

6.2.1 Adaptation de 3DLabelProp pour la MOS

3DLabelProp est initialement prévu pour la segmentation sémantique, et donc dans le cas de la MOS basée sur la sémantique, l'architecture ne demande quasiment aucun changement. La liste des labels statiques est juste mise à jour pour refléter les nouvelles annotations comprenant la MOS. Pour SemanticKITTI, on passe de 19 à 25 labels.

Dans ce cas-là, aucun paramètre géométrique n'est changé. La fenêtre temporelle est de taille 10, avec 20 sous-nuages extraits. La différence avec le chapitre 4, est que le jeu d'entraînement utilise la séquence 08, traditionnellement utilisée pour la validation. Ce choix est fait car certaines des classes (dont motorcyclist) ne sont représentées que dans cette séquence. Les résultats de la sous-section 6.2.3 dans le cadre monosource sont calculés sur le jeu de test en utilisant le benchmark en ligne.

Dans le cas de la MOS binaire, il est nécessaire de procéder à des ajustements. La quantité d'objets dynamiques dans SemanticKITTI est très faible, et le déséquilibre des classes étant accentué, certains paramètres géométriques doivent être adaptés. Le nombre de sous-nuages extraits est de 5 pour limiter les cas où les objets dy-

namiques sont scindés dans deux sous-nuages différents. De plus, le voisinage de l'enrichissement des sous-nuages est réduit pour limiter l'ajout de points statiques. La fenêtre d'accumulation est laissée à 10, valeur recommandée par la littérature. Pour comparer nos résultats à ceux de l'état de l'art en MOS, la séquence 08 est utilisé uniquement pour l'entraînement, même lors du calcul des performances sur le jeu de test, contrairement à ce qui est fait en segmentation sémantique.

Dans les deux cas, un léger changement dans les caractéristiques d'entrée a lieu. A la place d'utiliser uniquement l'information d'occupation en entrée de KPConv, le *timestamp* est ajouté. Une version très simple indiquant si le point considéré appartient au nuage courant ou au nuage accumulé est utilisée. Dans le tableau 6.3, l'avantage de cette caractéristique pour la MOS est démontrée. En pratique, le vecteur de caractéristique d'entrée est agrandi et les points du nuage courant ont la valeur 1 associée et les points précédents la valeur -1. Cela avait été aussi testé pour la segmentation sémantique mais n'apportait pas de résultats supplémentaires. Pour la MOS, on remarque une légère perte de performance monodomaine (SemanticKITTI val), mais améliore fortement les performances dans les autres cas.

Méthode	SK val	NS	AP	SKT
3DLabelProp	**79,0**	39,3	72,3	65,8
3DLabelProp avec *timestamp*	78,4	**41,6**	**76,3**	**71,2**

TABLE 6.3 – Influence du *timestamp* dans le cadre de la MOS binaire par comparaison des mIoU.

6.2.2 Analyse des performances de généralisation pour la MOS binaire

Contrairement à la segmentation sémantique traditionnelle, les modèles de MOS se posent déjà la question de la généralisation. Plus particulièrement, 4DMOS et MapMOS s'y sont intéressés et ont établi un panorama des performances de généralisation, qui est reporté dans le tableau 6.4. Trois des jeux de données sont acquis avec le même capteur, et seul nuScenes présente du *sensor shift*. Les résultats de 3DLabelProp Sémantique sont calculés en regroupant les prédictions des objets *moving* ensemble dans une seule classe, et en regroupant les autres prédictions dans une classe *static*.

Les méthodes non-conçues pour la généralisation (LMNet et MotionSeg3D) présentent de très mauvais résultats de généralisation, même sur Apollo qui est pourtant le même capteur que celui de SemanticKITTI. Les résultats sur nuScenes ne sont pas calculés car ce sont des méthodes basées *range* qui ont du mal à changer de résolution de capteurs.

MapMOS et 4DMOS prennent la généralisation en compte dans leur conception et démontrent de très bons résultats dans ce contexte. 4DMOS a les meilleures

performances sur nuScenes, qui est le cas le plus complexe de généralisation à cause du changement de résolution qui est très impactant dans l'acquisition des objets mobiles.

MapMOS présente les meilleures performances sur SK val, ainsi que sur AP et SKT, ce qui fait d'elle la méthode globale la plus efficace.

On voit que 3DLabelProp Binaire a des performances comparables à celles des autres méthodes à bonne capacité de généralisation (4DMOS et MapMos), en étant notamment plus performante en monodomaine et en étant compétitive sur la généralisation. On voit notamment que 3DLabelProp atteint les seconds meilleurs résultats sur Appolo et nuScenes.

En monodomaine, c'est RVMOS, une méthode basée *range* qui est la meilleure, démontrant l'intérêt de ces approches pour la MOS. Les résultats de généralisation ne sont pas calculés car le code n'est pas disponible.

Méthode	SK val	SK test	NS	AP	SKT
LMNet [159]	63,8	62,5	N/A	13,7	45,3
4DMOS [164]	77,2	65,2	**44,8**	73,1	75,5
MapMOS [165]	**86,1**	66,0	40,3	**81,7**	**78,4**
MotionSeg3D [160]	71,4	70,2	N/A	8,8	54,8
RVMOS [162]	71,2	**74,7**	N/A	N/A	N/A
3DLabelProp Binaire	78,4	67,0	41,6	76,3	71,2
3DLabelProp Sémantique	N/A	64,0	35,0	79,8	54,7

TABLE 6.4 – Performance des méthodes principales de MOS entraînées sur SemanticKITTI.

6.2.3 Analyse des performances de généralisation pour la MOS sémantique

En plus des résultats de MOS reportés dans le tableau 6.4, les performances sémantiques sur la totalité des 25 labels sont aussi évalués dans le tableau 6.5. Pour la tâche de MOS, la méthode par sémantique est globalement moins bonne que celle par classification binaire. Cependant, il est à noter que les performances sur Apollo sont grandement améliorées par cette approche, ce qui prouve l'intérêt de la considérer.

Pour la tâche de segmentation sémantique, les résultats sont globalement très insuffisants, et au vu des résultats par classe, les classes *moving* sont surapprises au détriment des éléments statiques. Que cela provienne de la stratégie d'entraînement ou des hyperparamètres, 3DLabelProp n'est pour le moment pas adapté. On note toutefois que sur 4 des 6 classes *moving*, 3DLabelProp surpasse sa baseline KPConv, démontrant la pertinence de l'approche par propagation.

Méthode	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	moving-car	moving-bicyclist	moving-person	moving-motorcyclist	moving-other	moving-truck	mIoU
KPConv [37]	**93,9**	44,9	**47,2**	**43,5**	**38,6**	**21,6**	0	0	**86,5**	**58,4**	**70,5**	**26,7**	**90,8**	**64,5**	**84,6**	**70,3**	**66,0**	**57,0**	53,9	69,4	67,4	67,5	**47,2**	4,7	**5,8**	**51,0**
3DLabelProp	93,4	**51,3**	28,9	25,5	38,3	8,8	0	0	84,3	51,5	64,3	17,2	88,5	54,9	82,5	68,4	62,7	56,5	47,3	**70,5**	**71,6**	**75,3**	24,4	**5,9**	0,5	46,9

TABLE 6.5 – Résultats de MOS sémantique sur SemanticKITTI test, comparaison avec KPConv pseudo-dense.

6.2.4 MOS multisource

Contrairement aux problématiques présentées dans le chapitre 5, un jeu de labels intermédiaire n'est pas nécessaire pour réaliser de la MOS multisource. En effet, naturellement la totalité des jeux de données ont le même jeu de labels. Cependant, la MOS est assujettie au *label shift*. En effet, dans SemanticKITTI, les véhicules à l'arrêt sont considérés comme *moving* s'ils sont sur la route, par exemple à un feu rouge. A contrario, nuScenes considère ces véhicules comme *static*.

Ce *label shift* n'étant pas pris en compte par les travaux précédents, nous ne le traiterons pas ici. On peut donc utiliser directement simultanément SemanticKITTI et nuScenes pour réaliser un entraînement. Nous entraînons 3DLabelProp avec ce jeu de données et ses résultats sont présentés dans le tableau 6.6. L'entraînement multisource est comparé aux entraînements monosource sur SemanticKITTI et nuScenes.

On observe d'abord l'incapacité à convenablement entraîner un modèle de MOS sur nuScenes, atteignant de faibles résultats même dans le cas monodomaine. Cette expérience permet tout de même de donner une borne haute à atteindre pour les performances vers nuScenes val.

Ensuite, on voit que la combinaison nuScenes et SemanticKITTI diminue les performances de généralisation vers SemanticKITTI-Tracking et Apollo ainsi que SemanticKITTI Val. Cela va de pair avec l'incapacité des modèles entraînés sur nuScenes à comprendre les scènes plus haute résolution, ce qui provoque une perte de performance sur les scènes acquises avec un capteur haute résolution. On observe un gain de performance vers nuScenes, ce qui met en avant un intérêt de l'apprentissage multisource.

Au vu des constats actuels, l'apprentissage multisource ne semble pas être une très bonne option pour la MOS. Cependant, contrairement au chapitre 5, les jeux de données d'évaluation montrent très peu de *domain shifts* relativement aux jeux d'entraînement et il reste compliqué de conclure précisément sur l'entraînement multisource.

Méthode	SK val	NS	AP	SKT
3DLabelProp entraîné sur SK	**78,4**	41,6	**76,3**	**71,2**
3DLabelProp entraîné sur NS	32,1	**48,9**	61,9	17,4
3DLabelProp entraîné sur SK+NS	73,9	44,8	24,9	70,0

TABLE 6.6 – Impact d'un entraînement multisource sur les performances de MOS.

6.3 Conclusion

Le domaine du MOS s'est posé les questions de la généralisation en amont de la segmentation sémantique, profitant notamment de l'uniformité des jeux de labels d'un jeu de données à l'autre. Certaines méthodes, dont MapMOS, sont déjà très efficaces sur cette question. Une des lacunes majeures reste leur sensibilité au *sensor shift*.

3DLabelProp passe très facilement de la segmentation sémantique à la MOS et est très compétitive sur cette tâche. Toutefois, les performances de généralisation sont satisfaisantes mais pas strictement supérieures au reste de l'état de l'art. Pour finir d'adapter complètement 3DLabelProp à la tâche de MOS, il serait probablement nécessaire de s'intéresser aux hyperparamètres d'apprentissage et géométriques plus finement.

Chapitre 7

Conclusion

Sommaire

7.1 Conclusion générale

Dans l'introduction, nous avons exprimé le souhait de nous intéresser à deux défis de la perception pour le véhicule autonome : la perception en milieu urbain dense et la fiabilité des systèmes de perception. Nous avions démontré l'intérêt théorique de la généralisation de domaine dans cette étude. A travers ce travail, nous avons démontré l'intérêt pratique de la généralisation de domaine pour l'amélioration de la fiabilité mais aussi pour l'amélioration de la perception en milieu urbain. Pour cela, nous avons du développer plusieurs outils, cadres d'analyse et expériences.

Premièrement, un nouveau de jeu de données a été créé pour améliorer la mesure des capacités de généralisation de domaine. Les jeux de données précédemment disponibles présentaient de fortes dissimilitudes dans leurs jeux de labels pour permettre des mesures équitables des performances de généralisation. ParisLuco3D a été conçu en gardant ce problème en tête, et son jeu de labels peut facilement être transposé sur les jeux de labels de nuScenes et SemanticKITTI, les deux jeux de données de référence.

Ensuite, la première observation est l'incapacité des méthodes actuelles de segmentation sémantique à continuer d'être performantes lors de *domain shifts* dans le cadre monosource. Il a été démontré que cela pouvait être compensé en s'intéressant à la représentation des nuages de points en entrée, les nuages pseudo-denses étant un vecteur d'amélioration des performances. De plus, l'intérêt de la généralisation multisource a aussi été mis en évidence. Ces méthodes peuvent même permettre à des modèles surapprenant dans le cadre monosource d'améliorer leurs performances de généralisation (comme Cylinder3D).

Pour répondre à cette observation, une nouvelle méthode de généralisation de domaine a été présentée. Au moment de l'écriture de cette thèse, cette méthode, 3DLabelProp, obtient les meilleures performances de généralisation de la littérature. En particulier, son extrême résilience au *sensor shift* lui permet d'être capable de segmenter des nuages de points acquis avec des capteurs fortement différents de ceux ayant permis l'acquisition des données d'entraînement. 3DLabelProp a aussi été étendue à la tâche de MOS, et est parvenue à rester compétitive dans le cas monosource et pour la généralisation de domaine.

L'intérêt de la généralisation de domaine a aussi été étendu au cas des performances monodomaine. Dans le chapitre 5, il a été vu que la tâche de généralisation multisource pouvait être utilisée dans le cadre de préentraînement des méthodes de segmentation sémantique et que les caractéristiques apprises pour la généralisation de domaine se transfèrent naturellement pour le *finetuning* vers de nouveaux jeux de données.

Finalement, j'espère avoir réussi à démontrer et convaincre de l'intérêt de la tâche de généralisation de domaine pour la segmentation sémantique de données LiDAR pour le véhicule autonome. Je crois que les outils et analyses présentés dans ce travail pourront constituer des bases solides pour les travaux suivants.

7.2 Développements futurs

Ce travail a eu pour vocation d'être le plus exhaustif possible, pour comprendre au mieux les interactions entre la littérature actuelle de segmentation sémantique et la tâche de généralisation de domaine. Cependant, il a été impossible de réaliser la totalité des travaux nécessaires à l'établissement d'un panorama intégral de la généralisation de domaine pour la segmentation de données LiDAR.

D'abord, ce travail n'a pas inclus l'analyse des méthodes de généralisation de domaine, mono- et multisource provenant de l'imagerie traditionnelle (dont [94, 95, 96, 104]). Certaines idées pourraient être utilisées, notamment dans le cadre monosource. Cependant, la taille excessive des réseaux 3D rend l'implémentation naïve de ces méthodes impossible et mérite donc un travail plus ambitieux.

L'établissement d'un panorama des performances de généralisation de la segmentation panoptique est l'étape suivante naturelle à ce travail. La généralisation de domaine pour la détection d'objets a déjà été partiellement traitée, et l'association des techniques de détection d'objets à celle de segmentation sémantique pour réaliser de la segmentation panoptique pourrait être étudiée.

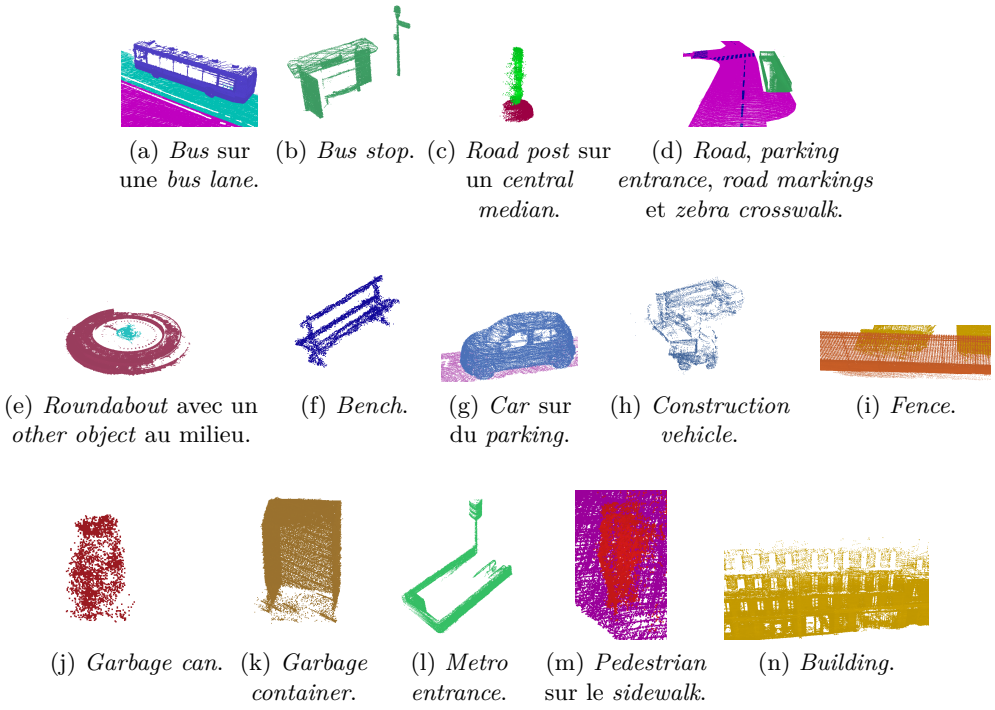
Pour finir, l'objectif final des travaux de généralisation de domaine, comme ceux d'adaptation de domaine, est d'être capable de passer de données d'inférence synthétiques à des données d'évaluation réelles. Les données synthétiques sont bien plus simples et moins coûteuses à acquérir et à annoter. Ce travail ne s'est pas intéressé à la question de l'utilisation de jeux de données synthétiques en source, cette question étant plus complexe que le cas réel vers réel, mais c'est un sujet d'intérêt avec des applications industrielles majeures.

Annexes

Annexe A

Détails sur ParisLuco3D

Des exemples des 42 labels de ParisLuco3D sont montrés sur la figure A.1 et la figure A.2.

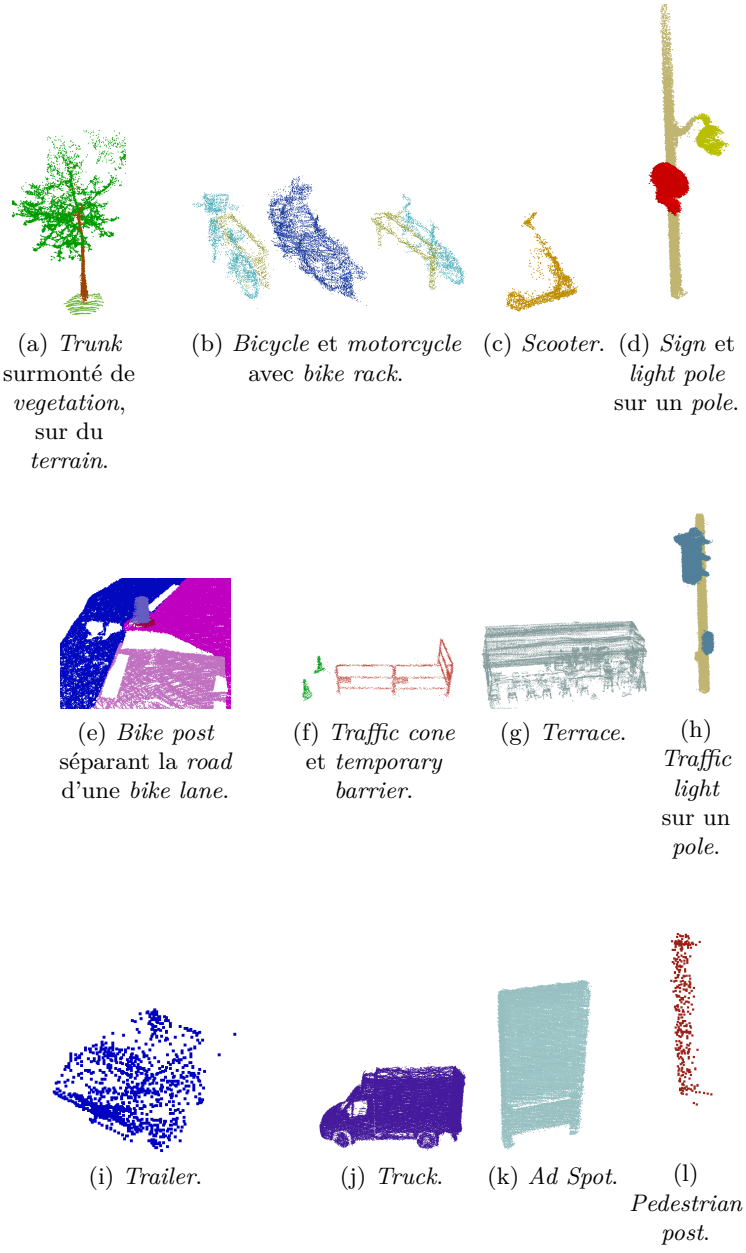


(a) *Bus* sur une *bus lane*. (b) *Bus stop*. (c) *Road post* sur un *central median*. (d) *Road*, *parking entrance*, *road markings* et *zebra crosswalk*.

(e) *Roundabout* avec un *other object* au milieu. (f) *Bench*. (g) *Car* sur du *parking*. (h) *Construction vehicle*. (i) *Fence*.

(j) *Garbage can*. (k) *Garbage container*. (l) *Metro entrance*. (m) *Pedestrian* sur le *sidewalk*. (n) *Building*.

FIGURE A.1 – Annotations de ParisLuco3D.



(a) *Trunk* surmonté de *vegetation*, sur du *terrain*.

(b) *Bicycle* et *motorcycle* avec *bike rack*.

(c) *Scooter*.

(d) *Sign* et *light pole* sur un *pole*.

(e) *Bike post* séparant la *road* d'une *bike lane*.

(f) *Traffic cone* et *temporary barrier*.

(g) *Terrace*.

(h) *Traffic light* sur un *pole*.

(i) *Trailer*.

(j) *Truck*.

(k) *Ad Spot*.

(l) *Pedestrian post*.

FIGURE A.2 – Annotations de ParisLuco3D (suite).

Annexe B

Résultats exhaustifs de généralisation monosource

	Car	Bicycle	Motorcycle	Truck	Other-vehicle	Person	Bicyclist	Motorcyclist	Road	Parking	sidewalk	Other-ground	building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic-sign	mIoU
CENet	91,3	24,8	60,6	81,8	57,1	56,9	76,5	0,0	93,2	51,6	78,7	0,1	84,6	54,9	82,9	60,1	68,6	54,6	38,1	58,8
Helix4D	96,0	21,2	63,5	68,2	61,4	64,9	73,9	0,5	93,3	39,4	79,2	1,4	88,6	54,1	88,5	60,7	77,4	61,2	46,0	60,0
KPConv	94,7	36,8	58,6	45,1	47,7	62,7	76,8	1,0	90,1	32,3	75,5	3,8	88,4	59,8	87,6	67,2	74,3	61,7	44,4	58,3
SRU-Net	96,2	7,9	51,6	65,5	51,5	64,7	64,5	0,0	93,2	48,3	80,1	0,1	91,0	62,3	88,4	67,7	75,4	62,8	43,3	58,6
SPVCNN	96,5	22,6	59,4	79,4	61,8	67,1	81,0	0,0	93,3	47,7	80,5	0,2	91,1	63,8	87,7	66,9	73,2	63,5	47,7	62,3
Cylinder3D	96,4	21,0	53,2	79,1	57,4	69,2	80,5	0,0	93,2	43,8	79,4	2,6	90,4	55,7	86,0	64,4	71,6	63,5	45,8	60,7
3DLabelProp	96,1	44,0	69,9	65,4	62,4	70,3	81,1	0,0	89,0	42,0	72,7	1,6	89,4	55,7	88,5	69,2	75,8	58,4	44,6	61,9

TABLE B.1 – Résultats monodomaine sur SemanticKITTI sur le jeu de labels $\mathcal{L}_{SK}$.

	Car	Bicycle	Motorcycle	Truck	Other-vehicle	Person	Bicyclist	Motorcyclist	Road	Parki-ng	sidewalk	Other-ground	building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic-sign	mIoU
CENet	82,7	1,8	20,8	38,2	16,4	13,1	51,2	0,0	86,7	18,5	67,4	0,2	69,7	32,6	76,3	38,4	69,9	46,5	11,7	39,1
Helix4D	93,6	20,3	42,1	35,0	52,7	56,1	63,4	2,7	89,6	24,9	73,4	3,3	86,3	48,4	86,8	56,3	73,7	58,2	43,5	53,2
KPConv	92,4	30,6	52,6	27,4	30,1	55,5	70,5	2,0	84,4	21,6	67,2	8,4	83,7	56,9	87,3	60,1	74,1	58,6	37,8	52,7
SRU-Net	94,0	7,3	45,1	57,0	43,9	54,8	53,6	0,0	90,7	37,1	75,5	1,4	88,9	54,9	87,6	62,8	74,1	60,1	37,2	54,0
SPVCNN	94,0	21,4	46,3	72,7	52,7	60,4	67,6	0,0	90,7	37,3	75,4	1,9	89,1	57,9	87,3	62,8	72,5	59,2	41,2	57,4
Cylinder3D	92,7	20,8	38,4	65,2	48,7	56,5	64,5	0,0	90,4	18,6	73,1	1,4	88,0	40,6	83,4	58,1	70,2	59,3	38,8	53,1
3DLabelProp	96,0	48,6	78,0	63,1	65,8	66,1	76,1	0,0	88,4	35,4	71,6	2,7	88,1	53,5	88,0	69,0	75,3	59,3	47,0	61,7

TABLE B.2 – Résultats de généralisation de SemanticKITTI à SemanticKITTI32 sur le jeu de labels $\mathcal{L}_{SK}$.

	2-wheeled	Pedestrian	D. Ground	Seidewalk	O.Ground	Manmade	Vegetation	4-Wheeled	mIoU
CENet	0,1	0,7	5,2	7,2	7,0	51,0	28,2	7,4	13,3
Helix4D	07,6	13,9	41,7	23,7	6,4	54,6	32,2	42,5	27,7
KPConv	6,4	24,6	44,9	25,7	15,2	51,3	40,5	53,0	32,7
SRU-Net	17,8	34,7	54,8	31,8	16,9	77,7	57,9	61,6	44,2
SPVCNN	11,0	23,8	51,2	33,9	17,5	77,1	53,4	53,4	40,2
Cylinder3D	2,7	11,9	15,5	18,4	10,4	52,2	25,1	10,9	18,4
3DLabelProp	32,8	58,3	66,5	40,7	24,9	80,2	65,9	89,0	57,3

TABLE B.3 – Résultats de généralisation de SemanticKITTI à Panda64 sur le jeu de labels $\mathcal{L}_{SK \cap PS}$.

	2-wheeled	Pedestrian	D. Ground	Seidewalk	O.Ground	Manmade	Vegetation	4-Wheeled	mIoU
CENet	0,0	0,0	4,9	1,4	2,0	11,1	21,4	3,1	4,9
Helix4D	3,5	4,0	24,0	5,4	4,7	28,4	20,4	23,5	14,2
KPConv	10,1	14,6	13,7	7,0	13,1	20,1	42,2	48,0	21,1
SRU-Net	5,6	12,1	29,4	7,2	7,1	41,9	42,2	32,2	22,2
SPVCNN	2,0	6,7	28,7	7,6	6,1	49,9	37,5	16,9	19,4
Cylinder3D	0,0	1,4	1,9	4,8	8,4	20,9	13,2	1,4	6,5
3DLabelProp	45,8	61,3	79,9	30,2	18,4	72,7	73,6	92,8	59,3

TABLE B.4 – Résultats de généralisation de SemanticKITTI à PandaFF sur le jeu de labels $\mathcal{L}_{SK \cap PS}$.

	Person	Rider	Bike	Car	Ground	Trunk	Vegetation	Traffic-sign	Pole	Building	Fence	mIoU
CENet	2,9	40,2	0,0	31,9	73,9	33,3	65,6	22,2	36,5	77,6	37,0	27,9
Helix4D	27,8	18,4	4,0	59,7	64,4	27,7	58,0	24,3	29,2	66,5	16,4	36,0
KPConv	43,9	32,7	8,6	63,4	74,8	31,4	59,7	7,6	32,6	57,6	17,6	39,1
SRU-Net	38,9	26,1	2,9	82,9	75,1	35,5	65,6	15,5	41,6	77,1	37,3	45,3
SPVCNN	44,6	18,1	4,6	83,2	75,0	36,0	65,8	18,5	41,6	75,8	36,0	45,4
Cylinder3D	44,6	13,6	2,8	81,5	73,6	33,2	61,6	13,1	36,0	71,3	30,1	41,9
3DLabelProp	53,6	34,5	10,0	87,7	74,1	30,5	65,6	19,5	40,0	73,9	30,2	47,2

TABLE B.5 – Résultats de généralisation de SemanticKITTI à SemanticPOSS sur le jeu de labels $\mathcal{L}_{SK \cap SP}$.

	Car	Bicycle	Motorcycle	Truck	Vegetation	Sidewalk	Road	Person	Bicyclist	Motorcyclist	Trunk	Other-vehicle	Sign	Pole	Building	Other-Ground	mIoU
CENet	6,8	0,0	0,0	0,0	32,1	0,9	0,0	1,0	0,0	0,0	12,3	0,6	5,3	13,2	39,2	6,5	7,4
KPConv	35,8	4,3	1,6	0,8	55,2	27,6	34,1	31,1	1,4	0,0	8,3	1,5	8,5	18,1	29,5	21,7	17,5
SRU-Net	64,7	0,1	15,9	17,0	72,1	32,3	60,6	40,3	16,2	0,0	41,7	8,1	35,4	21,4	80,0	23,5	33,1
SPVCNN	58,9	0,7	12,7	17,4	69,6	32,5	48,4	32,6	5,4	0,0	39,4	7,4	31,9	22,4	73,2	22,1	29,7
Cylinder3D	22,5	1,5	7,6	3,8	41,0	27,9	58,1	11,9	4,8	0,0	25,6	7,1	6,5	7,5	60,5	16,0	18,9
3DLabelProp	85,3	21,8	13,2	6,8	78,0	34,3	59,9	22,8	0,0	41,7	10,5	45,6	37,0	85,4	28,9	39,4	

TABLE B.6 – Résultats de généralisation de SemanticKITTI à Waymo sur le jeu de labels $\mathcal{L}_{SK \cap W}$.

	Motorcycle	Bicycle	Person	Road	Sidewalk	O. Ground	Manmade	Vegetation	Car	Terrain	mIoU
CENet	0,0	0,0	0,0	0,8	0,2	1,3	24,6	16,8	6,0	0,2	5,0
Helix4D	7,5	2,8	23,2	75,7	31,7	3,8	57,2	58,5	53,7	28,5	34,3
KPConv	33,7	7,8	52,8	80,3	34,9	4,0	70,7	70,5	74,3	37,8	46,7
SRU-Net	17,1	2,4	24,8	87,8	39,4	5,1	74,7	72,6	78,5	25,5	42,8
SPVCNN	35,9	3,0	27,3	88,3	39,6	6,6	75,0	72,4	78,2	24,7	45,1
Cylinder3D	3,3	2,2	1,0	77,3	32,0	7,6	67,3	60,6	56,2	19,8	32,7
3DLabelProp	29,3	8,1	38,9	81,5	35,0	3,2	69,2	68,4	76,4	45,9	45,6

TABLE B.7 – Résultats de généralisation de SemanticKITTI à nuScenes sur le jeu de labels$\mathcal{L}_{SK \cap NS}$.

	barrier	bicycle	bus	car	cnstrctn-vhcl	motorcycle	pedestrian	traffic-cone	trailer	truck	drvbl-grnd	other-flat	sidewalk	terrain	manmade	vegetation	mIoU
CENet	57,3	32,4	88,7	81,8	39,4	72,6	61,4	50,3	67,0	79,4	95,4	70,1	71,5	70,7	85,3	82,8	69,1
Helix4D	78,7	16,4	80,6	91,3	55,7	65,1	65,6	24,3	70,2	78,6	96,1	66,0	75,5	73,8	86,4	84,8	69,3
KPConv	66,2	17,8	72,4	87,8	26,3	67,8	69,8	51,5	25,6	73,5	93,8	53,6	66,3	71,2	83,0	82,8	63,1
SRU-Net	71,4	9,2	83,1	87,7	27,3	73,1	69,5	48,5	45,5	78,3	95,0	62,7	69,0	71,7	84,6	83,4	66,3
SPVCNN	72,6	14,0	82,9	88,7	32,2	73,4	69,9	47,8	46,1	78,3	95,0	64,4	69,4	71,8	84,6	83,4	67,2
Cylinder3D	71,5	29,4	84,3	86,4	40,5	70,5	72,9	54,3	57,2	79,7	96,1	65,8	71,9	71,6	86,4	85,0	70,2
3DLabelProp	74,1	44,4	85,4	85,4	37,4	81,6	74,9	59,4	49,3	78,7	94,1	63,3	69,8	73,0	87,6	86,1	71,5

TABLE B.8 – Résultats monodomaine sur nuScenes sur le jeu de labels $\mathcal{L}_{NS}$.

	Motorcycle	Bicycle	Person	Road	Sidewalk	O. Ground	Manmade	Vegetation	Car	Terrain	mIoU
CENet	0,0	0,0	0,2	26,6	2,6	0,0	19,3	14,5	07,1	29,9	10,0
Helix4D	12,0	18,5	13,8	72,1	51,8	0,2	63,5	71,4	54,2	42,5	40,0
KPConv	18,9	3,9	49,8	67,2	28,8	0,4	73,4	81,8	79,0	45,8	44,9
SRU-Net	17,8	2,3	46,3	71,1	45,8	0,0	72,3	78,8	85,0	43,6	46,3
SPVCNN	25,7	11,6	47,7	69,8	46,3	0,0	77,1	81,0	90,0	45,2	49,4
Cylinder3D	3,7	0,0	21,0	54,3	21,5	0,0	65,0	70,9	46,1	34,4	31,7
3DLabelProp	44,5	35,0	50,5	80,9	63,1	0,4	80,3	85,0	91,0	67,4	59,8

TABLE B.9 – Résultats de généralisation de from nuScenes à SemanticKITTI sur le jeu de labels $\mathcal{L}_{NS \cap SK}$.

	Motorcycle	Bicycle	Person	Road	Sidewalk	O. Ground	Manmade	Vegetation	Car	Terrain	mIoU
CENet	36,9	19,2	30,8	78,8	56,0	0,1	63,1	73,9	78,8	57,9	49,6
Helix4D	16,0	18,0	16,7	77,1	55,0	0,2	67,9	74,5	62,8	52,6	44,1
KPConv	23,1	3,5	46,3	78,9	48,6	1,9	75,9	83,7	81,1	62,7	50,6
SRU-Net	22,9	2,4	45,8	81,3	49,5	0,1	82,8	85,9	92,0	61,1	52,4
SPVCNN	28,6	8,3	45,0	81,4	50,9	0,2	81,6	84,5	91,5	60,0	53,2
Cylinder3D	14,0	5,6	37,5	75,8	48,0	0,0	74,5	77,9	78,8	52,0	46,4
3DLabelProp	48,1	40,6	57,4	83,9	64,1	0,3	82,0	85,9	90,9	68,0	62,1

TABLE B.10 – Résultats de généralisation de nuScenes à SemanticKITTI32 sur le jeu de labels $\mathcal{L}_{NS \cap SK}$.

	2-wheeled	Pedestrian	D. Ground	Seidewalk	O.Ground	Manmade	Vegetation	4-Wheeled	mIoU
CENet	0,0	1,3	8,1	2,7	5,6	36,7	11,4	12,9	9,8
Helix4D	1,9	12,6	8,7	0,4	18,8	52,8	2,0	11,4	13,6
KPConv	3,9	6,2	20,3	17,1	10,8	64,5	53,2	24,2	25,0
SRU-Net	4,7	26,1	36,4	22,3	12,3	69,7	51,1	42,2	33,1
SPVCNN	19,2	43,1	38,0	30,6	15,3	71,9	62,1	69,4	43,6
Cylinder3D	0,3	0,8	13,3	6,9	7,8	58,9	30,5	8,3	15,8
3DLabelProp	50,6	70,0	73,7	50,2	28,4	88,8	78,1	90,0	66,2

TABLE B.11 – Résultats de généralisation de nuScenes à Panda64 sur le jeu de labels $\mathcal{L}_{NS\cap PS}$.

	2-wheeled	Pedestrian	D. Ground	Seidewalk	O.Ground	Manmade	Vegetation	4-Wheeled	mIoU
CENet	0,0	0,8	1,5	0,3	2,2	20,0	14,9	8,5	6,0
Helix4D	1,4	2,1	8,1	0,0	11,0	31,7	3,8	2,8	7,6
KPConv	5,0	6,8	7,0	7,5	4,6	38,2	48,7	17,8	16,9
SRU-Net	0,0	2,0	3,3	2,7	6,7	30,6	24,8	8,2	9,8
SPVCNN	1,4	7,4	3,0	3,2	8,0	28,9	28,5	8,5	11,1
Cylinder3D	0,0	0,0	1,6	0,6	2,4	24,3	7,8	0,9	4,7
3DLabelProp	57,9	66,1	92,6	48,4	32,4	86,5	84,8	91,0	70,0

TABLE B.12 – Résultats de généralisation de nuScenes à PandaFF sur le jeu de labels $\mathcal{L}_{NS\cap PS}$.

	Person	Bike	Car	Ground	Vegetation	Manmade	mIoU
CENet	0,0	0,0	1,1	0,0	0,0	18,4	3,3
Helix4D	35,9	6,3	52,7	64,7	53,7	57,8	45,2
KPConv	56,0	17,2	72,4	73,8	70,0	74,5	60,7
SRU-Net	60,9	10,9	77,1	74,1	69,5	76,6	61,5
SPVCNN	64,5	15,5	85,6	74,2	71,8	78,1	64,8
Cylinder3D	22,9	0,6	29,5	74,1	63,0	66,9	42,8
3DLabelProp	68,4	14,0	82,1	72,9	72,0	78,2	64,6

TABLE B.13 – Résultats de généralisation de nuScenes à SemanticPOSS sur le jeu de labels $\mathcal{L}_{NS\cap SP}$.

	car	truck	bus	other vhcle	motorcycle	bicycle	pedestrian	traffic cone	manmade	vegetation	dvbl grd	sidewalk	other ground	mIoU
CENet	0,0	0,8	0,3	0,2	0,0	0,0	1,0	0,2	27,2	10,8	0,4	0,9	3,5	3,5
KPConv	5,2	0,6	0,9	1,5	0,5	0,1	3,0	0,0	52,1	56,3	32,2	17,2	27,5	15,2
SRU-Net	26,1	7,7	3,8	5,6	3,9	0,2	30,5	0,1	73,8	64,1	35,8	27,1	28,3	23,6
SPVCNN	61,4	12,7	24,9	5,5	16,1	2,6	54,0	2,3	80,1	76,3	62,0	46,3	39,1	37,2
Cylinder3D	0,2	1,1	0,0	1,1	0,0	0,0	1,3	0,4	59,7	34,3	31,1	16,0	19,8	12,7
3DLabelProp	77,1	18,2	34,6	4,6	29,3	28,1	72,8	8,7	88,2	85,7	74,6	47,7	43,1	47,1

TABLE B.14 – Résultats de généralisation de nuScenes à Waymo sur le jeu de labels $\mathcal{L}_{NS\cap W}$.

Annexe C

Résultats exhaustifs de généralisation multisource

Méthode	Driveable Ground	Structure	Vehicle	Nature	Pedestrian	Object	mIoU
KPConv	9,3	57,1	38,7	56,3	1,2	1,5	27,3
KPConv multisource	60,7	78,3	50,0	66,7	55,2	2,9	52,3
Cylinder3D	5,7	74,2	37,8	58,7	13,3	2,0	32,0
Cylinder3D multisource	61,5	82,7	57,1	69,8	63,1	5,5	56,6
SRU-Net	30,3	70,3	44,5	62,5	7,6	3,3	35,3
SRU-Net multisource	63,1	75,6	62,9	69,9	59,4	16,5	57,7

TABLE C.1 – Détail du benchmark de généralisation de domaine multisource pour SemanticPOSS.

Méthode	Driveable Ground	Structure	Vehicle	Nature	Pedestrian	Object	Other Ground	mIoU
KPConv	49,5	70,0	43,6	53,5	0,8	5,1	50,7	39,0
KPConv multisource	65,2	77,6	41,8	70,6	40,7	0,8	53,3	50,0
Cylinder3D	62,2	71,7	35,3	47,7	1,6	5,3	63,1	41,0
Cylinder3D multisource	88,3	83,3	78,6	77,8	59,2	3,8	77,7	67,0
SRU-Net	66,6	63,3	50,8	50,3	10,6	12,4	56,1	44,3
SRU-Net multisource	73,7	80,9	67,9	70,8	53,0	4,5	66,1	59,6

TABLE C.2 – Détail du benchmark de généralisation de domaine multisource pour ParisLuco3D.

Méthode	Driveable Ground	Structure	Vehicle	Nature	Pedestrian	Object	Other Ground	mIoU
KPConv	39,3	42,5	29,8	38,7	0,4	4,6	22,3	25,4
KPConv multisource	52,8	67,0	81,5	70,8	65,6	13,6	41,0	56,0
Cylinder3D	43,8	55,2	32,0	28,9	12,7	4,5	26,3	29,1
Cylinder3D multisource	76,0	70,1	75,8	62,9	64,0	17,2	52,9	59,8
SRU-Net	56,2	69,3	62,8	54,9	12,5	11,8	32,8	42,9
SRU-Net multisource	78,8	73,1	86,0	70,7	60,0	15,0	53,6	62,5

TABLE C.3 – Détail du benchmark de généralisation de domaine multisource pour Panda64.

Méthode	Driveable Ground	Structure	Vehicle	Nature	Pedestrian	Object	Other Ground	mIoU
KPConv	15,0	12,0	27,6	36,9	1,0	2,2	11,0	15,1
KPConv multisource	61,1	51,7	79,6	66,3	51,5	17,2	34,6	51,7
Cylinder3D	5,2	24,3	5,9	10,5	1,0	4,2	2,6	7,6
Cylinder3D multisource	40,0	27,5	27,8	35,0	35,8	11,7	20,4	28,3
SRU-Net	55,8	63,8	74,2	65,9	18,2	6,9	14,7	42,8
SRU-Net multisource	73,3	68,4	88,6	77,1	54,5	16,7	26,0	57,8

TABLE C.4 – Détail du benchmark de généralisation de domaine multisource pour PandaFF.

Bibliographie

[1] Jessica Van Brummelen, Marie O'Brien, Dominique Gruyer, and Homayoun Najjaran. Autonomous vehicle perception : The technology of today and tomorrow. *Transportation research part C : emerging technologies*, 89 :384–406, 2018.

[2] Dominique Gruyer, Valentin Magnier, Karima Hamdi, Laurene Claussmann, Olivier Orfila, and Andry Rakotonirainy. Perception, information processing and modeling : Critical stages for autonomous driving applications. *Annual Reviews in Control*, 44 :323–341, 2017.

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jürgen Gall. Semantickitti : A dataset for semantic scene understanding of lidar sequences. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9296–9306, 2019.

[4] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59 :167–181, 2004.

[5] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence*, 16(6) :641–647, 1994.

[6] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes : Active contour models. *International journal of computer vision*, 1(4) :321–331, 1988.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[8] Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256, 2010.

[9] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets : Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words : Transformers for image recognition at scale. *arXiv preprint arXiv :2010.11929*, 2020.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations (ICLR 2015)*, pages 1–14, 2015.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[16] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015.

[17] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015 : 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[19] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[20] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. 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, 2018.

[21] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[22] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6881–6890, 2021.

[23] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer : Simple and efficient design for semantic segmentation with transformers. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[24] Stefan Gumhold, Xinlong Wang, Rob S MacLeod, et al. Feature extraction from point clouds. In *IMR*, pages 293–305. Citeseer, 2001.

[25] Joachim Niemeyer, Franz Rottensteiner, and Uwe Soergel. Contextual classification of lidar data and building object detection in urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 87 :152–165, 2014.

[26] Martin Weinmann, Boris Jutzi, and Clément Mallet. Semantic 3d scene interpretation : A framework combining optimal neighborhood size selection with relevant features. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(3) :181, 2014.

[27] R Blomley, M Weinmann, J Leitloff, and B Jutzi. Shape distribution features for point cloud analysis-a geometric histogram approach on multiple scales. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(3) :9, 2014.

[28] Daniel Maturana and Sebastian Scherer. Voxnet : A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[29] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet : Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586, 2017.

[30] Benjamin Graham. Sparse 3d convolutional neural networks. In *British Machine Vision Conference*, 2015.

[31] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018.

[32] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg : Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.

[33] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars : Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[34] Larissa T Triess, David Peter, Christoph B Rist, and J Marius Zöllner. Scan-based semantic segmentation of lidar point clouds : An experimental study. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1116–1121. IEEE, 2020.

[35] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet : Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[36] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++ : Deep hierarchical feature learning on point sets in a metric space. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30 : Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5099–5108, 2017.

[37] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv : Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019.

[38] Timo Hackel, N. Savinov, L. Ladicky, Jan D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3D.NET : A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-1-W1, pages 91–98, 2017.

[39] Mingmei Cheng, Le Hui, Jin Xie, Jian Yang, and Hui Kong. Cascaded non-local neural network for point cloud semantic segmentation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8447–8452, 2020.

[40] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl : Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5589–5598, 2020.

[41] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet : Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1607–1616, 2019.

[42] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn : Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.

[43] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018.

[44] Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S Davis. Modeling local geometric structure of 3d point clouds using geo-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 998–1008, 2019.

[45] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–66, 2018.

[46] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[47] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10296–10305, 2019.

[48] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net : Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020.

[49] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++ : Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019.

[50] Joseph Redmon and Ali Farhadi. Yolov3 : An incremental improvement. *arXiv preprint arXiv :1804.02767*, 2018.

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

[52] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3 : Spatially-adaptive convolution for efficient point-cloud segmentation. In *European Conference on Computer Vision*, pages 1–19. Springer, 2020.

[53] Deyvid Kochanov, Fatemeh Karimi Nejadasl, and Olaf Booij. Kprnet : Improving projection-based lidar semantic segmentation. *arXiv preprint arXiv :2007.12668*, 2020.

[54] Larissa T Triess, David Peter, Christoph B Rist, and J Marius Zöllner. Scan-based semantic segmentation of lidar point clouds : An experimental study. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1116–1121. IEEE, 2020.

[55] Hui-Xian Cheng, Xian-Feng Han, and Guo-Qiang Xiao. Cenet : Toward concise and efficient lidar semantic segmentation for autonomous driving. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 01–06. IEEE, 2022.

[56] Shijie Li, Yun Liu, and Juergen Gall. Rethinking 3-d lidar point cloud segmentation. *IEEE transactions on neural networks and learning systems*, PP, December 2021.

[57] Angelika Ando, Spyros Gidaris, Andrei Bursuc, Gilles Puy, Alexandre Boulch, and Renaud Marlet. Rangevit : Towards vision transformers for 3d semantic segmentation in autonomous driving, 2023.

[58] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn : Point-voxel feature set abstraction for 3d object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10526–10535, 2020.

[59] Eren Erdal Aksoy, Saimir Baci, and Selcuk Cavdar. Salsanet : Fast road and vehicle segmentation in lidar point clouds for autonomous driving. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 926–932. IEEE, 2019.

[60] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet : An improved grid representation for online lidar point clouds semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9610, 2020.

[61] Martin Gerdzhev, Ryan Razani, Ehsan Taghavi, and Liu Bingbing. Tornado-net : multi-view total variation semantic segmentation with diamond inception module. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9543–9549, 2021.

[62] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Using a waffle iron for automotive point cloud semantic segmentation. *arXiv preprint arXiv :2301.10100*, 2023.

[63] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. Classification of point cloud for road scene understanding with multiscale voxel deep network. In *10th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, PPNIV'18*, October 2018.

[64] Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu. (af)2-s3net : Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12542–12551, 2021.

[65] Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3d : An effective 3d framework for driving-scene lidar semantic segmentation. *CoRR*, abs/2008.01550, 2020.

[66] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Wei Li, Yuexin Ma, Hongsheng Li, Ruigang Yang, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar-based perception. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10) :6807–6822, 2022.

[67] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguang Cui. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3101–3109, 2021.

[68] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[69] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European Conference on Computer Vision*, pages 685–702. Springer, 2020.

[70] Feihu Zhang, Jin Fang, Benjamin Wah, and Philip Torr. Deep fusionnet for point cloud semantic segmentation. In *ECCV*, volume 2, page 6, 2020.

[71] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. Rpvnet : A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16004–16013, 2021.

[72] Hanwen Cao, Yongyi Lu, Cewu Lu, Bo Pang, Gongshen Liu, and Alan Yuille. Asap-net : Attention and structure aware point cloud sequence segmentation. In *British Machine Vision Conference (BMVC)*, 2020.

[73] Hanyu Shi, Guosheng Lin, Hao Wang, Tzu-Yi Hung, and Zhenhua Wang. Spsequencenet : Semantic segmentation network on 4d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4574–4583, 2020.

[74] Fabian Duerr, Mario Pfaller, Hendrik Weigel, and Jürgen Beyerer. Lidar-based recurrent 3d semantic segmentation with temporal memory alignment. In *2020 International Conference on 3D Vision (3DV)*, pages 781–790, 2020.

[75] Romain Loiseau, Mathieu Aubry, and Loïc Landrieu. Online segmentation of lidar sequences : Dataset and algorithm. In *Computer Vision–ECCV 2022 : 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVIII*, pages 301–317. Springer, 2022.

[76] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla : An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[77] Jean-Emmanuel Deschaud. Kitti-carla : a kitti-like dataset generated by carla simulator. *arXiv preprint arXiv :2109.00892*, 2021.

[78] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360 : A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3) :3292–3310, 2023.

[79] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes : A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020.

[80] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurélien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving : Waymo open dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2443–2451, 2020.

[81] Yancheng Pan, Biao Gao, Jilin Mei, Sibo Geng, Chengkun Li, and Huijing Zhao. Semanticposs : A point cloud dataset with large quantity of dynamic instances. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 687–693, 2020.

[82] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, et al. Pandaset : Advanced sensor suite dataset for autonomous driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3095–3101. IEEE, 2021.

[83] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving ? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[84] Pierre Dellenbach, Jean-Emmanuel Deschaud, Bastien Jacquet, and François Goulette. Ct-icp : Real-time elastic lidar odometry with loop closure. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5580–5586, 2022.

[85] Li Yi, Boqing Gong, and Thomas Funkhouser. Complete & label : A domain adaptation approach to semantic segmentation of lidar point clouds. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15358–15368, 2021.

[86] Cristiano Saltori, Evgeny Krivosheev, Stéphane Lathuiliére, Nicu Sebe, Fabio Galasso, Giuseppe Fiameni, Elisa Ricci, and Fabio Poiesi. Gipso : Geometrically informed propagation for online adaptation in 3d lidar segmentation. In *European Conference on Computer Vision*, pages 567–585. Springer, 2022.

[87] Hyeonseong Kim, Yoonsu Kang, Changgyoon Oh, and Kuk-Jin Yoon. Single domain generalization for lidar semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17587–17598, 2023.

[88] Jules Sanchez, Jean-Emmanuel Deschaud, and Francois Goulette. Domain generalization of 3d semantic segmentation in autonomous driving, 2023.

[89] Cristiano Saltori, Fabio Galasso, Giuseppe Fiameni, Nicu Sebe, Elisa Ricci, and Fabio Poiesi. Cosmix : Compositional semantic mix for domain adaptation in 3d lidar segmentation. In *European Conference on Computer Vision*, pages 586–602. Springer, 2022.

[90] Ke Wang, Liang Pu, Jian Zhang, and Jianbo Lu. Gated adversarial network based environmental enhancement method for driving safety under adverse weather conditions. *IEEE Transactions on Intelligent Vehicles*, 8(2) :1934–1943, 2023.

[91] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains : A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2022.

[92] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization : A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2022.

[93] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[94] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize : Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[95] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy Hospedales. Episodic training for domain generalization. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1446–1455, 2019.

[96] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pages 2551–2559, 2015.

[97] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019.

[98] Zhenlin Xu, Deyi Liu, Junlin Yang, Colin Raffel, and Marc Niethammer. Robust and generalizable visual representation learning via random convolutions. In *International Conference on Learning Representations*, 2021.

[99] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. *Advances in neural information processing systems*, 31, 2018.

[100] Zakria, Jianhua Deng, Jingye Cai, Muhammad Umar Aftab, Muhammad Saddam Khokhar, and Rajesh Kumar. Visual features with spatio-temporal-based fusion model for cross-dataset vehicle re-identification. *Electronics*, 9(7), 2020.

[101] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[102] Xin Jin, Cuiling Lan, Wenjun Zeng, and Zhibo Chen. Style normalization and restitution for domain generalization and adaptation. *IEEE Transactions on Multimedia*, 24 :3636–3651, 2022.

[103] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once : Enhancing learning and generalization capacities via ibn-net. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 464–479, 2018.

[104] Toshihiko Matsuura and Tatsuya Harada. Domain generalization using a mixture of multiple latent domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11749–11756, 2020.

[105] Isabela Albuquerque, João Monteiro, Mohammad Darvishi, Tiago H Falk, and Ioannis Mitliagkas. Generalizing to unseen domains via distribution matching. *arXiv preprint arXiv :1911.00804*, 2019.

[106] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5715–5725, 2017.

[107] Alexander Lehner, Stefano Gasperini, Alvaro Marcos-Ramiro, Michael Schmidt, Mohammad-Ali Nikouei Mahani, Nassir Navab, Benjamin Busam, and Federico Tombari. 3d-vfield : Adversarial augmentation of point clouds for domain generalization in 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17295–17304, June 2022.

[108] Cristiano Saltori, Aljoša Ošep, Elisa Ricci, and Laura Leal-Taixé. Walking your lidog : A journey through multiple domains for lidar semantic segmentation, 2023.

[109] Xu Yan, Chaoda Zheng, Zhen Li, Shuguang Cui, and Dengxin Dai. Benchmarking the robustness of lidar semantic segmentation models. *ArXiv*, abs/2301.00970, 2023.

[110] Sicheng Zhao, Yezhen Wang, Bo Li, Bichen Wu, Yang Gao, Pengfei Xu, Trevor Darrell, and Kurt Keutzer. epointda : An end-to-end simulation-to-real domain adaptation framework for lidar point cloud segmentation. In *AAAI*, 2021.

[111] Aoran Xiao, Jiaxing Huang, Dayan Guan, Fangneng Zhan, and Shijian Lu. Transfer learning from synthetic to real lidar point cloud for semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2795–2803, 2022.

[112] Ferdinand Langer, Andres Milioto, Alexandre Haag, Jens Behley, and Cyrill Stachniss. Domain transfer for semantic segmentation of lidar data using deep neural networks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8263–8270, 2020.

[113] Bjoern Michele, Alexandre Boulch, Gilles Puy, Tuan-Hung Vu, Renaud Marlet, and Nicolas Courty. Saluda : Surface-based automotive lidar unsupervised domain adaptation, 2023.

[114] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[115] Yuhui Yuan, Jingyi Xie, Xilin Chen, and Jingdong Wang. Segfix : Model-agnostic boundary refinement for segmentation. pages 489–506, 2020.

[116] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv preprint arXiv :2005.10821*, 2020.

[117] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 405–420, 2018.

[118] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork convnets for video semantic segmentation. In *European Conference on Computer Vision*, pages 852–868. Springer, 2016.

[119] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2349–2358, 2017.

[120] Yu-Syuan Xu, Tsu-Jui Fu, Hsuan-Kung Yang, and Chun-Yi Lee. Dynamic video segmentation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6556–6565, 2018.

[121] Samvit Jain, Xin Wang, and Joseph E. Gonzalez. Accel : A corrective fusion network for efficient semantic segmentation on video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[122] Hanwen Cao, Yongyi Lu, Cewu Lu, Bo Pang, Gongshen Liu, and Alan Yuille. Asap-net : Attention and structure aware point cloud sequence segmentation. *arXiv preprint arXiv :2008.05149*, 2020.

[123] Hanyu Shi, Guosheng Lin, Hao Wang, Tzu-Yi Hung, and Zhenhua Wang. Spsequencenet : Semantic segmentation network on 4d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4574–4583, 2020.

[124] Fabian Duerr, Mario Pfaller, Hendrik Weigel, and Jürgen Beyerer. Lidar-based recurrent 3d semantic segmentation with temporal memory alignment. In *2020 International Conference on 3D Vision (3DV)*, pages 781–790. IEEE, 2020.

[125] Benedikt Mersch, Xieyuanli Chen, Ignacio Vizzo, Lucas Nunes, Jens Behley, and Cyrill Stachniss. Receding moving object segmentation in 3d lidar data using sparse 4d convolutions. *IEEE Robotics and Automation Letters*, 7(3) :7503–7510, 2022.

[126] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet : Deep learning on dynamic 3d point cloud sequences. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9245–9254, 2019.

[127] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. Pstnet : Point spatio-temporal convolution on point cloud sequences. In *International Conference on Learning Representations*, 2021.

[128] Mehmet Aygun, Aljosa Osep, Mark Weber, Maxim Maximov, Cyrill Stachniss, Jens Behley, and Laura Leal-Taixe. 4d panoptic lidar segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5527–5537, June 2021.

[129] Lars Kreuzberg, Idil Esen Zulfikar, Sabarinath Mahadevan, Francis Engelmann, and Bastian Leibe. 4d-stop : Panoptic segmentation of 4d lidar using spatio-temporal object proposal generation and aggregation. In *European Conference on Computer Vision Workshop*, 2022.

[130] Germán Ros, Simon Stent, Pablo Fernández Alcantarilla, and Tomoki Watanabe. Training constrained deconvolutional networks for road scene semantic segmentation. *ArXiv*, abs/1604.01545, 2016.

[131] Petra Bevandić, Marin Oršić, Ivan Grubišić, Josip Šarić, and Siniša Šegvić. Multi-domain semantic segmentation with overlapping labels. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2422–2431, 2022.

[132] Xiangyun Zhao, Samuel Schulter, Gaurav Sharma, Yi-Hsuan Tsai, Manmohan Chandraker, and Ying Wu. Object detection with a unified label space from multiple datasets. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 178–193, Cham, 2020. Springer International Publishing.

[133] Marco Leonardi, Davide Mazzini, and Raimondo Schettini. Training efficient semantic segmentation cnns on multiple datasets. In Elisa Ricci, Samuel Rota Bulò, Cees Snoek, Oswald Lanz, Stefano Messelodi, and Nicu Sebe, editors, *Image Analysis and Processing – ICIAP 2019*, pages 303–314, Cham, 2019. Springer International Publishing.

[134] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Simple multi-dataset detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7571–7580, 2022.

[135] Girish Varma, Anbumani Subramanian, Anoop Namboodiri, Manmohan Chandraker, and C.V. Jawahar. Idd : A dataset for exploring problems of autonomous navigation in unconstrained environments. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1743–1751, 2019.

[136] Panagiotis Meletis and Gijs Dubbelman. Training of convolutional networks on multiple heterogeneous datasets for street scene semantic segmentation. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1045–1050, 2018.

[137] Xiaodan Liang, Eric Xing, and Hongfei Zhou. Dynamic-structured semantic propagation network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 752–761, 2018.

[138] Petra Bevandić, Ivan Krešo, Marin Oršić, and Siniša Šegvić. Simultaneous semantic segmentation and outlier detection in presence of domain shift. In Gernot A. Fink, Simone Frintrop, and Xiaoyi Jiang, editors, *Pattern Recognition*, pages 33–47, Cham, 2019. Springer International Publishing.

[139] John Lambert, Zhuang Liu, Ozan Sener, James Hays, and Vladlen Koltun. Mseg : A composite dataset for multi-domain semantic segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2876–2885, 2020.

[140] Rui Gong, Dengxin Dai, Yuhua Chen, Wen Li, and Luc Van Gool. mdalu : Multi-source domain adaptation and label unification with partial datasets. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8856–8865, 2021.

[141] Ruijia Xu, Ziliang Chen, Wangmeng Zuo, Junjie Yan, and Liang Lin. Deep cocktail network : Multi-source unsupervised domain adaptation with category shift. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 2018.

[142] Yixun Liang, Hao He, Shishi Xiao, Hao Lu, and Yingcong Chen. Label name is mantra : Unifying point cloud segmentation across heterogeneous datasets. *CoRR*, abs/2303.10585, 2023.

[143] Jules Sanchez, Jean-Emmanuel Deschaud, and François Goulette. Cola : Coarse label pre-training for 3d semantic segmentation of sparse lidar datasets. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11343–11350. IEEE, 2023.

[144] S. Horache, J. Deschaud, and F. Goulette. 3d point cloud registration with multi-scale architecture and unsupervised transfer learning. In *2021 International Conference on 3D Vision (3DV)*, pages 1351–1361, Los Alamitos, CA, USA, dec 2021. IEEE Computer Society.

[145] Fabio Poiesi and Davide Boscaini. Learning general and distinctive 3d local deep descriptors for point cloud registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3) :3979–3985, 2022.

[146] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas Guibas, and Or Litany. Pointcontrast : Unsupervised pre-training for 3d point cloud understanding. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 574–591, Cham, 2020. Springer International Publishing.

[147] Ji Hou, Benjamin Graham, Matthias Nießner, and Saining Xie. Exploring data-efficient 3d scene understanding with contrastive scene contexts. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15582–15592, 2021.

[148] Ali Thabet, Humam Alwassel, and Bernard Ghanem. Self-supervised learning of local features in 3d point clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4048–4052, 2020.

[149] Mohamed Afham, Isuru Dissanayake, Dinithi Dissanayake, Amaya Dharmasiri, Kanchana Thilakarathna, and Ranga Rodrigo. Crosspoint : Self-supervised cross-modal contrastive learning for 3d point cloud understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9902–9912, June 2022.

[150] Ryosuke Yamada, Hirokatsu Kataoka, Naoya Chiba, Yukiyasu Domae, and Tetsuya Ogata. Point cloud pre-training with natural 3d structures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21283–21293, June 2022.

[151] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert : Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[152] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet : Richly-annotated 3d reconstructions of indoor scenes. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2443, 2017.

[153] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet : An Information-Rich 3D Model Repository. Technical Report arXiv :1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[154] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets : A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[155] Hanxue Liang, Chenhan Jiang, Dapeng Feng, Xin Chen, Hang Xu, Xiaodan Liang, Wei Zhang, Zhenguo Li, and Luc Van Gool. Exploring geometry-aware contrast and clustering harmonization for self-supervised 3d object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3273–3282, 2021.

[156] Junbo Yin, Dingfu Zhou, Liangjun Zhang, Jin Fang, Cheng-Zhong Xu, Jianbing Shen, and Wenguan Wang. Proposalcontrast : Unsupervised pre-training for lidar-based 3d object detection. In *European Conference on Computer Vision*, pages 17–33. Springer, 2022.

[157] Lucas Nunes, Rodrigo Marcuzzi, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Segcontrast : 3d point cloud feature representation learning through self-supervised segment discrimination. *IEEE Robotics and Automation Letters*, 7(2) :2116–2123, 2022.

[158] Lucas Nunes, Louis Wiesmann, Rodrigo Marcuzzi, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Temporal consistent 3d lidar representation learning for semantic perception in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5217–5228, 2023.

[159] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss. Moving Object Segmentation in 3D LiDAR Data : A Learning-based Approach Exploiting Sequential Data. *IEEE Robotics and Automation Letters(RA-L)*, 2021.

[160] Jiadai Sun, Yuchao Dai, Xianjing Zhang, Jintao Xu, Rui Ai, Weihao Gu, and Xieyuanli Chen. Efficient spatial-temporal information fusion for lidar-based 3d moving object segmentation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11456–11463. IEEE, 2022.

[161] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext : Fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving, 2020.

[162] Jaeyeul Kim, Jungwan Woo, and Sunghoon Im. Rvmos : Range-view moving object segmentation leveraged by semantic and motion features. *IEEE Robotics and Automation Letters*, 7(3) :8044–8051, 2022.

[163] Qipeng Li, Yuan Zhuang, Yiwen Chen, Jianzhu Huai, Miao Li, Tianbing Ma, Yufei Tang, and Xinlian Liang. 3d-seqmos : A novel sequential 3d moving object segmentation in autonomous driving. *arXiv preprint arXiv :2307.09044*, 2023.

[164] Benedikt Mersch, Xieyuanli Chen, Ignacio Vizzo, Lucas Nunes, Jens Behley, and Cyrill Stachniss. Receding moving object segmentation in 3d lidar data using sparse 4d convolutions. *IEEE Robotics and Automation Letters*, 7(3) :7503–7510, 2022.

[165] Benedikt Mersch, Tiziano Guadagnino, Xieyuanli Chen, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Building volumetric beliefs for dynamic environments exploiting map-based moving object segmentation. *IEEE Robotics and Automation Letters*, 2023.

[166] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. L3-net : Towards learning based lidar localization for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6389–6398, 2019.

[167] Xieyuanli Chen, Benedikt Mersch, Lucas Nunes, Rodrigo Marcuzzi, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Automatic labeling to generate training data for online lidar-based moving object segmentation. *IEEE Robotics and Automation Letters*, 7(3) :6107–6114, 2022.

RÉSUMÉ

La perception LiDAR pour le véhicule autonome a atteint des résultats convenables sur les différents benchmarks en ligne dans le cadre monodomaine, c'est-à-dire quand le domaine d'entraînement est le même que celui d'évaluation. A partir de là, les champs de recherche se sont diversifiés et orientés vers les questions de transférabilité, de robustesse et de généralisation.
Ce travail se concentre sur les questions de généralisation pour la segmentation sémantique LiDAR. Un panorama global des performances de généralisation monosource et multisource des méthodes de segmentation existantes est réalisé. Pour mener équitablement ces expériences, un jeu de données, ParisLuco3D, est créé spécifiquement pour évaluer la généralisation.
De plus, une nouvelle méthode de généralisation de domaine monosource, 3DLabelProp, est proposée. Cette méthode se distingue des stratégies existantes en exploitant la géométrie des données pour effectuer de l'alignement de domaine plutôt que des stratégies par apprentissage. Au-delà de la segmentation sémantique, 3DLabelProp est aussi appliquée à la tâche de segmentation d'objets mobiles.

MOTS CLÉS

Nuage de points, segmentation sémantique, généralisation de domaine, segmentation d'objet mobile

ABSTRACT

LiDAR-based perception for autonomous vehicles has reached statisfying performance on the various online monodomain benchmarks, i.e when the training and evaluation domains are the same. From there, research areas diversified and focused on the transferability of perception models, their robstuness and their generalization capabilities.
This work tackles domain generalization of LiDAR semantic segmentation. An in-depth benchmark of the mono- and multi-source generalization abilities of various state-of-the-art semantic segmentation models is proposed. In order to provide meaningful analysis and conclusions, a generalization-oriented dataset is provided, ParisLuco3D.
Furthermore, a novel mono-source domain generalization method, 3DLabelProp, is presented. This method seperates itself from existing approaches by exploiting geometric information to achieve domain alignment instead of focusing on learning-based domain alignement. Besides semantic segmentation, 3DLabelProp is also applied to moving object segmentation.

KEYWORDS

Point clouds, semantic segmentation, domain generalization, moving object segmentation