Deep learning for change detection in 3D point clouds

Iris De Gélis

COLLEGE MATHSTIC
DOCTORAL BRETAGNE
BRETAGNE OCEANE

Université Bretagne Sud
ubs

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ BRETAGNE SUD

ÉCOLE DOCTORALE N° 644
Mathématiques et Sciences et Technologies de l'Information et de la Communication en Bretagne Océane
Spécialité : *Informatique et Architectures numériques*

Par
Iris de GÉLIS

Apprentissage profond pour la détection de changements dans des nuages de points 3D

Deep learning for change detection in 3D point clouds

Thèse présentée et soutenue à Vannes, le 13 avril 2023
Unité de recherche : Institut de Recherche en Informatique et Systèmes Aléatoires, CNRS, UMR 6074
Thèse N° : 656

Rapporteurs avant soutenance :

Christian HEIPKE	Professor, Leibniz University Hannover
Bertrand LE SAUX	Senior Researcher, European Space Agency
Clément MALLET	Senior Researcher, IGN / Université Gustave Eiffel

Composition du Jury :

Président :	Dino IENCO	Senior Researcher, INRAE
Examinateurs :	Florent POUX	Associate Professor, Université de Liège
	Charlotte PELLETIER	Associate Professor, Université Bretagne Sud
Dir. de thèse :	Sébastien LEFÈVRE	Professor, Université Bretagne Sud
Co-dir. de thèse :	Thomas CORPETTI	Senior Researcher, Centre National de la Recherche Scientifique

Invités :

Thomas RISTORCELLI	Head of "Images & Applications" unit, Magellium
Pierre LASSALLE	Remote sensing engineer, Centre National d'Études Spatiales

Remerciements

Quel moment particulier (et pas forcément facile) que de se mettre à rédiger ces remerciements, mais aussi derniers mots de ce manuscrit. Ceci est l'occasion pour moi de me remémorer ces trois années passées, ainsi que tous les moments et les personnes qui m'ont permis d'en arriver jusque-là. Tant de monde que je souhaiterais mentionner, je vous prie de m'excuser par avance si j'omets certaines personnes. J'espère aussi ne pas être trop maladroite dans ces prochaines lignes.

Premièrement, cette thèse n'aurait pas pu avoir lieu sans mes directeurs de thèse : Thomas Corpetti et Sébastien Lefèvre. Plus j'ai avancé dans cette thèse, plus je me suis rendu compte de la chance que j'ai eu de vous avoir à mes côtés (même à distance) pour m'accompagner et me soutenir dans cette aventure. Merci pour votre patience et votre complémentarité, vous formez une belle équipe ! J'espère avoir la chance de travailler à nouveau avec vous ! En tout cas, nos échanges vont me manquer.

Je souhaite ensuite remercier Christian Heipke, Clément Mallet et Bertrand Le Saux pour avoir rapporté ma thèse, ainsi que Dino Ienco, Charlotte Pelletier et Florent Poux pour avoir accepté de participer à mon jury de thèse. Merci beaucoup pour vos retours précieux sur mon travail. J'en profite aussi pour remercier Dino, pour son accueil chaleureux à la maison de la Télédétection à Montpellier.

Je tiens également à remercier Loïc Landrieu et François Rousseau pour leurs conseils avisés durant les comités de suivi de ma thèse.

Mes remerciements au CNES et à Magellium pour avoir co-financé cette thèse. En particulier, je remercie Pierre Lassalle, Thomas Ristorcelli et Chloé Thenoz pour avoir suivi la thèse et m'avoir laissé l'entière liberté de mener la recherche que je voulais. Mes remerciements aussi à Joël, Gwenn et l'équipe EO de m'avoir accueilli dans leur équipe et dans les locaux Toulousains.

Tous ces travaux n'auraient pas non plus pu être réalisés sans l'accès aux clusters de calculs de Jean-Zay, IDRIS (allocation 2022-AD011011754R2 faite par GENCI). Le cluster de l'IRISA a aussi bien été utilisé, alors je remercie chaleureusement Mario pour son aide concernant tout ce qui a eu un rapport avec l'installation informatique. Merci d'avoir toujours été disponible pour aider.

Lors de ces trois années, j'ai aussi eu l'occasion de mener différentes collaborations avec d'autres laboratoires qui auront été très enrichissantes pour moi. Marion Jaud, Pauline Letortu, Dimitri Lague, merci de m'avoir permis de tester mes méthodes sur vos données et ainsi de permettre une nouvelle dimension applicative à ma thèse. J'ai beaucoup apprécié ces problématiques et les échanges que nous avons pu avoir.

Durant trois mois, j'ai aussi été accueillie dans le laboratoire AI4EO à Munich. *A great thanks to Xiaoxiang Zhu, Sudipan Saha and Julia Kollofrath for their welcome and scientific exchanges.* Merci aussi à Henri et Ariane d'avoir été là pendant mon séjour Munichois !! J'espère vous revoir prochainement.

Malgré la distance (et aussi grâce à celle-ci, finalement), j'ai pu faire de belles rencontres durant cette thèse. Voici un pêlemêle de remerciements :

- L'équipe Obelix et plus globalement l'IRISA Vannetais. Merci de votre accueil ! En particulier, merci Manal, Raounak, Elyes, Lucie, Claire, Monica (et Julian de m'avoir accueilli chez vous), Jean-Christophe, Corentin (belle rencontre de dernière minute), Badie, Behzad, Florent, Audrey, Guglielmo, Anne Le T., Thibault G., Minh-Tan, Nico, Laetitia, Chloé, Charlotte (oui, tu apparais une deuxième fois) ... Un grand merci aussi à Jamal pour ton aide et ton soutien ! C'était un plaisir de vous rencontrer.
- Merci Loïc M. et Paul G. pour vos contributions lors de vos stages.
- Merci aussi à Zoé B. pour cette collaboration. C'était un plaisir de travailler avec toi.
- Des rencontres Toulousaines : Valentine B. et Romain T. Merci Valou pour cette coloc'. Tu as été une super rencontre de ces trois années de thèse, j'espère pouvoir continuer à se voir à l'avenir !
- Gaston : une belle rencontre à mi-chemin entre conférence et escalade !
- Les doctorants du MIO : à Nadège, Marine, Emilie, Roxane, et bien entendu aux Béarnais Théo et Julie ! Merci de m'avoir accueilli dans votre groupe et même dans votre labo lors de mes nombreux passages à Marseille. Merci pour toutes ces sorties grimpe, rando, baignade, sud-ouest et même poterie !
- Les doctorants Montpelliérains et Sétois : Sofía, Gus, Emma et Amaël. De belles rencontres de fin de thèse !

Il y a aussi des personnes que je connaissais déjà avant le début de ma thèse, mais qui ont beaucoup compté pour moi. En particulier, je souhaiterais remercier :

- Roza, à notre amitié et à nos souvenirs brestois ! Un grand merci à toi et Chrystos pour votre accueil toujours chaleureux. σε ευχαριστώ και πάλι !
- Ma famille de l'ENSG : Maylis, Hugo, Marion H., Marie S. et Mathilde Pipou, mais aussi Juliette O. (merci pour ta chambre Toulousaine), Augustin et Damien. Une mention spéciale à Mathilde Pipou, merci pour tous nos échanges durant ces deux dernières années. J'étais heureuse de te retrouver. Bon courage pour les derniers mois de thèse qu'il te reste !
- Une deuxième famille de l'ENSG plus cannetonesque ! Un grand merci à Juliette D., Manna, Axel, Valentine M., Sandrine, Brubru ! Merci pour votre soutien et votre bienveillance ! Juju, merci pour ta patience ! Brubru pour ton accueil à Montpellier à notre arrivée, Manna en particulier pour le week-end soutien pré-rendu de manuscrit. Axel pour ton accueil et les soirées Obelix !
- Merci Moutou pour les sorties grimpes !
- Merci Valentina pour tout ton soutien depuis la prépa ! Notre amitié compte énormément à mes yeux et m'a beaucoup aidé durant ces trois dernières années. J'ai beaucoup aimé tous les échanges que nous avons pu avoir, merci pour tout ce que tu m'apportes !
- Cassandre, Pénélope, Zoé, Zaz, Marion I. ... Amies de longue date que j'ai malheureusement moins l'occasion de voir, mais je pense à vous souvent et un grand merci à tout ce que vous avez pu m'apporter !

Je souhaiterais aussi remercier certains professeurs qui m'auront permis par leur confiance et leurs encouragements d'en arriver jusque-là : M. Gardeil, M. Poirot, Mme Tessanne, Mme Gibelin, M. Nace et M. Xuereb (enseignants rencontrés durant mon collège et lycée), M. Chakroun (en prépa), Antoine P., Marc P., Serge B., Jacques B., Xavier C. et bien entendu Anna C. et Pierre-Yves H. de l'ENSG ! Merci aussi Catherine P. pour avoir éveillé mon goût pour la recherche et confirmé mon intérêt pour l'observation de la Terre !

J'en viens à remercier ma famille maternelle et paternelle. Malgré la distance, je pense souvent à vous tous ! En particulier pour leur soutien et leurs attentions qui ont été très

importantes pour moi, merci Emma L., Camille, Christiane, Gérard, Emile, Adrien, Alain, Laurent, Stéphanie, Pépé, Mémé... ainsi que Nathalie, Éric, Muriel, Jacques, Bibi, Michel, Amandine, Laurine (Zoé et Martin), Nico, Damien, Papy et Mamy... Merci à vous deux, papa et maman.

Enfin, je tiens à remercier du fond du cœur Corinne et Jacques P. pour leur accueil durant ces trois années de thèse et pour tous les bons moments passés durant ces confinements multigénérationnels.

Pour finir ces remerciements et parce que je ne serais pas non plus arrivée jusqu'ici sans ton soutien, merci Witold. Merci pour le recul que tu me permets de prendre par rapport à mes travaux, mais aussi pour toutes ces réflexions que tu m'as partagées sur la recherche, et également sur le monde qui nous entoure. Merci de partager ces moments avec moi, j'ai hâte de continuer cette aventure qu'est la vie avec toi !

Finalement, merci à tous d'avoir été là et de m'avoir soutenu à différents moments.

Merci !

A vous tous.

Table of Contents

Résumé

Contexte

L'analyse des changements pour le suivi de l'environnement

L'époque contemporaine s'accompagne de changements toujours plus rapides et fréquents de nos paysages, qu'ils soient causés par des processus géomorphologiques ou par des activités humaines. Que ce soit par rapport à l'évolution du littoral, aux changements dans les milieux montagneux ou au développement incessant des zones urbaines, l'ensemble de notre monde est en constante transformation. L'érosion des falaises (Letortu et al., 2015) ou des dunes côtières (Enríquez et al., 2019), souvent due à l'élévation du niveau de la mer liée au changement climatique (Allan et al., 2021), sont des exemples de cette modification du trait de côte. Dans les zones montagneuses, la fonte des glaciers (Hock, 2005) ou les glissements de terrain (Malamud et al., 2004) ont un impact considérable sur le paysage comme le montre la Figure 1. En plus des transformations du paysage, ces modifications peuvent mettre en danger la population locale (Pollock et Wartman, 2020), et avoir un impact important sur l'économie. Bien qu'il soit souvent difficile d'attribuer directement à un ou plusieurs processus géomorphologiques sous-jacents les modifications topographiques, il semble important, dans un premier temps, d'être capable d'identifier correctement les zones modifiées grâce à la détection et à la cartographie des changements.

Les milieux urbains sont en constante évolution en raison de la croissance constante permanente de la population mondiale et des activités humaines, voir l'exemple de Manaus au Brésil dans la Figure 1d. Selon les Nations Unies, le pourcentage de la population mondiale vivant dans des milieux urbains a doublé entre 1950 et 2020 et tend toujours à augmenter[1]. Cette évolution importante entraîne des modifications considérables des villes et de leurs alentours. Pour générer des cartes adéquates et actualisées (Rottensteiner, 2008 ; Champion et al., 2010), pour aider dans les plans d'urbanisme (Sandric et al., 2007 ; Feranec et al., 2007) et aussi pour identifier rapidement les dommages en cas de catastrophes naturelles (Sofina et Ehlers, 2016 ; Vetrivel et al., 2018), la détection

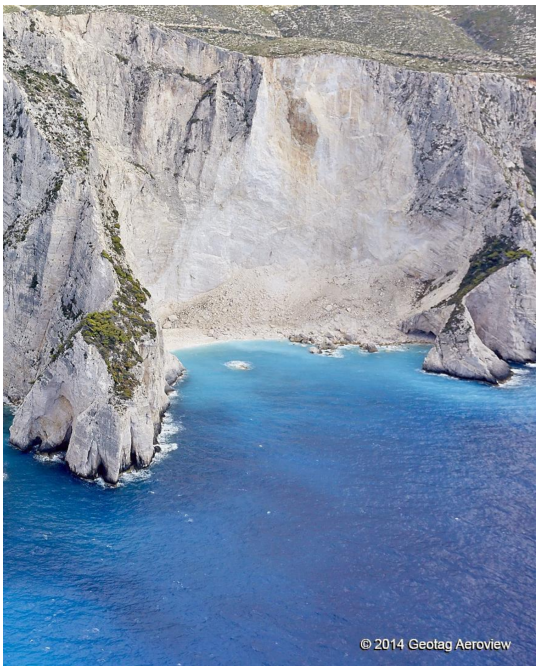
1. Rapport sur les villes du monde 2022 : Envisager l'avenir des villes (`https://unhabitat.org/wcr/`).

de changements et leur classification est une question cruciale également en ville.

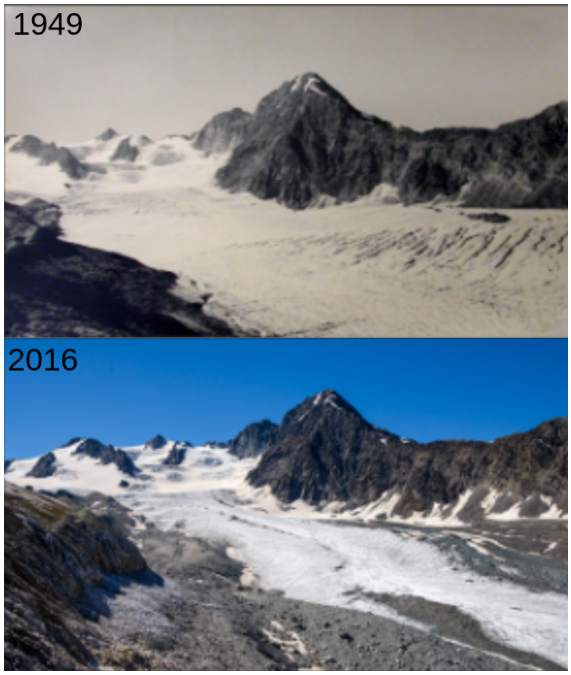
L'observation de tels milieux se fait généralement par imagerie bidimensionnelle, mais il parait également judicieux d'utiliser des données tridimensionnelles (3D) pour représenter notre monde. En milieu urbain, la plupart des objets (bâtiments, végétation, etc.) sont en effet principalement caractérisés par leur axe vertical. En ce qui concerne les changements topographiques, la dimension verticale est également fondamentale. Dans tous ces domaines, il est essentiel d'utiliser des données adéquates pour identifier et quantifier les changements dans des reliefs complexes. Par conséquent, nous préconisons l'utilisation de données tridimensionnelles (3D) telles que les nuages de points pour l'observation de la Terre. Alors que la plupart des études existantes concernant la détection de changement se concentrent uniquement sur des images bidimensionnelles (2D) (Shi et al., 2020a), nous proposons dans cette thèse de nous concentrer sur des données 3D, qui sont mieux adaptées pour refléter la géométrie du monde réel et évitent les problèmes des images 2D tels que la différence des angles de vue entre des acquisitions distinctes, la variabilité spectrale des objets dans le temps, la perspective ou encore les effets de distorsion (Qin et al., 2016). De plus, il a été remarqué que les informations radiométriques ne sont pas suffisantes pour une détection précise des multiples changements urbains (Waser et al., 2007 ; Guerin et al., 2014 ; Erdogan et Yilmaz, 2019).

La donnée 3D pour observer l'environnemnt

Grâce à l'acquisition photogrammétrique ou aux capteurs LiDAR, les nuages de points 3D sont de plus en plus populaires. Comme l'illustre la Figure 2a, le capteur LiDAR (Light Detection And Ranging) repose sur un calcul de distance en visant la scène avec un laser et en mesurant le temps de retour de la lumière réfléchie vers le récepteur. Ce capteur a l'avantage de pouvoir pénétrer la végétation et de détecter le sol sous le couvert végétal, ce qui est particulièrement intéressant en géosciences. Les levés LiDAR peuvent être effectués à partir de plates-formes fixes ou mobiles (terrestres ou aériennes) en fonction de l'application : pour la cartographie à grande échelle telle que la cartographie urbaine ou dans les vastes zones montagneuses, les levés LiDAR aéroportés sont souvent utilisés. Le levé terrestre est préféré pour analyser des objets plus spécifiques et locaux. La densité de points résultante peut varier de moins d'un point (avec les levés aéroportés) à des milliers (avec les levés terrestres) par mètre carré. En plus des coordonnées des points, les capteurs LiDAR fournissent des informations supplémentaires, telles que l'intensité du signal rétrodiffusé et le nombre d'échos.



(**a**) Erosions des falaises de Zakyntos (Plage du naufrage, Grèce)



(**b**) Fonte du glacier Gébroulaz (Vanoise, France)

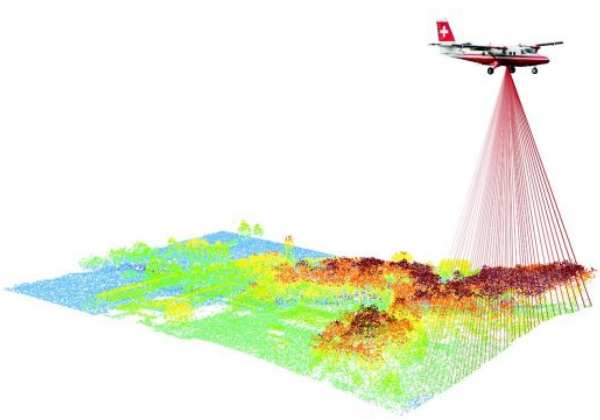
(**c**) Glissement de terrain survenant à la suite d'un séisme (Las Colinas, Santa Tecla, El Salvador)

(**d**) Expansion de la ville de Manaus (1984-2020) (Brésil)

FIGURE 1 : **Exemples de modifications géomorphologiques ou anthropiques de nos paysages** : (**a**) Source : 2014 Geotag Aeroview ; (**b**) Source : Office national des Forêts et Parc national de la Vanoise ; (**c**) Source : GARCÍA-RODRÍGUEZ et MALPICA (2010) ; (**d**) Source : Google Earth Timelapse (Google, Landsat, Copernicus).

La photogrammétrie est un autre moyen habituel d'obtenir des nuages de points 3D. Celle-ci consiste à assembler plusieurs images 2D avec des points de vue légèrement différents pour obtenir une reconstitution 3D de la scène, comme l'illustre la Figure 2b. Elle s'appuie sur la vision stéréoscopique, qui rassemble des images du même objet provenant de deux ou plusieurs sources avec des angles de vue différents, pour obtenir une position 3D des objets observés. Ces données sont généralement composées de bandes de couleurs (rouge, vert, bleu) dont les valeurs sont complémentaires aux coordonnées des points. Notons que par rapport aux données LiDAR, même si la précision spatiale est généralement moindre avec les levés photogrammétriques, le processus d'acquisition (reconstruction 3D à partir des images) est moins coûteux. L'acquisition photogrammétrique peut être terrestre, aérienne, ou même satellitaire. En particulier, les missions satellitaires sont intéressantes, car, une fois lancées, l'acquisition de données régulières sur une grande surface est peu coûteuse par rapport à un levé aérien, même si les nuages de points obtenus sont moins denses et moins précis. Parmi les principales missions satellitaires d'acquisition de données 3D, la constellation Pléiades fournit, depuis 2011, des images panchromatiques (à 70 cm de résolution) qui peuvent être utilisées pour la reconstruction photogrammétrique. Comme cette mission va bientôt prendre fin, elle devrait normalement être remplacée par la constellation Pléiades Néo, fournissant des images à une résolution de 30 cm permettant d'améliorer considérablement l'acquisition de données 3D depuis l'espace. Le contexte de cette thèse repose également sur le futur lancement de la mission CO3D (Constellation Optique 3D) qui est spécialement prévue pour produire une acquisition 3D des surfaces terrestres du monde entier (Lebègue et al., 2020). Cette mission vise à obtenir un modèle numérique de surface (MNS), c'est-à-dire une modélisation de l'élévation du terrain, à une résolution d'1 m. Enfin, bien que beaucoup moins courante, la tomographie à partir de radars à synthèse d'ouverture (SAR) peut également être utilisée pour générer des données 3D en couplant plusieurs images SAR avec des angles de vue légèrement différents (Zhu et Bamler, 2010 ; Aghababaei et al., 2020).

Quel que soit le mode d'acquisition, les nuages de points 3D partagent des caractéristiques communes radicalement différentes des images 2D habituelles. Un nuage de points 3D est un ensemble non ordonné et épars de points représentés par leurs coordonnées 3D dans un système de référence (système de coordonnées cartésiennes), un exemple de telles données est fourni dans la Figure 3. Contrairement aux images 2D organisées à travers une grille régulière de pixels (rasters 2D), les nuages de points 3D sont désordonnés et distribués de manière irrégulière, ce qui rend l'extraction d'informations dans ces données



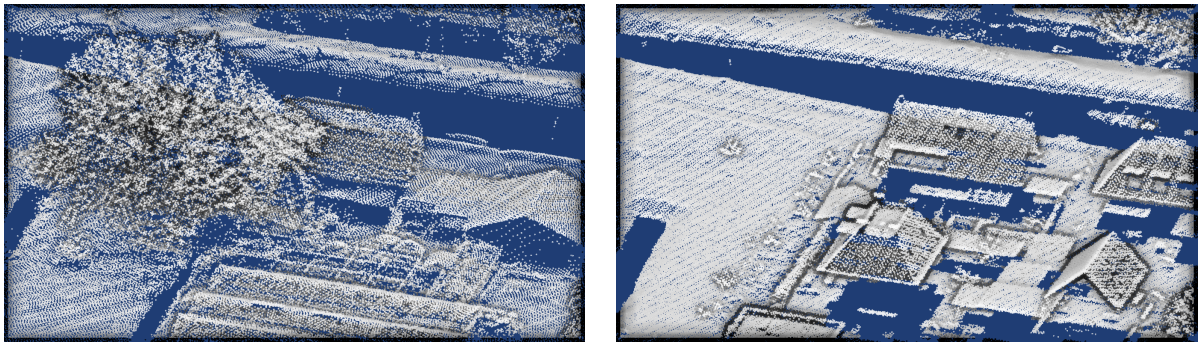
(**b**) LiDAR aérien
Source : `swisstopo.admin.ch`

(**a**) Photogrammetrie aérienne
Source : J. Vallet

FIGURE 2 : **Schéma d'une acquisition aérienne LiDAR ou photogrammétrique.**

délicate. Les comparaisons entre les dates le sont encore plus. En effet, il n'y a pas de correspondance directe point à point et, comme on peut le voir sur la Figure 3, l'emplacement et la distribution des points peuvent varier de manière significative même dans des zones inchangées, ce qui rend la tâche de détection de changement plus difficile. Une autre difficulté des nuages de points 3D est qu'ils contiennent des millions de points pour un seul levé, ce qui complique le traitement des zones larges échelles étudiées pour l'observation de la Terre. Les nuages de points sont épars, ce qui implique qu'aucune autre information que l'emplacement précis des points n'est fournie. En plus de cette caractéristique, les nuages de points 3D contiennent beaucoup d'occlusions, phénomène rencontré lorsqu'une partie de la scène réelle n'est pas représentée dans le nuage de points. Des exemples de telles occlusions sont visibles sur le nuage de points à la date 2 de la Figure 3, où les façades des bâtiments ne sont pas acquises par le capteur LiDAR. Lorsque les occlusions varient d'une date à l'autre, elles rendent délicate la distinction entre le changement réel et les modifications des données dues à la configuration de l'acquisition.

Compte tenu de la complexité de la manipulation de ces données brutes, la rastérisation des nuages de points sur des grilles régulières de pixels de hauteur, appelée modèle numérique de surface (MNS), est une solution pour faciliter l'application des approches traditionnelles de traitement d'images et la comparaison directe des élévations. Cependant, le processus de rastérisation implique une perte d'informations potentiellement intéressantes, par exemple sur les façades des bâtiments. De plus, selon la taille de la grille choisie, les points sont agrégés en une seule valeur, ce qui entraîne une perte d'information



Nuage de points à la date 1 Nuage de points à la date 2

FIGURE 3 : **Exemple de nuage de points LiDAR à deux dates.** Les points 3D sont représentés par les points blancs.

importante si les pas de discrétisation sont trop grands. D'autre part, une taille de grille trop fine conduit à de nombreux pixels vides, généralement remplis par une interpolation provoquant des données approximatives. Dans un esprit similaire à celui des MNS, une grande majorité de méthodes en géosciences s'appuie sur des modèles numériques de terrain (MNT), c'est-à-dire sur une rasterisation en 2,5D de nuages de points au niveau du sol (sans le couvert végétal ou le bâti), pour mettre en évidence les modifications du sol. En dehors des MNS ou MNT 2.5D, la rastérisation 3D en une grille de voxels 3D est également une possibilité pour faciliter le traitement. Cependant, comme pour la rastérisation 2D, un grand pas d'échantillonnage implique également une perte d'informations, et une taille de grille trop fine devient rapidement coûteuse en termes de calcul en raison de la caractéristique éparse des environnements 3D.

Apprentissage profond pour le traitement de données d'observation de la Terre

Grâce à l'augmentation des capacités de calcul, l'utilisation de l'apprentissage profond (ou *deep learning*) s'est généralisée au cours de la dernière décennie. L'apprentissage profond est un sous-ensemble d'algorithmes d'apprentissage automatique (ou *machine learning*) qui utilise des approches mathématiques et statistiques pour donner aux ordinateurs la capacité d'apprendre à partir de données afin de résoudre un problème posé, également appelé tâche. L'apprentissage automatique et l'apprentissage profond font tous deux partie de ce que l'on appelle l'intelligence artificielle (IA). Alors que l'apprentissage automatique repose sur l'élaboration manuelle de caractéristiques représentant la donnée, l'apprentissage profond apprend à extraire ses propres caractéristiques liées à la tâche à résoudre.

Depuis la première proposition de perceptrons (i.e., neurones artificiels) multi-couches jusqu'aux réseaux profonds actuels, de nombreuses améliorations ont été apportées. Ils peuvent désormais s'attaquer à de multiples tâches liées à la compréhension des données, grâce à un apprentissage entièrement supervisé reposant sur des échantillons annotés, ou même à un apprentissage non supervisé (c'est-à-dire sans l'utilisation d'échantillons étiquetés). La télédétection et l'observation de la Terre ont bénéficié de ces améliorations méthodologiques (Zhu et al., 2017). En particulier, l'apprentissage profond est utilisé avec succès pour la classification des sols (Kussul et al., 2017), la détection d'objets (Ding et al., 2021), l'analyse de séries temporelles d'images satellites (Pelletier et al., 2019), la super-résolution (Wang et al., 2020c), et enfin dans le contexte qui nous intéresse, la détection des changements dans les images (Daudt et al., 2018).

Compte tenu des grandes différences entre les images et les nuages de points, et de la complexité de ces derniers, l'utilisation de méthodes basées sur l'apprentissage profond pour la compréhension des nuages de points a commencé plus tard et de nombreux travaux restent à faire. Cependant, depuis l'apparition de PointNet (Qi et al., 2017a), l'apprentissage profond a prouvé sa capacité à traiter des données aussi compliquées.

À notre connaissance, au début de la thèse, les méthodes d'apprentissage profond n'avaient jamais été utilisées pour la détection de changement dans les nuages de points bruts. Cependant, compte tenu de leur capacité d'analyse de données complexes et de leur possibilité de compréhension globale des scènes, il semble évident que l'extraction des changements dans ces données particulières peut bénéficier des avancées méthodologiques de l'apprentissage profond. Par conséquent, cette thèse vise à étudier comment **utiliser les concepts de l'apprentissage profond pour détecter les modifications (c'est-à-dire les changements) dans nos paysages à partir de données 3D**.

Objectifs de la thèse

L'**objectif général** de cette thèse est d'explorer la question de la **détection de changement dans des données tridimensionnelles**, et en particulier dans des **nuages de points** 3D en se basant sur les récentes avancées méthodologiques de l'**apprentissage profond**. En particulier, nous désirons comparer deux nuages de points acquis à deux dates différentes comme dans la Figure 3, et à fournir en outre une identification des zones modifiées via une segmentation au niveau des points du deuxième nuage de points (c'est-à-dire du plus récent). Concernant les nuages de points 3D, des informations binaires

(changement/absence de changement) ou multiples (nature du changement) peuvent être extraites. La catégorisation des changements fait également référence à l'identification de ceux-ci parmi plusieurs classes. Ensuite, comme pour les images en 2D, les méthodes de détection des changements peuvent donner des résultats de classification ou de segmentation. Dans un cadre de classification des changements, les résultats sont obtenus au niveau des nuages de points en entier, c'est-à-dire une étiquette pour une paire de nuages de points. En revanche, les résultats de la segmentation des changements sont donnés à l'échelle du point. Les différents paramètres du problème de détection et de catégorisation des changements à partir de nuages de points 3D sont résumés dans la Figure 4. Dans cette thèse, nous visons principalement à **aborder la tâche de segmentation de changements multiples** (Figure 4d) qui parvient à une précision plus fine des résultats. Notons que la résolution de la tâche de segmentation de changements multiples résout également les autres tâches présentées dans la Figure 4.

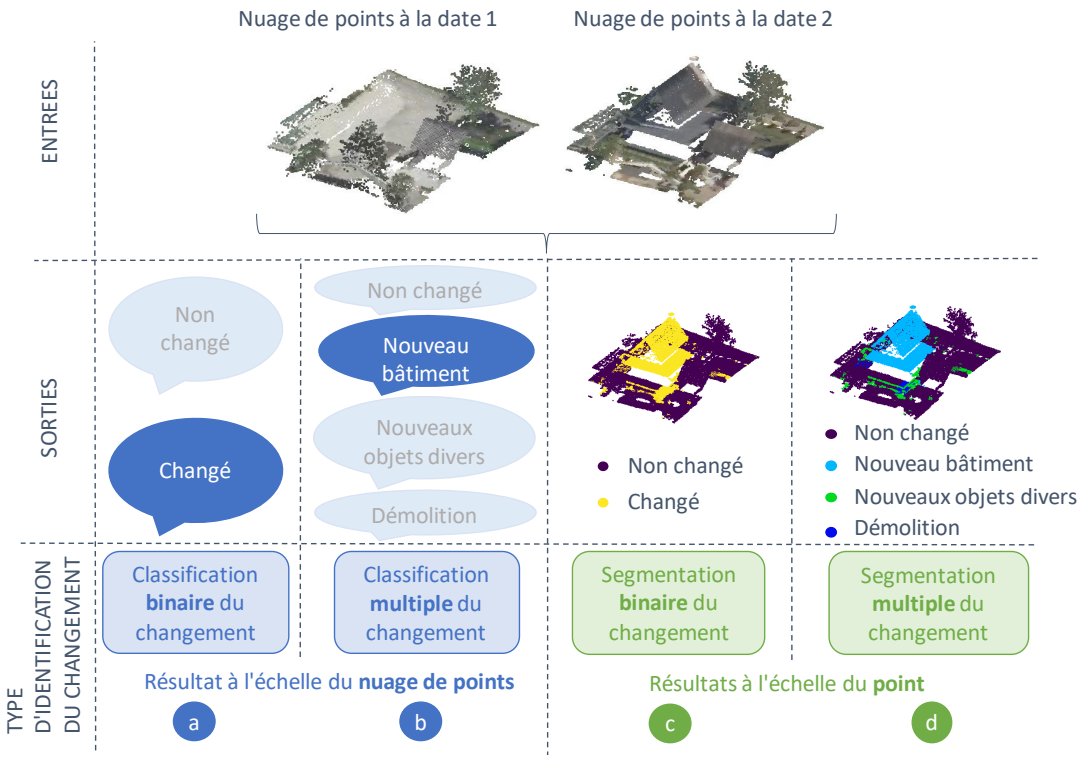


FIGURE 4 : **Différents types de résultats de détection de changement**, à l'échelle du nuage de points (**a** et **b**) ou des points directement (**c** et **d**), et avec des classes binaires (**a** et **c**) ou multiples (**b** et **d**). Cette thèse vise à aborder le problème des changements multiples (**d**). Notons qu'à partir de cette tâche plus complexe, les autres tâches (**a**, **b** et **c**) peuvent également être résolues.

Contributions

Cette thèse est structurée autour de cinq chapitres, résumés ci-dessous.

Chapitre 1 : La donnée pour la détection de changement en 3D

Ce chapitre présente une vue d'ensemble des jeux de données 3D multi-temporels existants ainsi qu'une analyse de leurs propriétés. Les jeux de données publics existants ne sont pas directement adaptés à notre objectif car ils ne disposent pas d'annotation en lien avec le changement. Par conséquent, nous avons proposé de **créer plusieurs jeux de données** pour aborder la tâche de segmentation des changements dans des nuages de points 3D.

À cette fin, un **processus semi-automatique** est proposé afin d'**adapter des jeux**

de données 3D multi-temporels existants avec une annotation sémantique non-relative au changement. Grâce à cette chaîne de traitement, un jeu de données (AHN-CD) dérivé du jeu de données LiDAR aéroporté AHN (Actueel Hoogtebestand Nederland) est créé. Bien que le processus semi-automatique permette de convertir rapidement et facilement certains jeux de données 3D multi-temporels avec annotation sémantique à des fins de détection des changements, l'annotation issue de ce processus contient quelques erreurs de classification.

Afin d'obtenir un jeu de données avec une vérité terrain en laquelle il est possible d'avoir pleinement confiance, un **simulateur original de modèles urbains multi-temporels** est proposé, où les acquisitions 3D sont simulées en imitant un levé LiDAR aéroporté. Ce simulateur permet d'acquérir des paires de nuages de points annotés afin d'entraîner des méthodes supervisées. Le simulateur permet de personnaliser l'acquisition des nuages de points selon différentes configurations. Ainsi, cinq sous-jeux de données différents avec des configurations d'acquisition variées sont créés afin d'illustrer la variabilité en termes de qualité et de type de capteur à laquelleauxquels nous sommes confrontés lors d'une acquisition réelle de nuages de points. Ces sous-jeux de données forment le jeu de données Urb3DCD-V1. Une deuxième version du simulateur est développée pour renforcer le réalisme des nuages de points en ajoutant de la végétation et des objets mobiles aux scènes simulées. Une nouvelle simulation, Urb3DCD-V2, a été générée pour proposer deux sous-jeux de données obtenus selon différentes conditions d'acquisition. Enfin, pour imiter le jeu de données existant de Change3D (Ku et al., 2021), une dernière version des jeux de données simulés, nommée Urb3DCD-Cls, fournit une paire de nuages de points bi-temporelle centrés sur un point d'intérêt avec une annotation de changement au niveau de la scène.

Les jeux de données conçus sont accessibles au public[2] afin de soutenir la recherche reproductible et d'encourager d'autres travaux sur la détection des changements en 3D, notamment avec des méthodes d'apprentissage profond.

Chapitre 2 : État de l'art de la détection de changement 3D en milieu urbain

Ce chapitre aborde tout d'abord une **revue des méthodes existantes**. Ensuite, une c**omparaison expérimentale** de différentes méthodes de détection et de catégorisation

2. Les jeux de données sont disponibles au lien suivant : `https://ieee-dataport.org/open-access/urb3dcd-urban-point-clouds-simulated-dataset-3d-change-detection`.

des changements dans un environnement urbain est proposée.

Six méthodes différentes utilisant soit des MNS, soit directement les nuages de points 3D sont évaluées. Plus précisément, des méthodes traditionnelles basées sur des distances, une technique d'apprentissage automatique avec un algorithme de forêt d'arbres décisionnels (*random forest*) basé sur des caractéristiques créées manuellement, et des architectures d'apprentissage profond sont comparées. Ces méthodes fournissent des résultats à des échelles différentes, du pixel 2D au point 3D. Pour les méthodes supervisées, la capacité de l'apprentissage par transfert et l'influence de la taille de l'ensemble d'apprentissage sont également étudiées.

Les résultats de cette évaluation montrent que le jeu de données simulé (Urb3DCD-V1) en zones urbaines denses est un défi pour toutes les méthodes existantes. Les problèmes restants concernent la gestion des nuages de points à faibles densités et la compréhension globale de la scène en cas d'occlusions dans les nuages de points. Pour les résultats à l'échelle du point 3D, l'apprentissage automatique traditionnel est meilleur que les méthodes basées sur des distances. Les méthodes d'apprentissage profond existantes ne fournissent que des résultats à l'échelle d'une imagette extraite des MNS, ce qui est beaucoup moins précis en termes de résultats. Alors que les méthodes d'apprentissage profond semblent plus adaptées à l'apprentissage par transfert que les méthodes d'apprentissage automatique, ces expériences mettent en évidence la nécessité de disposer de méthodes d'apprentissage profond qui se basent directement sur les nuages de points 3D bruts.

Chapitre 3 : Détection de changement supervisée

Ce chapitre présente, tout d'abord, une architecture profonde originale, appelée *Siamese KPConv*, dédiée à la **détection et à la catégorisation des changements dans les nuages de points**. Cette méthode est basée notamment sur une architecture Siamoise et des convolutions à points noyau (KPConv) pour élaborer, à notre connaissance, le premier réseau profond capable de s'adapter à des paires de nuages de points 3D bruts et d'effectuer une tâche de segmentation des changements. Différentes expériences sont menées dans un environnement urbain en utilisant des jeux de données réels (AHN-CD) et synthétiques (Urb3DCD-V2). Pour chaque jeu de données, notre technique dépasse l'état de l'art avec une marge significative (environ 30% de l'indice de Jaccard (*Intersection Over Union*) moyen sur les classes de changement).

Ensuite, la plus-value relative à l'ajout en entrée du réseau de **caractéristiques conçues manuellement** est évaluée. Cela améliore considérablement les résultats de la

segmentation des changements (environ 4,5% de l'indice de Jaccard moyen sur les classes de changement). Plus précisément, l'ajout d'une caractéristique liée au changement joue un grand rôle dans l'amélioration des résultats. Suite à cette étude, trois autres architectures sont conçues pour la segmentation des changements (*OneConvFusion*, *Triplet KPConv* et *Encoder Fusion SiamKPConv*) afin de mettre l'accent sur les caractéristiques profondes liées au changement. Toutes ces trois architectures obtiennent de meilleurs résultats que le *Siamese KPConv* (de 1,5% à 5% de l'indice de Jaccard moyen sur les classes de changement). Le réseau *Encoder Fusion SiamKPConv*, avec un encodeur spécifique qui fusionne les caractéristiques de changement et mono-dates, permet de s'affranchir de l'ajout de caractéristiques manuelles. Cela souligne l'**importance d'appliquer la convolution également sur les différences de caractéristiques**.

Chapitre 4 : Détection de changement non-supervisée

Ce chapitre se concentre sur la manière de **réduire le nombre de données annotées** nécessaires aux réseaux profond. Pour ce faire, nous avons d'abord proposé d'évaluer les performances d'**apprentissage par transfert** du réseau profond *Siamese KPConv* par rapport à d'autres réseaux profonds travaillant sur des MNS ou à l'algorithme de forêt d'arbres décisionnels (*random forest*). Ensuite, ce chapitre évalue le bénéfice du **pré-entraînement** du réseau sur un jeu de données simulées pour diminuer la taille de l'ensemble d'entraînement nécessaire sur les données réelles. Grâce au pré-entraînement, une réduction drastique du nombre de cylindres du domaine cible est possible pour atteindre le score maximal. Cela réduit considérablement la charge de l'annotation manuelle.

Dans un deuxième temps, deux **méthodes entièrement non supervisées** sont proposées pour aborder la détection de changement binaire. Elles sont basées sur de l'auto-supervision et de l'apprentissage par transfert profond pour caractériser efficacement la donnée cible dans l'espace latent. Ensuite, la méthode d'analyse de vecteurs de changement profond (DCVA : *deep change vector analysis*) a été utilisée pour comparer les points dans l'espace latent et segmenter la zone en parties changées et inchangées. Bien que sur le jeu de données réel (AHN-CD), la méthode basée sur l'apprentissage auto-supervisé ait permis d'obtenir une prédiction plus précise des changements que les méthodes traditionnelles basées sur la distance, ces méthodes non supervisées manquent toujours de précision dans les zones urbaines denses et ne fournissent en outre que des informations binaires sur les changements.

Ainsi, dans une dernière partie, ce chapitre propose une méthode faiblement supervisée

basée sur le principe du clustering profond, et en particulier de *DeepCluster*, pour aborder la segmentation des changements multi-classes dans les nuages de points bruts. Les expériences montrent l'importance du choix d'une architecture appropriée pour extraire des caractéristiques liées aux changements. De même, il est préconisé de guider le réseau en utilisant en entrée, en plus des coordonnées de points 3D, des caractéristiques créées manuellement. Grâce à cette configuration recommandée (architecture appropriée et ajout de caractéristiques manuelles en entrée), la méthode que nous proposons, DC3DCD, permet d'obtenir de meilleurs résultats qu'un algorithme d'apprentissage automatique traditionnel entièrement supervisé reposant sur des caractéristiques pré-calculées, et d'atteindre les scores d'un réseau profond entièrement supervisé travaillant sur les rastérisations 2,5D des nuages de points en MNS. Nous avons également proposé d'améliorer DC3DCD en introduisant une fonction de perte contrastive pouvant conduire à des résultats comparables à ceux d'un réseau profond entièrement supervisé dans un cas idéal où certaines informations basiques sur les changements sont disponibles, ce qui n'est pas encore le cas. Cela ouvre néanmoins à des perspectives prometteuses.

Chapitre 5 : Applications aux géosciences

En plus des expériences en milieu urbain effectuées tout au long des chapitres précédents, dans ce chapitre, nous expérimentons nos architectures supervisées (*Siamese KPConv* et *Encoder Fusion SiamKPConv*) dans deux applications liées aux géosciences, à savoir la détection de l'érosion dans des falaises, et l'identification des sources et des dépôts de glissements de terrain se manifestant à la suite d'un séisme. Au travers de ces deux expériences, sur les acquisitions 3D multi-capteurs des falaises du Petit Ailly (Varengeville-sur-Mer, France), et sur les levés LiDAR aériens de la région montagneuse de Kaikōura en Nouvelle-Zélande, nous mettons en évidence que l'architecture *Encoder Fusion SiamKPConv* n'est pas spécifique à l'environnement urbain pour autant que le jeu de données soit suffisamment représentatif des classes d'intérêt. Une fois de plus, ces évaluations insistent sur la supériorité du réseau *Encoder Fusion SiamKPConv* par rapport au modèle *Siamese KPConv*, soulignant les conclusions précédentes relatives à la nécessité d'encoder les changements via des convolutions sur la différence de caractéristiques mono-dates.

Publications

Cette thèse a donné lieu à la publication de divers articles et communications dans des revues et conférences à comité de lecture, comme détaillé ci-dessous :

Journaux

J1 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Change Detection in Urban Point Clouds : An Experimental Comparison with Simulated 3D Datasets". *Remote Sensing* 13 (2021) : 2629. Link to the open-datasets : `https://ieee-dataport.org/open-access/urb3dcd-urban-point-clouds-simulated-dataset-3d-change-detection`

J2 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Siamese KPConv : 3D multiple change detection from raw point clouds using deep learning". *ISPRS Journal of Photogrammetry and Remote Sensing* 197 (2023) : 274-291. Link to the open-source codes : `https://github.com/IdeGelis/torch-points3d-SiameseKPConv`

J3 Iris de Gélis, Sudipan Saha, Muhammad Shahzad, Thomas Corpetti, Sébastien Lefèvre and Xiao Xiang Zhu. "Deep Unsupervised Learning for 3D ALS Point Clouds Change Detection". *arXiv preprint arXiv : 2305.03529* Link to the open-source codes : `https://github.com/IdeGelis/torch-points3d-SSL-DCVA`

J4 Iris de Gélis, Thomas Corpetti, and Sébastien Lefèvre. "Change detection needs change information : improving deep 3D point cloud change detection". *arXiv preprint arXiv : 2304.12639*. Link to the open-source codes : `https://github.com/IdeGelis/torch-points3d-SiamKPConvVariants`

J5 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "DC3DCD : unsupervised learning for multi-class 3D point cloud change detection". *arXiv preprint arXiv : 2305.05421*. Link to the open-source codes : `https://github.com/IdeGelis/torch-points3d-DC3DCD`

Conférences internationales avec actes

C1 Iris de Gélis, Sébastien Lefèvre, Thomas Corpetti, Thomas Ristorcelli, Chloé Thénoz and Pierre Lassalle. "Benchmarking change detection in urban 3D point clouds".

IEEE International Geoscience and Remote Sensing Symposium – IGARSS (2021) : 3352-3355.

C2 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "3D urban change detection with point cloud siamese networks". *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 43 (2021) : 879-886.

C3 Iris de Gélis, Zoé Bessin, Pauline Letortu, Marion Jaud, Christophe Delacourt, Stéphane Costa, Olivier Maquaire, Robert Davidson, Thomas Corpetti, and Sébastien Lefèvre. "Cliff Change detection using Siamese KPConv deep network on 3D point clouds". *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* 3 (2022) : 649-656.

C4 Iris de Gélis, Thomas Bernard, Dimitri Lague, Thomas Corpetti and Sébastien Lefèvre. "Landslide detection in 3D point clouds with deep Siamese convolutional network". *IEEE International Geoscience and Remote Sensing Symposium – IGARSS* (2023)

Conférence nationale avec actes

C5 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Détection de changements urbains 3D par un réseau Siamois sur nuage de points". *Journées francophones des jeunes chercheurs en vision par ordinateur – ORASIS* (2021).

Conférences internationales sans acte

O1 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Urb3DCD V2 : An open dataset for urban 3D point clouds change detection". *Joint Urban Remote Sensing Event (JURSE)* (2022)

O2 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Can simulated data help when dealing with EO data ? An illustration with 3D point cloud change detection". *ESA Living Planet Symposium* (2022)

Conférences nationales sans acte

O3 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Change detection in 3D point clouds based on deep learning". *Journées Jeunes Chercheurs – Centre National*

d'études Spatiales (2021) – Best poster award

O4 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "3D point clouds change detection : From supervised to unsupervised deep learning methods". *Webinaire « Autour de la 3D »MAGIS/IG-RV* (2022)

O5 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Approche DeepCluster faiblement supervisée pour la détection de changement dans des nuages de points 3D". *GdR ISIS « Approches faiblement supervisées en Télédétection »* (2023)

O6 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Détection de changement en milieu urbain au sein de nuages de points 3D par apprentissage profond". *Télédetection pour l'étude du milieu urbain (TEMU)* (2023)

INTRODUCTION

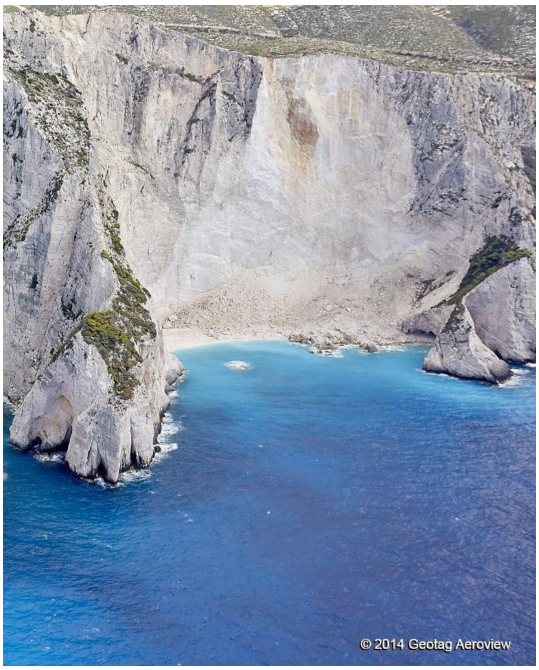
Context

Due to natural topography modifications, anthropogenic activities, and extreme events, contemporary times are accompanied by ever more rapid and frequent changes in our landscapes. Whether it is in relation to evolution in the coastline, to changes in the mountainous environment, or to the incessant development of urban areas, all of our world is finally in constant transformation. Cliff (Letortu et al., 2015) or coastal dune (Enríquez et al., 2019) erosion often due to the sea level rise linked to the climate change (Allan et al., 2021) are examples of such coastal line modification. When it comes to mountainous areas, the glaciers melting (Hock, 2005) or landslides (Malamud et al., 2004) are considerably impacting the landscape as visible in Figure 5. On top of landscape transformations, these modifications may endanger local population (Pollock and Wartman, 2020), and have a great impact on local economy. While it is often hard to directly attribute to one or more underlying geomorphic processes causing the topographic modifications, it seems important to, first, be able to identify correctly the modified areas through change detection and mapping.

Because of the constant growth of the world population and human activities, urban areas are continuously evolving, see the example of Manaus, Brazil in Figure 5d. According to the United Nations, the percentage of earth population living in urban areas has doubled between 1950 and 2020 and still tends to increase[3]. This important evolution is surely leading to considerable modifications in the urban areas and their surroundings. To generate adequate and updated maps (Rottensteiner, 2008; Champion et al., 2010), to help territorial planners in city management (Sandric et al., 2007; Feranec et al., 2007) and also to quickly identify damage in the case of natural disasters (Sofina and Ehlers, 2016; Vetrivel et al., 2018), multi-class change detection is a crucial issue also in urban areas.

Rather than limiting ourselves to a two-dimensional representation, it seems appro-

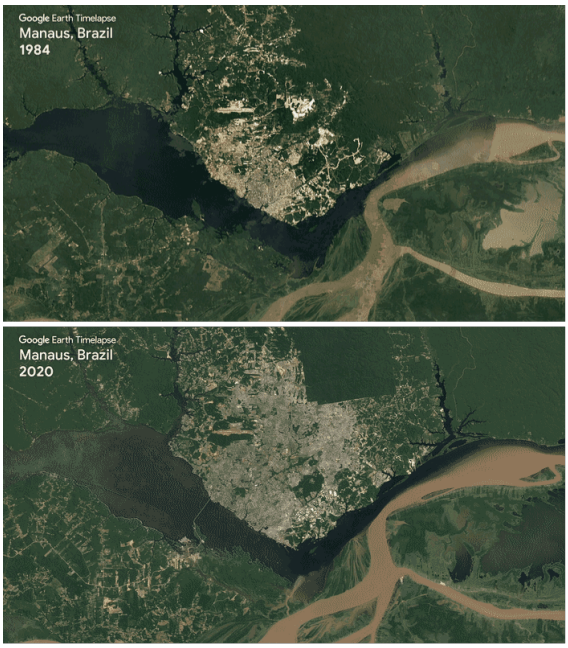
3. United Nations Habitat: World Cities Report 2022: Envisaging the Future of Cities (`https://unhabitat.org/wcr/`).



(**a**) Zakyntos cliffs erosion (Shipwrek beach, Greece)

(**b**) Gébroulaz glacier melting (Vanoise, France)

(**c**) Landslide following an earthquake (Las Colinas, Santa Tecla, El Salvador)



(**d**) Manaus city expansion (1984-2020) (Brazil)

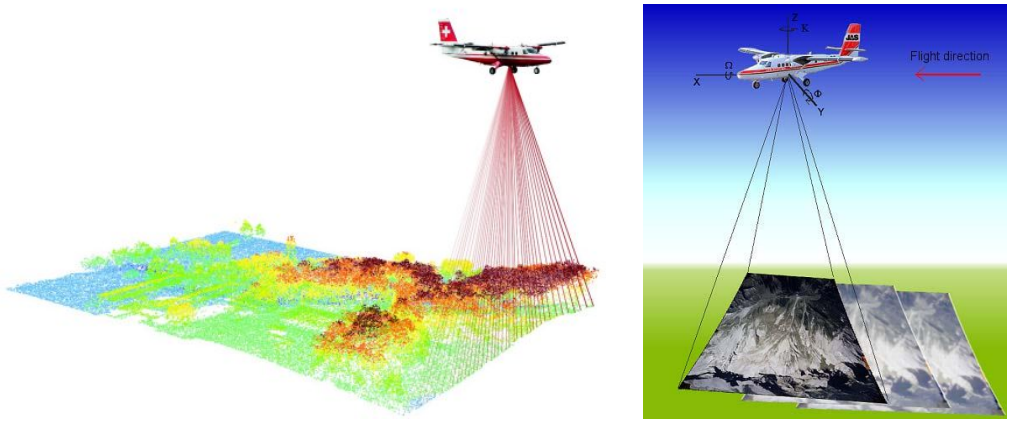
Figure 5: **Examples of geomorphic or anthropogenic modifications of our landscape**: (**a**) Source: 2014 Geotag Aeroview; (**b**) Source: Office national des Forêts and Parc national de la Vanoise; (**c**) Source: García-Rodríguez and Malpica (2010); (**d**) Source: Google Earth Timelapse (Google, Landsat, Copernicus).

priate to use three-dimensional (3D) data to embody our world. In urban environments, most objects (buildings, vegetation, etc.) are mainly characterized by their vertical axis. With respect to topographic changes, it is essential to use adequate data to identify and quantify changes in complex landforms. Therefore, we advocate for the use of 3D data such as 3D Point Clouds (PCs) in Earth observation. While most existing studies concerning change detection only focus on two-dimensional (2D) images (Shi et al., 2020a), we propose to concentrate on 3D data, which is better suited to reflect real world geometry and avoids 2D image problems such as the difference of viewing angles between distinct acquisitions, spectral variability of objects over time, perspective, and distortion effects (Qin et al., 2016). Furthermore, it has been noticed that radiometric information is not sufficient for accurate multiple urban change detection (Waser et al., 2007; Guerin et al., 2014; Erdogan and Yilmaz, 2019).

3D data for Earth observation

Thanks to photogrammetric acquisition or Light Detection And Ranging (LiDAR) sensors, 3D PC data are becoming more popular. As illustrated in Figure 6a, LiDAR sensor relies on distance computation by targeting the scene with a laser and measuring the time for the reflected light to return to the receiver. This sensor has the great advantage of being able to penetrate the vegetation and detect the ground under the vegetation cover, which is particularly interesting in the geosciences. LiDAR surveys can be performed from fixed or mobile platforms (terrestrial or aerial) depending on the application: for large scale mapping, such as for urban monitoring or for wide mountainous area, Aerial Laser Scanning (ALS) is often used. Terrestrial Laser Scanning (TLS) is preferred to analyze more specific and local objects. The resulting point density can vary from less than a point (with ALS) to thousands (with TLS) per square meter. In addition to the coordinates of points, LiDAR sensors provide additional information, such as the intensity of the back-scattered signal and number of echoes.

The other usual way of obtaining 3D PCs is through photogrammetric acquisition. This latter consists in assembling multiple 2D images with slightly different points of views to obtain a 3D reconstitution of the scene, as it is illustrated in Figure 6b. It relies on the stereoscopic vision of human, who gather two images of the same object coming from both eyes to obtain a relief view. Such data are usually composed of color bands (Red Green Blue (RGB)) whose values are complementary to the point coordinates. Let us note that compared to LiDAR data, even if the spatial precision is generally lower with



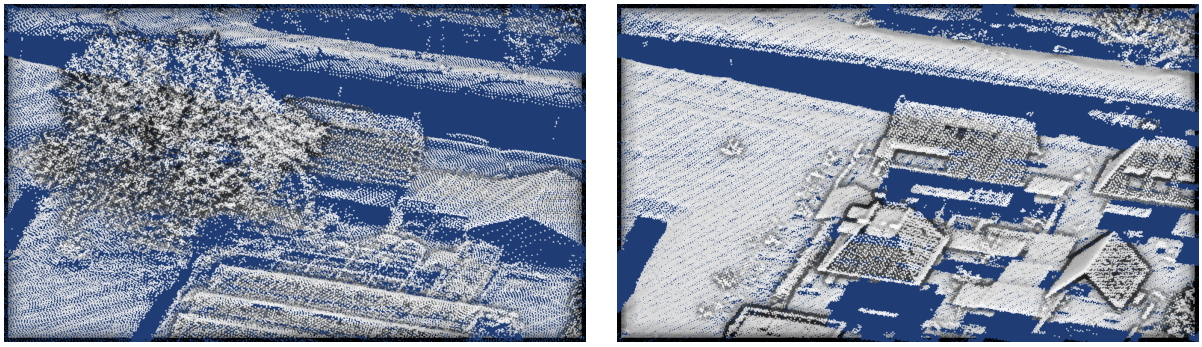
(**a**) Aerial LiDAR
Source: `swisstopo.admin.ch`

(**b**) Aerial photogrammetry
Source: J. Vallet

Figure 6: **Illustration of aerial LiDAR and photogrammetry acquisition.**

photogrammetric images, the acquisition process (3D reconstruction from images) is less costly. Photogrammetric acquisition can be terrestrial, aerial, or even from satellites. In particular, satellite missions are interesting since once launched, the acquisition of regular data over large area is costless compared to an aerial survey, even though obtained PCs are less dense and accurate. Among the main satellite missions to acquire 3D data, the Pléiades constellation provides, since 2011, 70 cm pan-chromatic images that can be used for photogrammetric reconstruction. As it will soon end, it should normally be replaced by the Pléiades Néo constellation providing 30 cm resolution images allowing to considerably improve 3D data surveying from space. The context of this thesis also relies on the future launch of the 3D Optical Constellation (CO3D) mission that is planned to produce 3D acquisitions of worldwide land surfaces (Lebègue et al., 2020). This latter mission aims to obtain a 2.5D modelling of the surface (i.e., Digital Surface Model (DSM)) at 1 m resolution. Finally, even though far less common, Synthetic Aperture Radar (SAR) tomography can also be used to generate 3D data by coupling several SAR images with slightly different viewing angles (Zhu and Bamler, 2010; Aghababaei et al., 2020).

No matter the way of acquisition, 3D PCs share common characteristics drastically different from usual 2D images. A PC is an unordered and sparse set of points represented by their 3D coordinates in a frame of reference (Cartesian coordinate system), an example of such data is given in Figure 7. Unlike 2D images organized through a regular grid of pixels (2D rasters), 3D PCs are disordered and irregularly distributed, which makes the extraction of information from such data tricky, and between-timestamp comparisons are



PC date 1 PC date 2

Figure 7: **Example of LiDAR 3D Point Clouds (PCs) at two timestamps.** 3D points are represented by the white dots.

even trickier. Indeed, there is no direct point-to-point correspondence. And, as visible in Figure 7, point locations and distribution can vary significantly even in unchanged areas, making the change detection task harder. Another difficulty of 3D PCs is that they involve millions of points for a single survey, which complicates the processing of large remote sensing scenes. Lastly, in addition to the sparsity, which implies that no other information is contained in the 3D space aside from the precise points' location, 3D PCs contain a lot of occlusions, i.e., part of the scene existing in the reality but not represented in the PC. Examples of such occlusion are visible on the PC at date 2 of Figure 7, where building facades are not acquired by the LiDAR sensor. When occlusions are varying from one timestamp to another, they are making delicate the distinction between real changes and changes in data due to the configuration of acquisition.

Considering the complexity of handling such raw data, rasterizing PCs onto regular grids of pixels of height is a solution to ease the application of traditional 2D image processing approaches and facilitate the direct elevation comparison. An example of 2.5D DSM rasterization, also called Digital Elevation Model (DEM), is given in Figure 8. However, the rasterization process involves a loss of possibly interesting information, e.g., on building facades. Furthermore, depending on the chosen grid size, points are aggregated into a single value, leading to a potentially drastic loss of information with too large steps. On the other hand, a too thin grid size leads to plenty of empty pixels, generally filled with interpolation causing approximate data. In a similar spirit of DSM, a wild majority of methods in geosciences relies on Digital Terrain Model (DTM), i.e., a 2.5D raterization of PCs at the ground level (without vegetation or buildings), to highlight ground modification. Aside from 2D DSM or DTM rasterization, 3D rasterization into

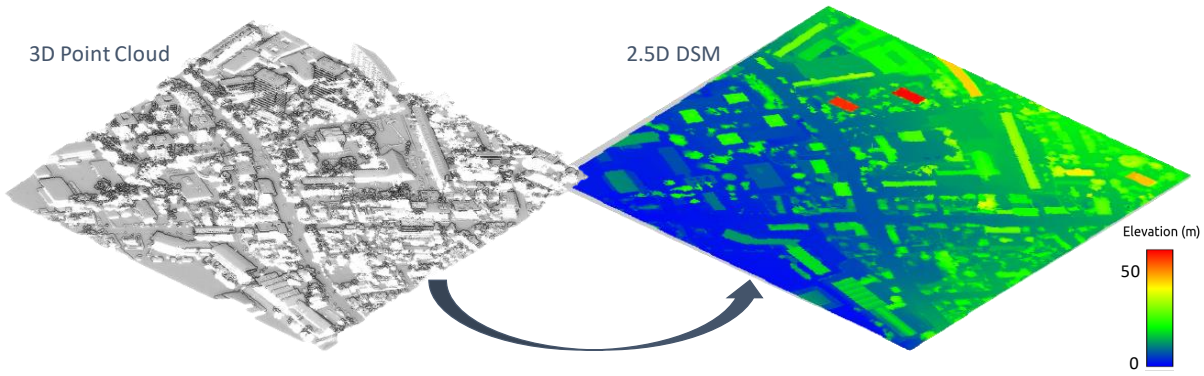


Figure 8: **Illustration of a 3D point cloud and its corresponding rasterization into 2.5D DSM.** The DSM is colorized as function of the height.

a 3D voxel grid is also a possibility to facilitate the processing. However, similarly to 2D rasterization, a large grid size also implies a loss of information, and a too thin grid size quickly becomes computationally expensive because of the sparse characteristic of 3D environments.

Deep learning for Earth observation

Thanks to the increase in computing capacity[4], the use of deep learning has become widespread in the recent decade. Deep learning is a subset of machine learning which uses mathematical and statistical approaches to give to computers the ability to learn from data to solve a given problem, also called task. Both machine and deep learning are part of what is called artificial intelligence (AI). While machine learning relies on hand-crafted features to represent the data, deep learning learns to extract its own features related to the task to be solved. From the first proposal of multi-layer perceptron (a perceptron being an artificial neuron) to the current deep networks, many improvements have been made. They can now tackle multiple tasks related to data understanding, thanks to fully supervised learning relying on annotated samples, or even unsupervised learning (i.e., without the use of labeled samples). Remote sensing and Earth observation also benefits from these methodological improvements (Zhu et al., 2017). In particular, deep learning has been successfully used for land cover classification (Kussul et al., 2017),

4. Experimentation conducted in this thesis has been made available thanks to the access to clusters from IRISA (`https://cluster-irisa.univ-ubs.fr`) and Jean Zay (`http://www.idris.fr/jean-zay/`).

object detection (Ding et al., 2021), satellite image time series analysis (Pelletier et al., 2019), super-resolution (Wang et al., 2020c), and last but not least, 2D images change detection (Daudt et al., 2018).

Given the great differences between 2D images and 3D PCs, and the complexity of the latter, the use of deep learning-based method for 3D PCs understanding has started later. However, since the emergence of PointNet (Qi et al., 2017a), deep learning has proven its ability to handle such complex data.

To our knowledge, at the beginning of the thesis, deep learning methods had never been used to tackle change detection in raw point clouds. However, given their ability to analyze complex data and the possibility of global understanding of scenes, it seems clear that the computation of changes in these particular data may be enhanced by deep learning. Therefore, this thesis aims **to investigate how to use deep learning concepts to detect modifications (i.e., changes) in our landscapes from 3D data**.

Objectives of the thesis

The **general objective** of this thesis is to explore the question of **change detection into three-dimensional data**, and in particular 3D **point clouds** using recent developments in **deep learning**. In particular, we aim at comparing two PCs acquired at two different dates such as in Figure 7, and further providing an identification of changed areas via a segmentation at point level of the second PC (i.e., the newer). Concerning 3D PCs change detection, binary (change/no-change) or multiple (nature of change) information can be extracted. Change categorization also refers to identification of changes among multiple classes. Then, similarly to 2D images, change detection methods can return either classification or segmentation results. In a change classification framework, results are obtained at the level of the PCs, i.e., one label for one pair of PCs. Conversely, change segmentation results are given at the point scale. The different settings of the change detection and categorization problem from 3D point clouds are summarized in Figure 9. In this thesis, we mainly aim to **tackle the multiple change segmentation task** (Figure 9d) giving a finer precision of results. Note that solving the multiple change segmentation task also solves the other tasks presented in Figure 9.

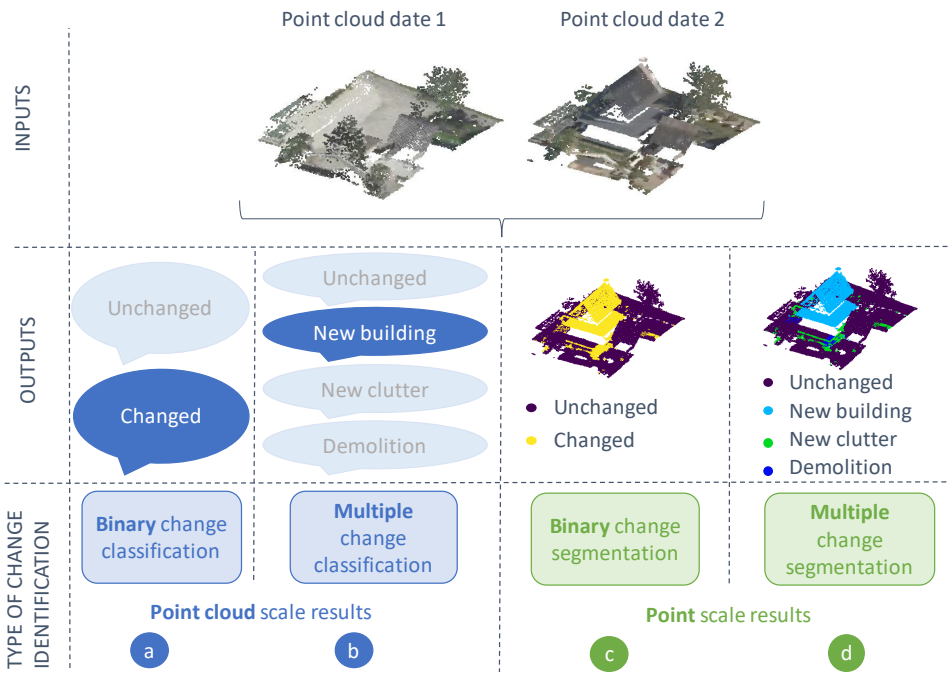


Figure 9: **Different types of change detection results**, at the scale of PCs (**a** and **b**) or points (**c** and **d**) and with binary (**a** and **c**) or multiple classes (**b** and **d**). This thesis aims to tackle the multiple change issue (**d**). Note that from this more complex task other tasks (**a**, **b** and **c**) can be solved as well.

To this end, the manuscript addresses the following research questions:

1. **How traditional methods deal with change detection in such complex 3D PCs data?** How state-of-the-art methods are highlighting changes? How do they classify different types of changes? How well they perform on this particular task? We propose to benchmark main existing methods on common datasets to evaluate their forces and weaknesses in various conditions.

2. **How to make use of recent progresses in deep learning to solve multiple change segmentation task in raw 3D PCs?** How to handle bi-temporal 3D PCs data in a deep neural network? How to compare two PCs, considering that there is no point-to-point correspondence conversely to pixels of registered 2D images and that even in unchanged areas point distribution are dissimilar? How to deal with remote sensing large scale PCs? Given the wide range of applications, how to develop method agnostic to the field of study? We propose to rely on both advances in 2D change detection deep learning framework and 3D deep learning to propose

different architectures dealing with multiple class change segmentation in raw 3D PCs. We show the effectiveness of using such deep learning approaches compared to traditional state-of-the-art methods in different use cases (urban and geosciences).

3. **How to deal with the complexity of obtaining annotated data?** How to train deep networks given the lack of public dataset with change-related annotation? Is it possible to rely on simulated data to train deep networks and to obtain interesting results on real data? How to diminish and even get rid of the need for annotated data to train deep models? We developed an original simulator to generate annotated synthetic bi-temporal PCs. Then, we also explore unsupervised learning paradigm to lighten the time-consuming annotation step.

What the thesis is not tackling

When using time series of 3D PCs, one first tricky step is PCs registration to obtain each PC in the same coordinate frame. Performing accurate registration is a research question in itself that we are not going to tackle. However, in the following experiments, PCs are registered together using traditional methods such as iterative closest point (ICP) (Besl and McKay, 1992) or its derivatives. We hope that developed methods will not be sensible to small registration errors.

While the initial scope of the thesis was related to PC time series in the context of the future CO3D mission, both the challenges raised by the bi-temporal change detection problem and the lack of (annotated) datasets for PC time series led us to focus on the change detection task.

Organization of the manuscript

Our manuscript is built around five main chapters.

Considering the compelling need for training data when machine learning is concerned, and to be able to benchmark existing methods on a common dataset, Chapter 1 focuses on reviewing potential existing multi-temporal PCs datasets. Observing the lack of annotated public data, this chapter proposes to adapt an existing dataset to change detection task and also provides a simulator to generate Urb3DCD, a synthetic annotated dataset suited for multiple change segmentation task in urban areas. Chapter 1 answers question 3 partially and enables development of methods for question 2 by providing training data.

Chapter 2 gives an overview of existing methods for change detection in 3D data. Then, seven different methods representative from the state-of-the-art based on distance computation, machine learning with hand-crafted features and even deep learning on 2D rasterization of PCs are compared in the various configuration of Urb3DCD sub-datasets. This chapter answers to question 1.

Based on the observation that none of the existing methods was trying deep learning for change detection in the raw 3D PCs, Chapter 3 explores existing deep learning methods (question 2) for 2D images change detection and for 3D PCs understanding to build upon these developments several architectures related to 3D PCs change detection task. In this chapter, an assessment of the methods in various conditions is performed.

In the view of the difficulty of 3D PCs annotation, Chapter 4 investigates how to reduce the amount of annotated data required in usual supervised deep learning settings, this completes the answer to question 3. Several strategies are explored based on transfer learning, self-supervised learning, deep change vector analysis and deep clustering.

While experiments in Chapter 2, 3 and 4 were conducted on existing public datasets all related to urban environment, Chapter 5 assesses proposed supervised methods in the geoscience context for cliff erosion identification and landslides detection in wide mountainous areas.

Finally, the Conclusion chapter summarizes the contributions of the manuscript and provides final conclusions of the thesis. This part also presents some perspectives either at short or long term to foster research in the understanding of our landscape modifications using 3D PCs data and deep learning methodology.

Publications

This thesis has led to the publication of various articles and communications in peer-reviewed journals and conferences, as detailed below:

Journal articles

J1 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. “Change Detection in Urban Point Clouds: An Experimental Comparison with Simulated 3D Datasets”. *Remote Sensing* 13 (2021): 2629. Link to the open-datasets: `https://ieee-dataport.org/open-access/urb3dcd-urban-point-clouds-simulated-dataset-3d-change`

`-detection`

J2 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Siamese KPConv: 3D multiple change detection from raw point clouds using deep learning". *ISPRS Journal of Photogrammetry and Remote Sensing* 197 (2023): 274-291. Link to the open-source codes: `https://github.com/IdeGelis/torch-points3d-SiameseKPConv`

J3 Iris de Gélis, Sudipan Saha, Muhammad Shahzad, Thomas Corpetti, Sébastien Lefèvre and Xiao Xiang Zhu. "Deep Unsupervised Learning for 3D ALS Point Clouds Change Detection". *arXiv preprint arXiv: 2305.03529* Link to the open-source codes: `https://github.com/IdeGelis/torch-points3d-SSL-DCVA`

J4 Iris de Gélis, Thomas Corpetti, and Sébastien Lefèvre. "Change detection needs change information: improving deep 3D point cloud change detection". *arXiv preprint arXiv: 2304.12639*. Link to the open-source codes: `https://github.com/IdeGelis/torch-points3d-SiamKPConvVariants`

J5 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "DC3DCD: unsupervised learning for multi-class 3D point cloud change detection". *arXiv preprint arXiv: 2305.05421*. Link to the open-source codes: `https://github.com/IdeGelis/torch-points3d-DC3DCD`

International conferences with proceedings

C1 Iris de Gélis, Sébastien Lefèvre, Thomas Corpetti, Thomas Ristorcelli, Chloé Thénoz and Pierre Lassalle. "Benchmarking change detection in urban 3D point clouds". *IEEE International Geoscience and Remote Sensing Symposium – IGARSS* (2021): 3352-3355.

C2 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "3D urban change detection with point cloud siamese networks". *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 43 (2021): 879-886.

C3 Iris de Gélis, Zoé Bessin, Pauline Letortu, Marion Jaud, Christophe Delacourt, Stéphane Costa, Olivier Maquaire, Robert Davidson, Thomas Corpetti, and Sébastien Lefèvre. "Cliff Change detection using Siamese KPConv deep network on 3D point clouds". *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* 3 (2022): 649-656.

C4 Iris de Gélis, Thomas Bernard, Dimitri Lague, Thomas Corpetti and Sébastien Lefèvre. "Landslide detection in 3D point clouds with deep Siamese convolutional network". *IEEE International Geoscience and Remote Sensing Symposium – IGARSS* (2023).

National conferences with proceedings

C5 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Détection de changements urbains 3D par un réseau Siamois sur nuage de points". *Journées francophones des jeunes chercheurs en vision par ordinateur – ORASIS* (2021).

International conferences without proceeding

O1 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Urb3DCD V2: An open dataset for urban 3D point clouds change detection". *Joint Urban Remote Sensing Event (JURSE)* (2022)

O2 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Can simulated data help when dealing with EO data? An illustration with 3D point cloud change detection". *ESA Living Planet Symposium* (2022)

National conferences without proceeding

O3 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Change detection in 3D point clouds based on deep learning". *Journées Jeunes Chercheurs – Centre National d'études Spatiales* (2021) – Best poster award

O4 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "3D point clouds change detection: From supervised to unsupervised deep learning methods". *Webinaire "Autour de la 3D"MAGIS/IG-RV* (2022)

O5 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Approche DeepCluster faiblement supervisée pour la détection de changement dans des nuages de points 3D". *GdR ISIS "Approches faiblement supervisées en Télédétection"* (2023)

O6 Iris de Gélis, Sébastien Lefèvre, and Thomas Corpetti. "Détection de changement en milieu urbain au sein de nuages de points 3D par apprentissage profond". *Télédetection pour l'étude du milieu urbain (TEMU)* (2023)

Chapter 1

Data for 3D change detection

Contents

As far as supervised machine learning is concerned, annotated data are required to train a model. Even in the case of an unsupervised method, some annotations on the testing set are welcome to perform a quantitative evaluation.

As this PhD work refers to multi-class change detection, datasets made of multi-temporal Point Clouds (PCs) associated with change annotations given at point level are needed. In practice, such datasets with a reliable level of annotation are rare, especially for point clouds, mainly because a manual annotation on such data is very time-consuming. In addition, it is difficult to automate the annotation and even more if multi-temporal comparisons are required because no point-to-point comparison is possible. Therefore, a manual annotation is often the only possibility.

This chapter is concerned with this specific issue. After giving an overview of existing multi-temporal datasets of 3D PCs (Section 1.1), in Section 1.2, we propose a strategy to adapt these existing datasets to our specific task: 3D PCs change segmentation. Indeed, this task requires change annotations at point level which, as will be seen in the first section, are not available in public remote sensing datasets. However, this semi-automatic annotation process is far from being perfect. As a consequence, in this chapter, we also propose a simulator allowing us to generate multi-temporal 3D PCs containing semantic changes. Using this simulator, we have created several datasets[5] under different acquisition conditions that are presented in Section 1.3.

1.1 Existing public datasets

Similarly to 2D images, during last years an increasing number of public annotated datasets has been published concerning 3D PCs mainly for semantic segmentation tasks. But unfortunately, a small number of datasets is available regarding remote sensing multi-temporal 3D PCs.

It should be noted that some public datasets containing only DSMs or DEMs are excluded from this overview of existing multi-temporal 3D datasets. Although the field is active for DSM change detection (Zhang et al., 2019; Coletta et al., 2022), it does not exactly fall within our scope of change detection within 3D PCs. Indeed, in this thesis, we focus on detecting changes directly into 3D PCs. Any rasterization process leads to a loss of information especially for 2.5D rasterizations where only one point (e.g., top point)

5. These datasets have been presented in two different journal publications: Remote Sensing in 2021 (de Gélis et al., 2021b), and ISPRS Journal of Photogrammetry and Remote Sensing in 2023 (de Gélis et al., 2023d).

in a 2D cell is considered. In this case, all information on building facades, for example, are lost.

Multi-temporal 3D point clouds datasets without annotation

When dealing with 3D data for earth observation, one should mention the OpenTopography[6] portal that provides access to terabytes of data over different areas in the world in various context (e.g., urban, forest, land, ...)(Krishnan et al., 2011; Crosby et al., 2011). While some places contain multi-temporal data, it does not come with any annotation. Some other examples of multi-temporal 3D datasets without any annotation are furnished in Table 1.1. Notice that this is not an exhaustive list.

Multi-temporal 3D point clouds datasets with mono-date annotation

Focusing on annotated datasets, a large majority of multi-temporal 3D PCs datasets only provides single-date semantic annotation. Though interesting, the single-date annotation only segments each PC separately into different objects (e.g., ground, building, vegetation, ...). For example, Actueel Hoogtebestand Nederland (AHN) (Sande et al., 2010), Abenberg-ALS (Hebel et al., 2013) and Hessigheim 3D (H3D) (Kölle et al., 2021) are ALS datasets containing at least two acquisitions and single-date semantic annotation at point level. Similarly, the TUM City Campus dataset (Gehrung et al., 2017; Zhu et al., 2020) is made of two Mobile Laser Scanning (MLS) acquisitions, and provides single-date semantic information in a total of 8 classes. Notice that Abenberg-ALS and TUM City Campus datasets only provide the training set.

Multi-temporal 3D point clouds datasets with change-related annotation

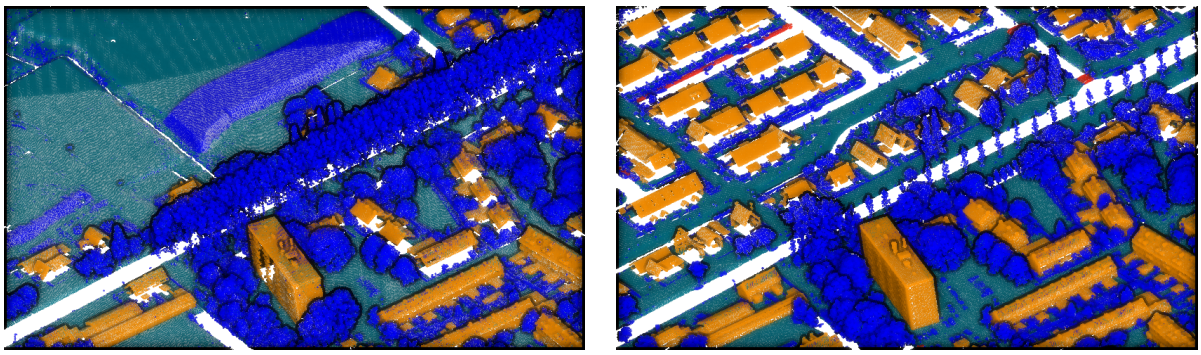
Finally, in terms of datasets providing a change annotation, only a few papers can be found in the literature. One can mention the Change3D dataset (Ku et al., 2021) and the 2017 Change Detection dataset (Palazzolo and Stachniss, 2018). However, both of these datasets do not contain annotation at 3D point level. The 2017 Change Detection dataset is composed of images, and corresponding 3D meshes derived from a photogrammetric process. The annotation is provided for main changed object on corresponding 2D images. This dataset only contains four scenes with one changed object. A summary of existing

6. The platform is available at the following link: `https://opentopography.org/`, accessed on 27/02/2023.

public datasets is presented in Table 1.1. This list is not exhaustive for datasets without annotation and with only single-date semantic annotation. As can be shown in this table, most of them have been released recently (since 2020 or later) and concern mainly bi-temporal data. For longer temporal monitoring, additional acquisitions can be scheduled, as done in the Kijkduin beach–dune dataset (Vos et al., 2022) where acquisitions have been made every hour for six months. Three of these datasets (AHN, H3D and Change3D) are described in more detail later in this section.

Dataset	Annot. type		Annot. level			Sensor	Nb of dates	Type	Ref.
	Change	Sem.	PC	Pt	Img				
OpenTopography	None			None		LiDAR		Multiple	(Krishnan et al., 2011; Crosby et al., 2011)
CG-PB-M3C2	None			None		TLS	6	Glacier	(Zahs et al., 2022)
Kijkduin beach–dune	None			None		TLS	4,000	Coastal	(Vos et al., 2022)
Abenberg-ALS		X		X		ALS	2	Urban	(Hebel et al., 2013)
AHN		X		X		ALS	4	Multiple	(Sande et al., 2010)
						UAV LiDAR			
H3D		X		X		Photogra.	4	Urban	(Kölle et al., 2021)
TUM City Campus		X		X		MLS	2	Urban	(Gehrung et al., 2017; Zhu et al., 2020)
2017 Change Detection	X				X	Photogra.	2	Urban	(Palazzolo and Stachniss, 2018)
Change3D	X		X			MLS	2	Urban	(Ku et al., 2021)
AHN-CD	X	X		X		ALS	2	Urban	Ours (de Gélis et al., 2023d)
Urb3DCD-V1	X			X		ALS	2	Urban	Ours (de Gélis et al., 2021b)
Urb3DCD-V2	X	X		X		ALS	2	Urban	Ours (de Gélis et al., 2023d)
Urb3DCD-Cls	X		X			ALS	2	Urban	Ours (de Gélis et al., 2023d)

Table 1.1: **Existing public datasets for multi-temporal 3D data.** The date corresponds to the year when the dataset has been set publicly available. This list is not exhaustive for datasets without annotation and with only single-date semantic annotation. The bottom part corresponds to our contributions and will be detailed in Section 1.2 and 1.3. Annot. stands for annotation; Sem. for semantic; Pt for point; Img for image; Nb for number; UAV for unmanned aerial vehicle; Photogra. for photogrammetry; Ref. for reference.



(**a**) AHN3 acquisition (**b**) AHN4 acquisition

Figure 1.1: **Example of AHN multi-temporal dataset** colorized as a function of mono-date semantic annotation furnished with 3D PCs. Green represents ground points, orange building points, red civil engineering points and blue clutter points.

Actueel Hoogtebestand Nederland (AHN) dataset

The Netherlands government was the first to provide a full coverage of their country by ALS data (Sande et al., 2010). Since 2003, a total of four surveys has been delivered, constituting the AHN dataset. Acquired in the early 2000s, the first survey has a density ranging from 0.06 to 1 points/m^2. The density of second and third versions (noted AHN2 and AHN3) varies from 10 to 14 points/m^2 while AHN4 varies from 10 to 24 points/m^2. Because of the acquisition process, height and planimetric stochastic errors are 5 cm for the three last versions. In addition to point coordinates, AHN data also includes RGB colors, LiDAR intensity and the number of returns. Besides 3D data, the third and fourth versions contain an annotation of points into 5 categories: ground, buildings, water, civil engineering structures (e.g., bridges) and clutter. The classification is first made automatically as a function of the height and number of echoes, then a manual correction is performed to enhance the quality of the provided annotation. An example of two versions of AHN dataset (AHN3 and AHN4) is given in Figure 1.1.

Pros and Cons of this dataset

Although very dense (the entire Netherlands is acquired) and multi-temporal, this dataset does not contain any annotation concerning changes, but labels are rather related to the semantic of the PC at a specific date. Furthermore, the semantic annotation is quite general, in particular the 'clutter' class contains a lot of different objects (e.g., vegetation, cars, etc.), see blue points in Figure 1.1.

Figure 1.2: **Example of H3D dataset** (Kölle et al., 2021). The 11 classes are represented in the different colors. Source: Image from (Kölle et al., 2021).

Hessigheim 3D (H3D) dataset

The H3D dataset consists of four different PCs at various dates and comes with labels related to 11 semantic classes that have been manually annotated (Kölle et al., 2021). Three of the acquisitions (March 2018, November 2018 and March 2019) have been captured over the same area with the same sensor configuration. The sensor platform includes a LiDAR and cameras to perform multi-view-stereo photogrammetry. This results in three acquisitions at a very high density (800 points/m^2) as well as the corresponding 3D meshes. H3D also includes a LiDAR-only epoch (March 2016) captured from an ALS campaign with a sparse density and less accurate data since it has been acquired at large scale by the Netherlands national mapping agency. Figure 1.2 shows an example of H3D PC.

Pros and Cons of this dataset

This dataset contains high density data that are manually annotated. The annotation is very precise, but, similarly to the AHN dataset, the annotation is not temporal. Again, this dataset cannot be directly used for our purpose.

Figure 1.3: **Example of a scene from the Change3D dataset** with the points of interests and their corresponding labels taken from Ku et al. (2021).

Labels	No change	Removed	Added	Change	Color change	Total
Training set	351 (59.79%)	54 (9.20%)	100 (17.03%)	63 (10.73%)	19 (3.24%)	587
Testing set	90 (58.44%)	25 (16.23%)	15 (9.74%)	17 (11.04%)	7 (4.55%)	154

Table 1.2: **Class distribution for the Change3D training and testing splits.** For each class, the number of samples along with the class proportion (in %) is given.

Change3D

Change3D is a dataset provided by CycloMedia Technology for the Shape Retrieval Challenge 2021 (SHREC21) track. This dataset consists of pairs of PCs from 2016 and 2020 in street scenes acquired over the city of Schiedam, The Netherlands. It aims at detecting changes from bi-temporal PCs in a complex street environment (Ku et al., 2021). PCs are acquired thanks to LiDAR sensors mounted on vehicles, and RGB information for each point is also provided. In these 78 3D scenes, 741 urban objects, also called points of interest, are identified by their coordinates and an associated label (see Figure 1.3 for a scene example). Urban objects correspond for example to road signs, advertisements, statues, or garbage bins. Each point of interest is manually annotated into one of the following classes: no change, removed, added, change or color change. The distribution of annotated objects in the training and testing splits is given in Table 1.2. As can be seen, one major constraint when using this dataset for learning purposes is its highly imbalanced settings, thus making the training set on less represented class very restricted.

Pros and Cons of this dataset

Change3D dataset has been specifically designed for 3D multi-change classification task. However, as explained, in the Introduction (see Figure 9), in this thesis we wish to tackle multi-class change segmentation. Therefore, the annotation of this dataset (at the scene level, i.e., at the point cloud scale results, see Introduction Figure 9b) is not precise enough to meet our objectives.

According to this section, none of the existing public datasets are suitable for 3D PCs multiple change segmentation task. Therefore, in the following sections, we will propose to adapt an existing public dataset to change segmentation, and we will introduce a simulator to generate annotated 3D PCs datasets.

1.2 Adaptation of public dataset to change detection task

As seen in the previous section, available public datasets with multi-temporal PCs do not contain annotation concerning changes at point level. Thereby, in the context of this study, we need to adapt an existing dataset to change segmentation task. A first option is to perform manual annotation. However, this solution is very time-consuming and does not scale for large areas. Thus, we rather prefer a semi-automatic change annotation process based on existing semantic segmentation labels. This allows us to derive a change detection version of AHN public dataset, noted AHN Change Detection (AHN-CD).

1.2.1 Semi-automatic annotation of AHN-CD dataset

Using some bi-temporal data issued from AHN3 and AHN4, we define four labels of change: unchanged, new building, demolition and new clutter. Our choice was motivated by the relative imprecision of labels for other possible classes of change. Notice that we decided to fuse the class “bridge” with the class “clutter” since the former contains only a few points. Over selected areas, there are almost no points concerning water, thus remaining water points are deleted.

When dealing with 3D PCs, comparison of point labels is not obvious since there is no direct corresponding pair of points between PCs (refer to the Introduction chapter for more details about PCs characteristics). A naive solution is to take the nearest point in PC from AHN3 for comparison with labels of points in AHN4, but this yields very

noisy results. By doing so, numerous errors are likely to occur, due to occlusions or to object proximity. For example, in dense urban areas, points on new building roofs can be associated with a point on the roof of a neighbor building present at the older date, thereby the point is then annotated as unchanged. Direct nearest point label comparison leads to numerous other similar failures in annotation. As a consequence, a more complex processing chain has been chosen, illustrated in Figure 1.4. In particular, to obtain smoother annotations, a label is given to each 3D connected component (CC) by majority vote when comparing each point of the CC to the nearest 3D point in AHN3 for new building and new clutter classes. Concerning demolition, a first extraction of potential demolition points is made by comparing to the 2D nearest point in AHN3. 2D nearest points are found by removing the Z coordinate of points in the search for the nearest neighbor. A first removal of isolated points is made automatically by removing smaller demolition CCs. Finally, a manual assessment is preferred to distinguish between real demolition and building shadows. Figure 1.5 shows an example of the annotation derived from this semi-automatic process.

1.2.2 AHN-CD properties

As previously mentioned, AHN provides full coverage of the Netherlands. So we select some tiles to define our training, validation and testing sets. Selected areas are shown in Figure 1.6. Tiles have been chosen in areas where changes occurred between AHN3 and AHN4. In particular, we chose the following tiles: 31HN1_22, 31HN1_23, 31HZ1_04, 37FN1_06, 37EN1_08 and 37EN1_13 from the divided AHN dataset provided by the website `https://geotiles.nl/` (accessed on 27/02/2023). Indeed, divided tiles are easier to manipulate than original AHN tiles which cover large areas. Some of these tiles are only partly taken to focus on places containing changes.

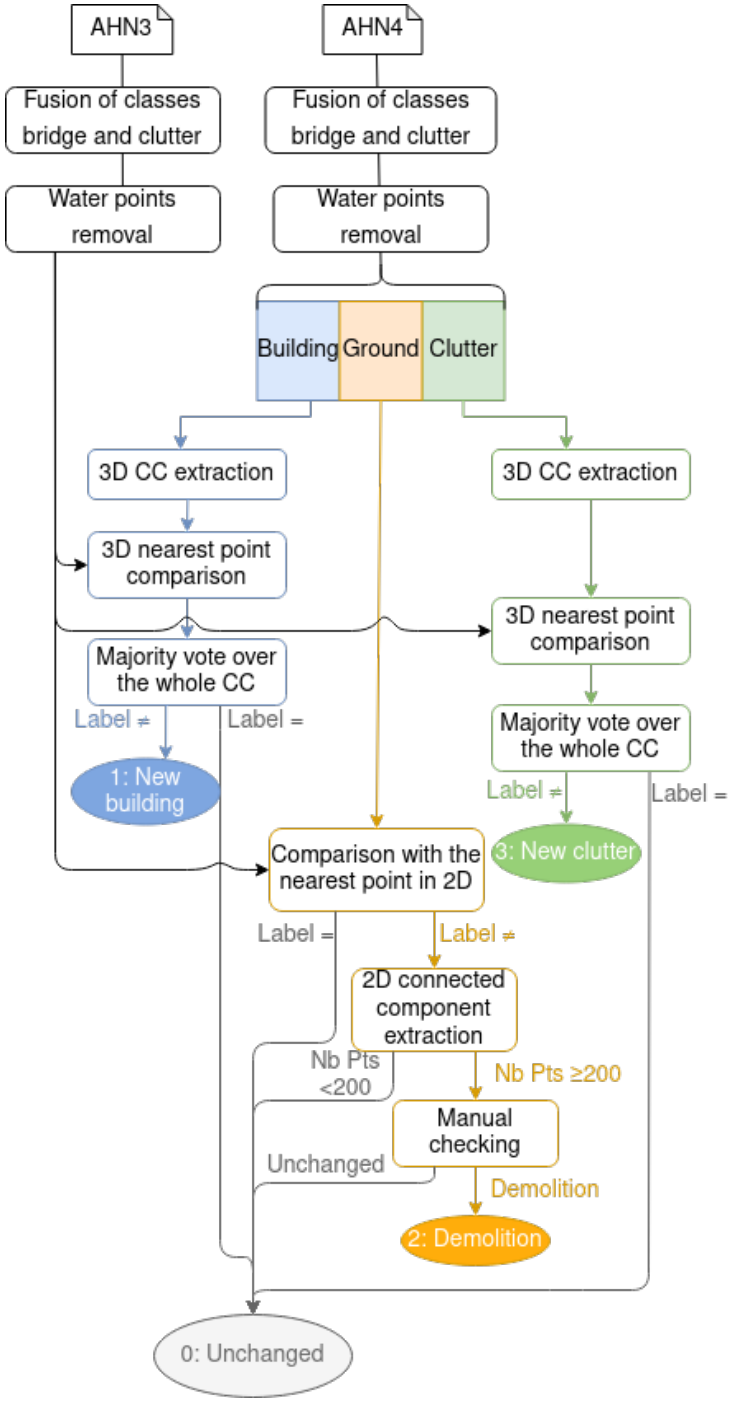


Figure 1.4: **Flowchart for change detection annotation of AHN pairs** (a.k.a. AHN-CD) into four classes: unchanged, new building, demolition and new clutter. CC stands for connected component.

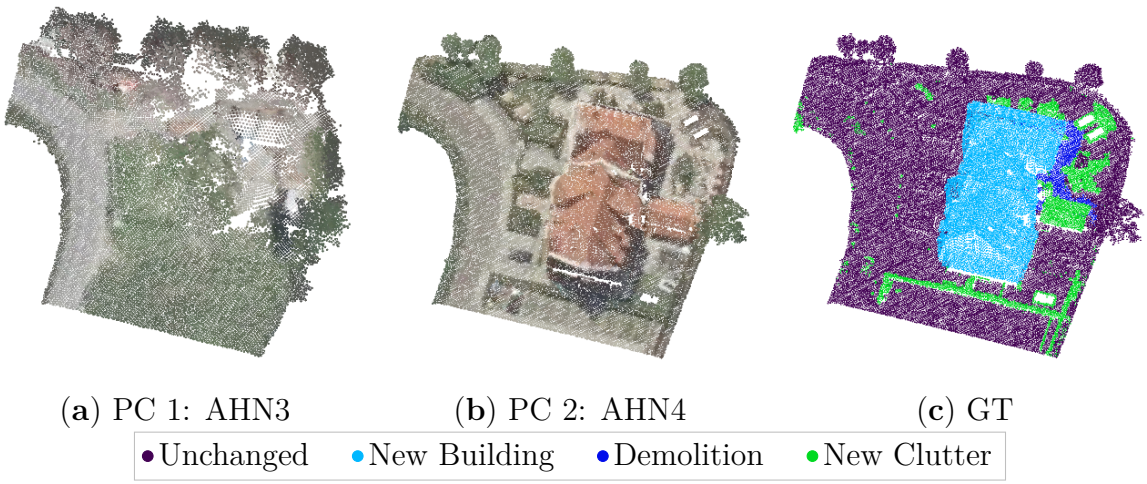


Figure 1.5: **Sample extracted from AHN-CD dataset.**

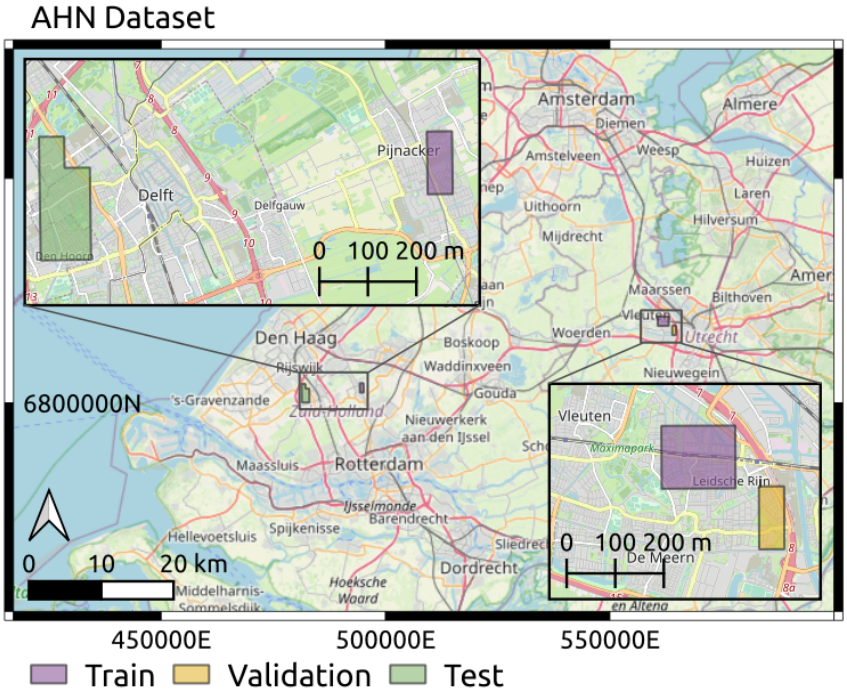
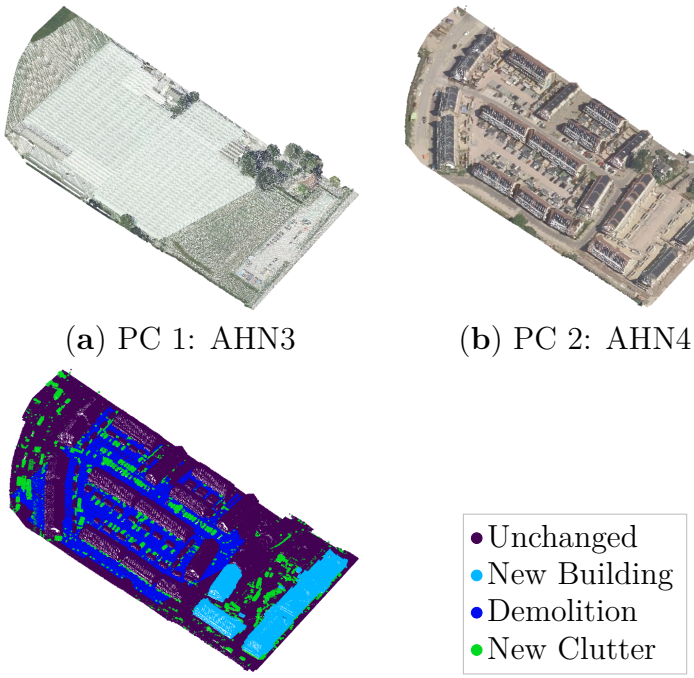


Figure 1.6: **Selected parts of AHN-CD dataset for training, validation and testing.**



(**a**) PC 1: AHN3 (**b**) PC 2: AHN4

(**c**) GT

Figure 1.7: **Sample from AHN-CD dataset illustrating some ground truth errors.**

1.2.3 On AHN-CD quality

However, this process has some limitations. First of all, in order to obtain our annotations, we performed an automatic comparison of the two PCs, leading to a lot of misclassifications, since objects may have changed even if the label has not. To illustrate this, one can focus on the left side of the house in Figure 1.5. With manual processing, the small garden would have been annotated as new clutter because it is totally different to the vegetation existing previously in AHN3 (Figure 1.5a), yielding difficulties in practice. Another example is given in Figure 1.7 where we can observe a lot of new buildings omitted by the ground truth. Indeed, in AHN3 the whole surface was covered by a glasshouse marked as a building in the AHN classification. Therefore, in the label comparison step of our annotation processing chain, new buildings were overlooked. Furthermore, it should be outlined that the AHN classification of the term 'building' itself does not have exactly the same definition for the building class for AHN3 and for AHN4.

Another difficulty comes with the clutter class of AHN, which is a mix of various types of objects, ranging from all kinds of vegetation to cars or rubble. The boundary between

mAcc (%)	mIoU_{ch} (%)	Per class IoU (%)			
		Unchanged	New building	Demolition	New clutter
87.34	81.08	78.70	57.19	97.98	88.06

Table 1.3: **Evaluation of AHN-CD semi-automatic annotation compared to the manual annotation** (considered as the ground truth). Metrics are defined in Chapter 3, Equation 3.5 for the mean of accuracy (mAcc) and in Chapter 2, Equation 2.5 for the Intersection over Union (IoU) and mIoU_{ch}.

the clutter and building classes in AHN annotation is not very clear in some cases, for example when dealing with garden sheds, as visible on the right side of the house in Figure 1.5, the shed is marked as clutter in the annotation.

Though the automatic process leads to a quick annotation of the dataset, it generates a lot of misclassifications because of the difficulty of comparing one point of a PC with several points in the other PC. Occlusions due to shadows of buildings, for example, make the comparison and the manual checking over demolition part even more complex. To be precise in the annotation, the manual checking should be applied to all classes, but this is very time-consuming for large areas. Furthermore, such manual checking would not have been sufficient to solve problems in clutter class or on objects that have changed but kept the same semantics.

Finally, a sub-part of the AHN-CD testing set has been manually annotated to guarantee the consistency in area where the ground truth is entirely reliable. The sub-area has been chosen to be representative of each class of change. It contains a total of 707,199 points distributed as follows: 61% 'unchanged', 29% 'new building', 7% 'demolition' and 3% 'new clutter'. The selected area is about 12,400 m^2. To assess the semi-automatic process for annotation, a quantitative evaluation has been performed comparing the semi-automatic annotation and the manual annotation considered as the ground truth. Results are available in Table 1.3. These results confirm qualitative assessment of the dataset previously presented.

1.3 Proposed simulated datasets for point cloud change detection

Due to the difficulty of having accurate annotated datasets for 3D PCs change detection (see Tabel 1.3), we decided to create a simulator of urban ALS PCs. This simulator allows

us to bypass time-consuming manual annotation and errors coming from the automation of the annotation process.

1.3.1 Simulator of urban 3D point clouds

Principles of the simulator

We have developed a simulator to generate time series of PCs for urban datasets. Given a 3D model of a city, the simulator allows us to introduce random changes in the model and generates a synthetic aerial LiDAR Survey above the city. The 3D model is in practice issued from a real city, e.g., with Level of Detail 2 (LoD2) precision. The model was provided in CityGML format, and we converted it to an OBJ file. From this model, we extracted each existing building and the ground by parsing the input OBJ file corresponding to the city 3D model. By adding or removing buildings in the model, we can simulate construction or demolition of buildings. Notice that depending on the area, the ground is not necessarily flat. The simulator allow us to obtain as many 3D PCs over urban changed areas as needed. It could be useful, especially for deep learning supervised approaches that require lots of training examples. Moreover, the created PCs are all directly annotated by the simulator according to the changes. As in each time-stamp 3D city models are created by the simulator, the changes between two acquisitions directly depend on the new/old objects added/removed by the simulator, and their annotation is trivial. Thus, no time-consuming manual annotation needs to be done with this process. The overall framework of the simulator is presented in Figure 1.8.

This simulator has been developed in Python 3. For each obtained model, the ALS simulation is performed thanks to a flight plan and ray tracing with the Visualisation ToolKit (VTK) Python library [7]. Figure 1.9 gives an example of PCs at two timestamps generated by the simulator. The second PC is labeled according to the changes between the two timestamps generated by the simulator. Each simulation takes between a few seconds and half an hour according to model's dimensions and PC density. A simulation is computed on a single central processing unit (CPU). Space between flight lines is computed in accordance with predefined parameters such as density, covering between swaths and scanning angle. Following this computation, a flight plan is set with a random starting position and direction of flight in order to introduce more variability between two acquisitions. Moreover, Gaussian noise can be added to simulate errors and lack of

7. `https://vtk.org/`, accessed on 27/02/2023.

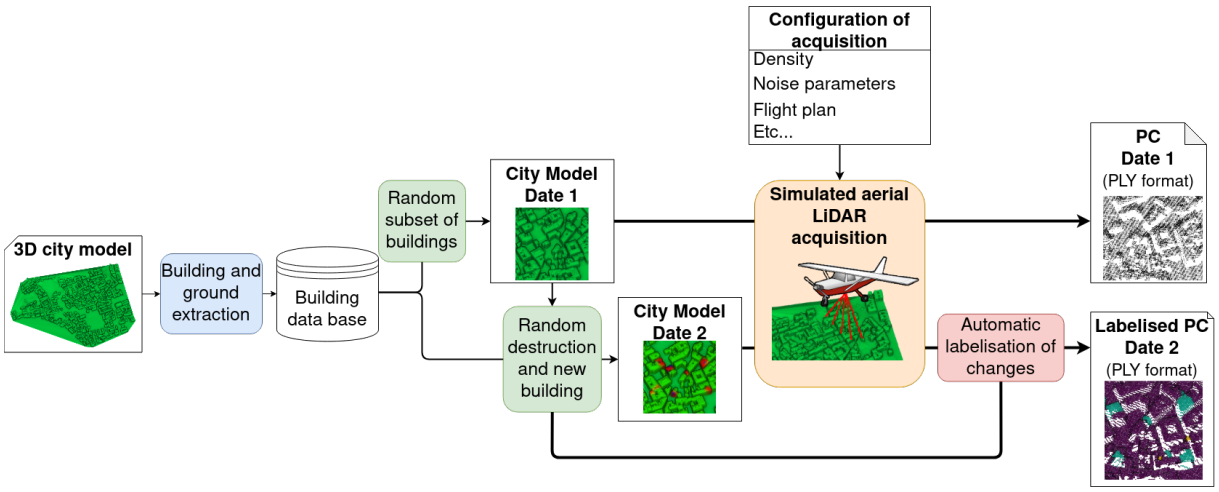
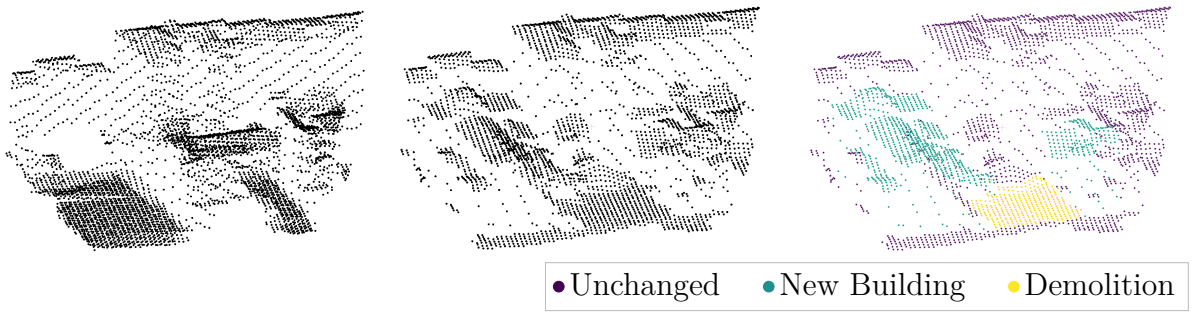


Figure 1.8: **Overall framework of the first version of the simulator generating bi-temporal urban 3D PCs.**

precision in LiDAR range measuring and scan direction.



(**a**) PC 1 (**b**) PC 2 (**c**) Labeled changes on PC 2

Figure 1.9: **Sample of simulated PCs at two timestamps** (**a**,**b**), with new buildings (green), demolished buildings (yellow) and unchanged objects (purple) in (**c**).

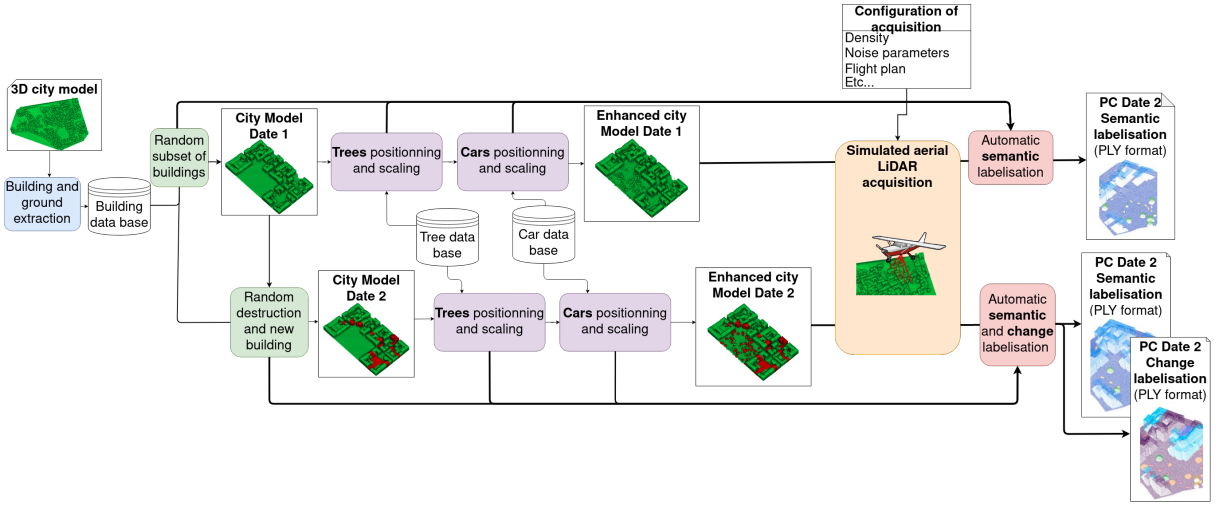
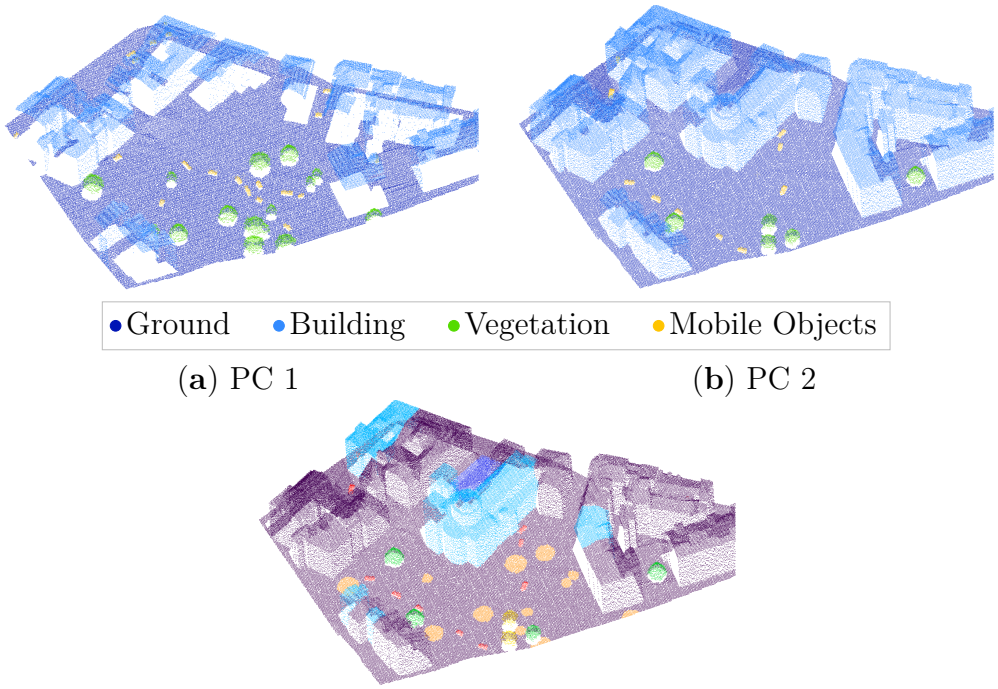


Figure 1.10: **Overall framework of the second version of the simulator generating bi-temporal urban 3D PCs.**

Improvement of the simulator

We also propose an improved version of our simulator of urban 3D PCs to make it more realistic. Vegetation has been added thanks to tree models created with Arbaro software (Diestel, 2003). Three different models have been chosen and added to the city in bare ground areas. Trees have been randomly scaled and rotated around the vertical axis to add diversity to the vegetation. Between each city model, changes have also been introduced into vegetation: some trees have been added, some have been deleted as if they have been cut and finally, we simulated tree growth between timestamps. Moreover, some mobile objects (cars and trucks) have also been added in streets. The size of mobile objects, and in particular their length, is also set randomly, within a realistic range. All mobile objects are randomly placed in the 3D model so that there is no collision with other objects. Because the aim of our study is to retrieve long-term changes only, mobile objects are assigned a single class in the change-related annotations. The flowchart of this second version of the simulator is shown in Figure 1.10. We illustrate a pair of simulated bi-temporal PCs in Figure 1.11. As can be seen on this figure, the new version of the simulator also enables the annotation for single-date semantic segmentation.



(**a**) PC 1 (**b**) PC 2

(**c**) Labeled changes on PC 2

Figure 1.11: **Sample of simulated PCs with the second version of the simulator** at two timestamps (**a**,**b**) with the corresponding 7 types of changes simulated in (**c**).

1.3.2 Urb3DCD: simulated datasets in different conditions

Notice that Urb3DCD datasets as well as their classification variant are made publicly available[8].

Version 1

To conduct fair qualitative and quantitative evaluation of PC change detection techniques, we have built some datasets based on LoD2 models of the first and second districts of Lyon, France[9]. For each simulation, buildings have been added or removed to introduce changes into the model and to generate a large number of pairs of PCs. We also considered various initial states across simulations, and randomly updated the set of buildings from the first date through random additions or deletions of buildings to create the second landscape. In addition, flight starting position and direction were always set randomly. As a consequence, the acquisition patterns were not the same among the PCs generated; thus, each acquisition may not have had exactly the same visible or hidden parts. In order to illustrate various results of a given simulation over a same area, Figure 1.12 shows three pairs of PCs generated by running the simulator three times over the same location. One can see that even if the input model is the same, the resulting pairs are different. Moreover, this figure also illustrates the different flight plans for each acquisition. Indeed, the scan lines are not oriented similarly; thus, different facades are visible. This can also be noticed in Figure 1.9(a-b) through a pair of PCs. However, to be sure that no redundancy exists among training, validation and testing sets, the whole area was separated into three distinct parts to constitute a proper split, as illustrated in Figure 1.13.

From terrestrial LiDAR surveying to photogrammetric acquisition by satellite images, many different types of sensors and acquisition pipelines exist to obtain 3D PCs for urban areas, resulting in PCs with different characteristics. By setting different acquisition parameters to our simulator, our goal is to provide a variety of sub-datasets with heterogeneous qualities to reproduce the real variability of LiDAR sensors or to mimic datasets coming from a photogrammetric pipeline with satellite images (by using a tight scan angle with high noise).

8. `https://ieee-dataport.org/open-access/urb3dcd-urban-point-clouds-simulated-dataset-3d-change-detection`, accessed on 27/02/2023.

9. `https://geo.data.gouv.fr/datasets/07319893497428 67f8e659b4d70b707612bece89`, accessed on 27/02/2023.

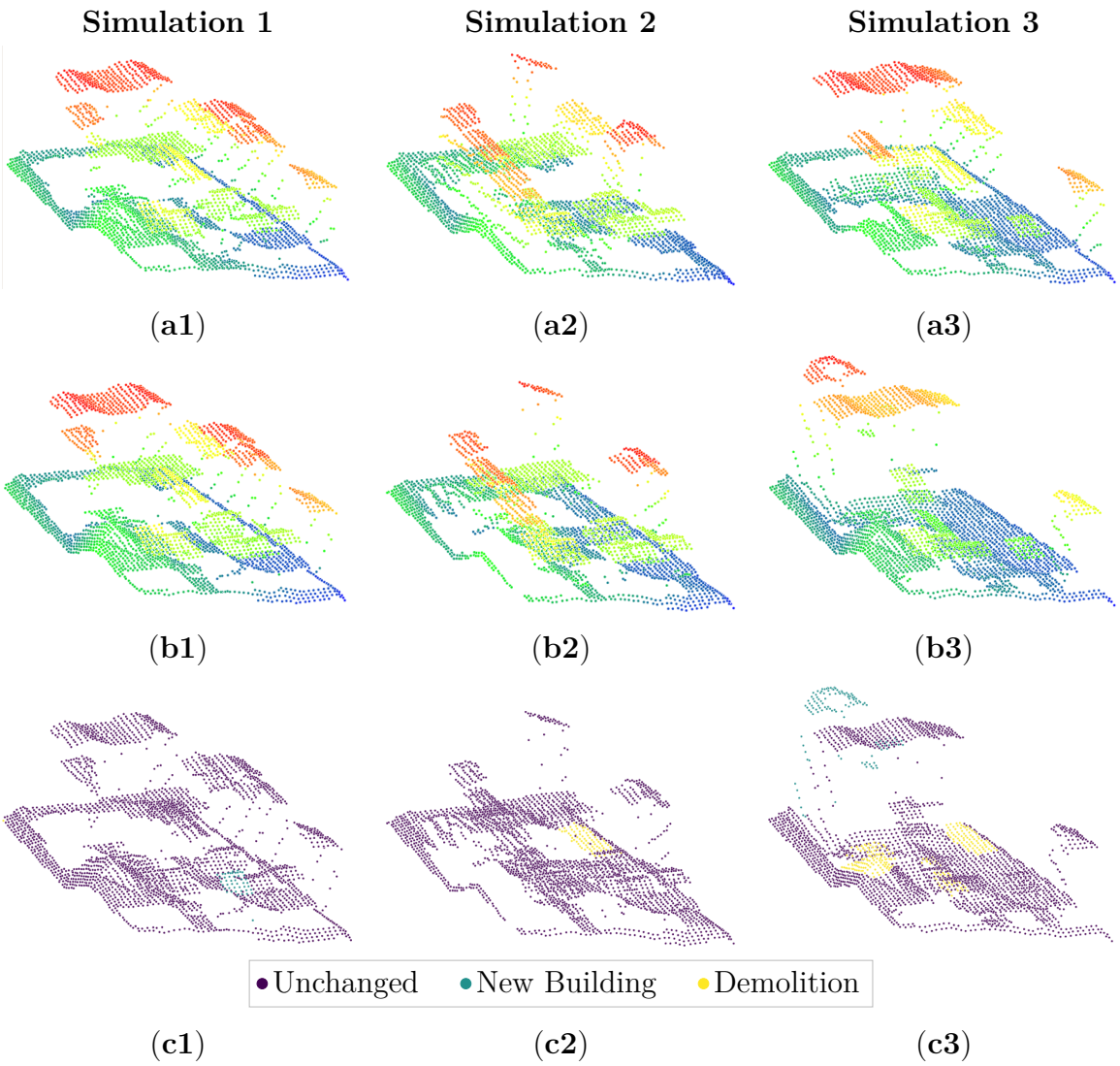


Figure 1.12: **Three simulation results generated from the same district of Lyon.** Elevation (relative to each PC) is shown in color for the first date (**a**) and the second date (**b**). The automatic change annotations on the second PC are presented in (**c**).

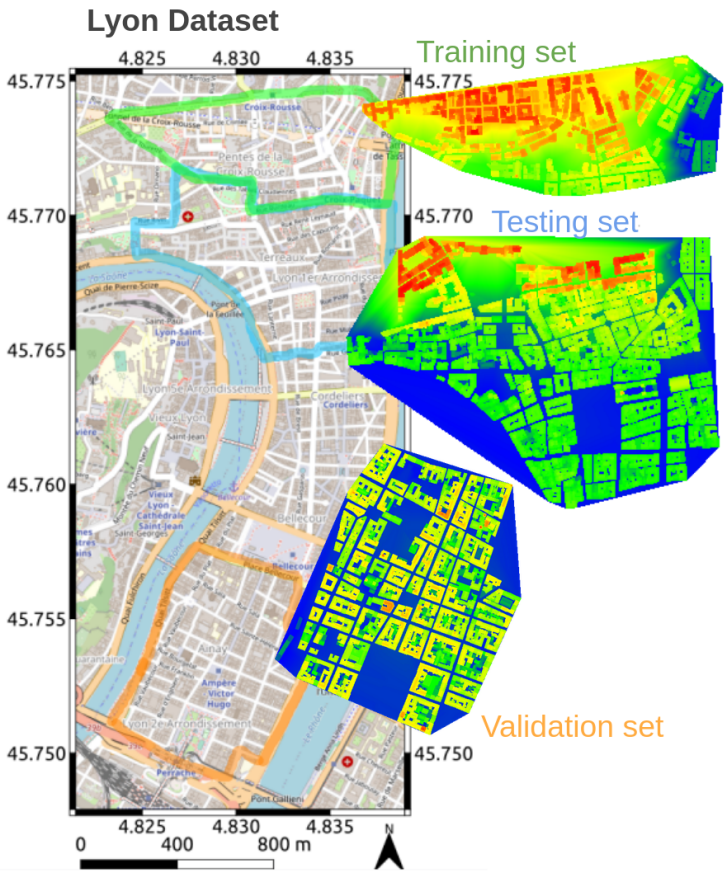


Figure 1.13: **Urb3DCD dataset splits into 3 distinct parts**: training, validation and testing sets over the city of Lyon, France. Elevation is shown in color and is relative to each sub-set.

Thus, we generated the following sub-datasets:

1. ALS with low density, low noise for both dates;
2. ALS with high density, low noise for both dates;
3. ALS with low density, high noise for both dates;
4. ALS with low density, high noise, tight scan angle (mimicking photogrammetric acquisition from satellite images) for both dates;
5. Multi-Sensor (MS) data, with low density, high noise at date 1, and high density, low noise at date 2.

Parameters	**Urb3DCD-V1 Sub-Datasets**							
	1			**2**	**3**	**4**	**5**	
	a	**b**	**c**				PC 1	PC 2
Amount of training pairs	1	10	50	10	10	10	10	
Density (points/m^2)	0.5			10	0.5	0.5	0.5	10
Noise range across track (°)	0.01			0.01	0.2	0.2	0.2	0.01
Noise range along track (°)	0			0	0.2	0.2	0.2	0
Noise scan direction (m)	0.05			0.05	1	1	1	0.05
Scan angle (°)	−20 to 20			−20 to 20	−20 to 20	−10 to 10	−20 to 20	
Overlapping (%)	10			10	10	5	10	
Height of flight (m)	700			700	700	700	700	
Annotation level	Point			Point	Point	Point	Point	

Table 1.4: **Acquisition configurations for all sub-datasets of Urb3DCD-V1.**

Notice that sub-datasets 3 and 4 are quite similar, but the latter provides less visible facades, due to the smaller scanning angle and overlapping percentage.

Finally, for the first configuration (ALS low density, low noise), we provided the three following different training sets:

(1.a) Small training set: 1 simulation;

(1.b) Medium training set: 10 simulations;

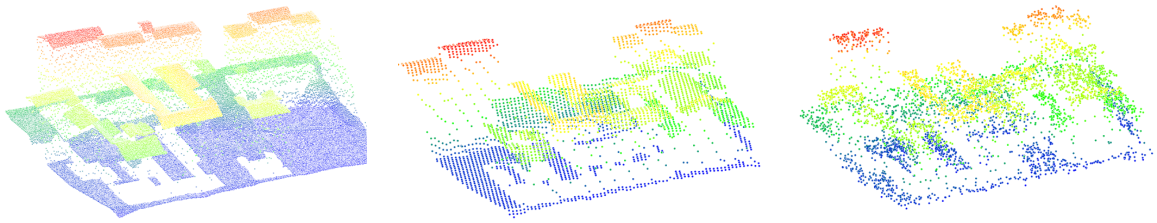
(1.c) Large training set: 50 simulations.

By doing so, we wish to analyze the sensitivity of the methods with respect to the size of the training set.

A summary of all acquisition configurations is given in Table 1.4. As for validation area, only one simulation was performed on a larger area. In order to have statistically relevant results, the testing set was composed of three simulations.

Figure 1.14 shows illustrations of sub-datasets 1.b, 2 and 3 to highlight differences in terms of density and noise. Let us outline that, as expected, the multi-sensor sub-dataset 5 has a similar PC to one belonging to sub-dataset 3 (Figure 1.14c) and another PC similar to one of sub-dataset 2 (Figure 1.14a).

Note that only the density has changed between sub-datasets 1 and 2. As a consequence, one could have simply performed a sub-sampling of sub-dataset 2. However, this would have required a sub-sampling strategy (e.g., grid sub-sampling), which would have



(**a**) Sub-dataset 2 (**b**) Sub-dataset 1 a-b-c (**c**) Sub-dataset 3

Figure 1.14: **Samples of sub-dataset 2 with high density** (**a**), **sub-dataset 1 with low density** (**b**) **and sub-dataset 3 with low density and high noise** (**c**). Height is shown in color.

led to less realistic results than the proposed acquisition simulation tuned to a lower density. Therefore, for each sub-dataset, the simulator has been used with the configuration given in Table 1.4.

Version 2

Using the second version of the simulator, we built another version of Urb3DCD dataset containing two sub-datasets. The first sub-dataset has been simulated in very low density conditions (0.5 points/m^2) with a very low noise level. This low density should be challenging for detecting small objects such as cars. According to comparison made on Urb3DCD-V1 sub-datasets (see Chapter 2), the most challenging data configuration for change detection methods is the multi-sensor setting. Thus, we decided to simulate another sub-dataset with a first PC with low density, high noise (mimicking PCs coming from satellite photogrammetry) and a second PC with a higher density and very low noise (mimicking aerial LiDAR acquisition). Acquisition configurations are summarized in Table 1.3.2. Training, validation and testing areas are similar to Urb3DCD-V1 (see Figure 1.13). Similarly to the previous version of the dataset, we run 10, 1 and 3 simulations over each training, validation and testing areas respectively, in order to constitute a dataset large enough for training and testing methods. Notice that the annotation is given at the point scale. Thereby, 10 simulations over the training area corresponds to about 1.5 million of labeled points for the low density sub-dataset and 20 times more for the MS sub-dataset (as the labeled PC is the second one and it has a finer resolution).

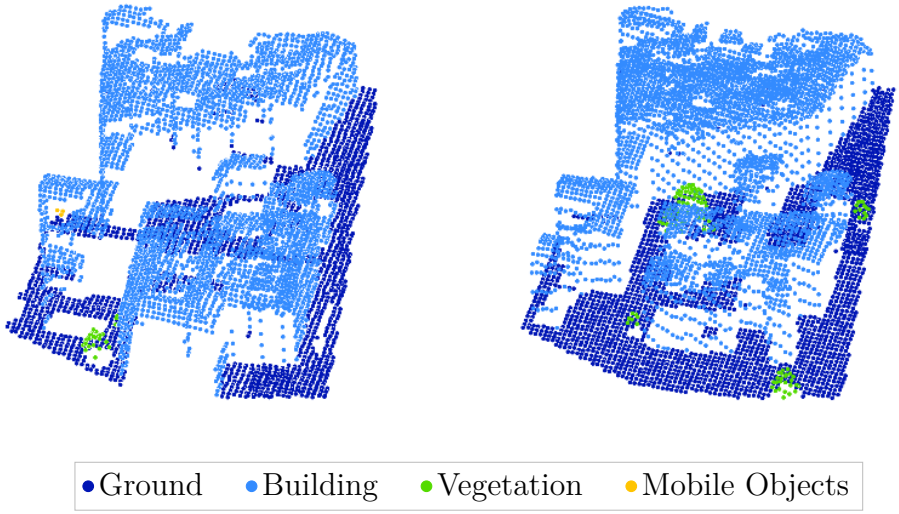
Another particularity of these simulated datasets is that PCs contain occluded parts. Occlusions are very typical of 3D PCs data because of objects shadows. In dense urban areas, some parts could be very hard to sense using only ALS campaigns. Hence, we

decided to simulate these 3D PCs artifacts to enhance the realism of our synthetic dataset and challenge change detection methods on this particular problem. Indeed, by varying the flight plan of the simulated ALS, occluded areas differ between the two acquisitions. An example of such occlusion is given in Figure 1.15(a-b) where building facades without point (hidden facades) are not at the same location in the two acquisitions.

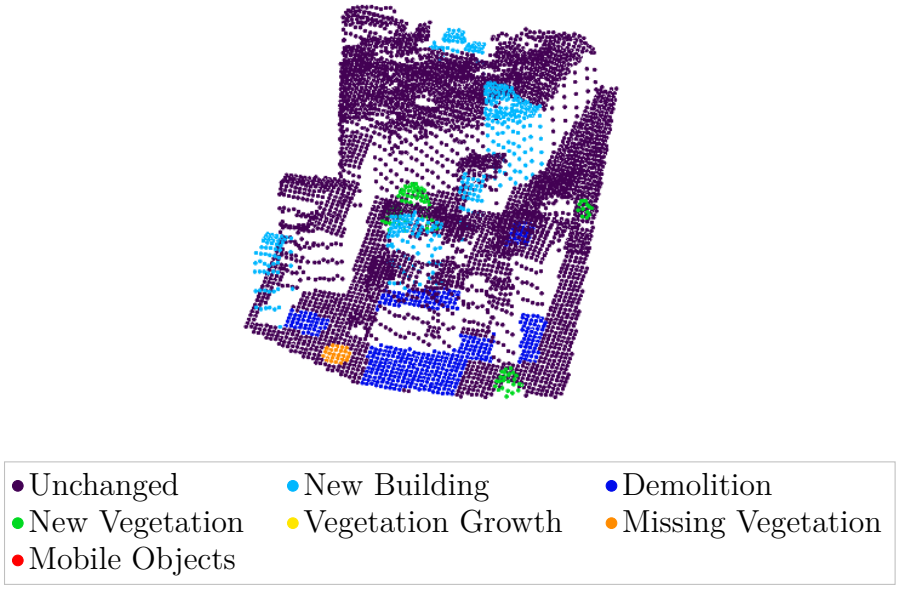
In the following work, this dataset is referred to as Urb3DCD-V2.

Parameters	**Urb3DCD-V2**-1 LiDAR low dens.	**Urb3DCD-V2**-2 MS		**Urb3DCD-Cls**
	Both PCs	PC 1	PC 2	Both PCs
Density (points/m^2)	0.5	0.5	10	10
Noise range across track (°)	0.01	0.2	0.01	0.01
Noise range along track (°)	0	0.2	0	0
Noise scan direction (m)	0.05	1	0.05	0.05
Scan angle (°)	−20 to 20	−20 to 20		−20 to 20
Overlapping (%)	10	10		10
Height of flight (m)	700	700		700
Annotation level	Point	Point		PC

Table 1.5: **Acquisition configurations for the three sub-datasets of Urb3DCD-V2 and Urb3DCD-Cls.** Dens. stands for density.



(**a**) PC 1 (**b**) PC 2



(**c**) Labeled changes on PC 2

Figure 1.15: **Sample of Urb3DCD-V2 PCs illustrating examples of occlusions** at two timestamps (**a**,**b**) with the corresponding 7 types of changes simulated in (**c**).

Labels	No change	New building	Demolition	New vege.	Missing vege.	Total
Training set	2,395 (51.77%)	865 (18.70%)	660 (14.27%)	321 (6.94%)	385 (8.32%)	4,626
Val. set	412 (51.12%)	173 (21.46%)	158 (19.60%)	19 (2.36%)	44 (5.46%)	806
Testing set	1,233 (51.10%)	554 (22.96%)	329 (13.63%)	160 (6.63%)	137 (5.68%)	2,413

Table 1.6: **Class distribution for the Urb3DCD-Cls training, validation and testing splits.** Val. stands for validation; Vege. stands for vegetation. For each class, the number of samples along with the class proportion (in %) is given.

1.3.3 Urb3DCD-Cls

A third sub-dataset version has been created from the simulated data. The aim here is to propose pairs of PCs focused on mainly one type of change. The annotation is given as a function of the majority change in the pair of PCs, thereby this sub-dataset allows us to focus on the multiple change classification task (see Introduction, Figure 9b). This classification version of the dataset is inspired by Change3D dataset (see Section 1.1) (Ku et al., 2021). To build this dataset, pairs of cylinders of 15 m in radius have been extracted in simulated acquisitions over the training, validation and testing areas presented in Figure 1.13. The acquisition configuration of the PCs is given in Table 1.3.2 in Urb3DCD-Cls column. A label is given to the pair of cylinders as a function of the majority class. Pairs of cylinders where too many different classes were present are excluded from this sub-dataset. Finally, pairs of cylinders are distributed into five different classes: no change, new building, demolition, new vegetation and missing vegetation. The class distribution of this dataset is given in Table 1.6.

1.4 Conclusion

In this chapter, an **overview of existing multi-temporal 3D datasets** has been proposed along with an analysis of their properties. We showed in Section 1.1 that existing public datasets are not directly suitable for our purpose. Indeed, to tackle 3D PCs change segmentation, a dataset should contain multi-temporal PCs as well as corresponding annotation at point level. However, we saw that **most of existing datasets do not contain any labels or only contain single-date semantic annotation**. The single-date semantic annotation can be useful for multi-temporal semantic segmentation of 3D PCs, but cannot be directly used for change detection. Furthermore, existing datasets with change related annotation only provide the labelisation at the scene level. Therefore, **we**

proposed to create several datasets to tackle the 3D PC change segmentation task.

To this end, we first suggested in Section 1.2 a **semi-automatic process to adapt existing 3D multi-temporal datasets with single-date semantic annotation** to the change detection task. Thanks to this workflow, we created **AHN-CD dataset** derived from AHN ALS dataset. By investigating the AHN-CD dataset, we showed that it still contains some misclassifications. Although the semi-automatic process allows to rapidly and easily convert some 3D multi-temporal datasets with single-date semantic annotation to change detection purpose, the annotation coming out of this process is not perfect.

Thereby, we created an original **simulator of multi-temporal urban models**, where **3D PC acquisitions** are simulated by **mimicking airborne LiDAR surveying**. As presented in Section 1.3, this simulator allowed us to acquire various pairs of annotated PCs to further train supervised methods. Moreover, we were able to generate five different sub-datasets with **various acquisition configurations** illustrating the **variability in terms of sensor quality and type** as faced in real 3D PC acquisition. For one of the sub-datasets, we also considered **several training set sizes** to assess the robustness of the methods with respect to several training configurations. These seven sub-datasets form **Urb3DCD-V1 dataset**. A second version of the simulator has been developed to **enhance realism of PCs** by adding vegetation and mobile objects to simulated scenes. Thanks to the second version of the simulator, **Urb3DCD-V2 dataset** was created with two sub-datasets at different acquisition conditions. Finally, to mimic Change3D existing dataset, **Urb3DCD-Cls** version has been created. The latter provides bi-temporal PCs with **change annotation at the scene level**.

The datasets designed in this chapter are publicly available [10] to support reproducible research and to foster further works on 3D change detection, especially with deep learning methods.

In the next chapter, these datasets will be used to experimentally assess existing methods for 3D PC change detection.

10. Designed datasets are available at the following link: `https://ieee-dataport.org/open-access/urb3dcd-urban-point-clouds-simulated-dataset-3d-change-detection`.

Chapter 2

State-of-the-art in urban 3D change detection

Contents

This chapter attempts to understand how 3D change detection is performed in the literature, especially within the scope of urban environment. Notice that outside this application, other methods do exist but are mostly DSM-based (for computational reason or tailored for specific applications) implying, therefore, a limited use (Okyay et al., 2019).

While several methods for change detection in 3D data have already been published (see Section 2.1), this line of research is still new, and there is no evaluation of the existing approaches on a common dataset, making difficult the choice of a specific technique. Indeed, available review papers (Qin et al., 2016; Okyay et al., 2019; Shirowzhan et al., 2019; Kharroubi et al., 2022) do not provide quantitative comparison of methods on a common public dataset but rather introduce or qualitatively analyze existing approaches.

Thus, Section 2.2 aims to propose a comparison[11] between methods from the state-of-the-art at different levels: 2D patches, 2D pixels and 3D points. Compared methods are chosen to be representative of the different types of methods (distance-based, machine learning on handcrafted features-based and deep learning-based) and depending on the ease of use. Furthermore, analyzed methods solely rely on 3D data. Thanks to Urb3DCD-V1 sub-datasets (described in Chapter 1, Section 1.3.2), an evaluation of methods is performed on the different sub-datasets with various qualities. Furthermore, we evaluate the robustness of supervised methods on training sets of various sizes and assess their transfer learning capacities.

Section 2.3 gives a discussion about compared methods with regard to our results; we also propose a critical analysis of the Urb3DCD-V1 dataset and the proposed simulator.

2.1 3D change detection methods in urban environment

In the following section, the related works concerning 3D change detection methods in urban environment are given. When dealing with 3D data, methods can be divided into two categories: the ones based on 2D rasterization in DSMs and others that directly process raw 3D PCs.

11. This benchmark has been presented at IGARSS 2021 (de Gélis et al., 2021c) and further detailed in a paper published in Remote Sensing in 2021 (de Gélis et al., 2021b).

2.1.1 Digital Surface Model-based methods

Within this family, the principle for detecting changes between two 3D PCs is to compute DSMs for both of them, and to directly subtract them to retrieve differences. It was used for the first time for building change extraction in Murakami et al. (1999). Due to its simplicity and its quality of results, it is still often used. This approach is also commonly used in the earth observation community (Okyay et al., 2019).

DSMs difference (DSMd) for building change detection can be further refined with a non-empirical choice of the threshold thanks to the histogram of obtained values (Vu et al., 2004). One can also use the Otsu thresholding algorithm to segment resulting differences. This algorithm extracts thresholds from a histogram of values by minimizing the variance between each class (Otsu, 1979). As DSMs contain artifacts (due to interpolation in occlusions or the difficulty of retrieving precise building boundaries, for example) (Gharibbafghi et al., 2019), several methods apply more sophisticated pipelines to derive more accurate and finer changes than positive or negative ones. For example, Choi et al. (2009) used DSMd to identify change areas; then each change area was segmented via filtering and grouping and finally, thanks to specific indicators such as roughness, size and height, each segmented area was classified into three clusters: ground, vegetation and building. Comparing clusters at each date among them allows one to characterize changes. The selection of 3D building changes can also be done with regard to the size, height and shape of remaining clusters of pixels after an empirical thresholding of DSMd results (Dini et al., 2012). After applying a threshold to DSMd, Stal et al. (2013) removed non-building objects by applying morphological filters (erosion and dilation) and a threshold of roughness in order to get rid of vegetation. From 3D point clouds, one can rely on ground points to extract DSMs and DTMs. Teo and Shih (2013) used DSMd and a DTM to retrieve and classify each object at each date. Segmented objects could then be compared between the two periods to identify changes. Still based on DSMd, Pang et al. (2014) extracted building change candidates with a threshold on DSMd. After a connected component analysis, they used a Random Sample Consensus (RANSAC) to extract roofs and verify whether the objects were buildings. Finally, by comparing the heights of buildings, the authors classified them into four categories: 'newly built', 'taller', 'demolished' and ' shorter' buildings. Other approaches use statistical region merging segmentation of DSMd and compute shape similarity attributes to derive changes (Lyu et al., 2020) and finally highlight building changes through a k-means algorithm. One can notice that some studies use both information from DSMd and optical images in order to combine

the advantages of both types of data (Peng and Zhang, 2016; Wang and Li, 2020; Warth et al., 2019). Finally, DSMd with a basic thresholding or with further refinement is still widely used in the literature on change detection for 3D urban monitoring (Warth et al., 2019; Jang et al., 2019; Amini Amirkolaee and Arefi, 2019) and post-disaster building damage assessments (Erdogan and Yilmaz, 2019; Wang and Li, 2020).

With the rise of deep learning methods in earth observation, change detection in urban areas also benefits from these progresses using 2D images. This point will be discussed in Chapter 3, Section 3.1.2. By focusing on the use of DSMs data, Zhang et al. (2018b) and Zhang et al. (2019) have explored different architectures of Convolutional Neural Network (CNN) to detect 3D changes in urban areas. Since the objective of Zhang et al. (2018b) and Zhang et al. (2019) was to use bi-temporal multi-modal 3D information from both ALS and photogrammetric PCs, they chose a Siamese architecture (this will be detailed in Chapter 3, Section 3.1.2) in order to feed into one branch DSMs (directly the DSMd or into two channels) and in the other branch the corresponding RGB orthoimages. In their study, they also compute changed areas only with DSM information. In Zhang et al. (2019), the authors also tried a more basic feed forward (FF) network, where both DSMs were given as a single input with several channels, including both dates. They achieved reliable results with a precision of 63% for both FF and the Siamese network on DSM only, whereas DSMd only reached 38% of precision. Notice that DSMd was used per patch; i.e., an average of DSMd was made for the whole patch, with which the distinction between changed and unchanged patches was generated using a threshold.

2.1.2 Point Cloud-based methods

Another family of 3D change detection approaches directly rely on raw PCs, but has not been directly applied on urban areas yet. First, Girardeau-Montaut et al. (2005) proposed a cloud-to-cloud (C2C) comparison based on the Hausdorff point-to-point distance and on an octree sub-division of PCs for a faster computation. Lague et al. (2013) developed a more refined method for measuring mean surface change along a normal direction. Surface normal and orientation are extracted at a consistent scale according to local surface roughness. This method is called Multi-Scale Model-to-Model Cloud Comparison (M3C2). This second technique allows the distinction between positive and negative changes, which is not possible with the C2C approach (Shirowzhan et al., 2019).

Though previously presented methods use a distance computation, other kinds of approaches directly extract information from each PC, then segment PCs and finally

compare the results to obtain changes. These methods belong to the *post-classification* change detection retrieval family, because they are first based on a semantic segmentation of data and then a comparison of labeled data. There are few studies falling in this category, especially for an urban environment. Among them, Awrangjeb et al. (2015) first extracted buildings' 2D footprints with boundary extraction from LiDAR data and aerial images. Then, footprints were compared to highlight changes on a 2D map. In the same spirit of building extraction and comparison, and inspired by Awrangjeb et al. (2015), Siddiqui and Awrangjeb (2017) kept 3D information of buildings to retrieve changes. More precisely, 3D buildings' roof planes were retrieved at each date and then 3D buildings' models were cross-correlated by using their sizes and heights in order to classify them into categories of changes. The study in Xu et al. (2015b) also suggested segmenting each PC in order to extract buildings. Then, a 3D surface difference map should be created by computing a point-to-plane distance between a point in the first set and the nearest plane in the second set. A classification should finally be performed to identify various kind of changes (new dormer, addition of a floor, etc.). Dai et al. (2020) also tried to retrieve and classify changes by relying on object extraction from the second PC through the use of: i) a surface-based segmentation of filtered non-ground points, ii) segment-based screening to extract roofs and iii) connected component analysis to extract vegetation clusters. Then, a comparison of heights of roof segments extracted from both PCs was performed to highlight and classify changes. The final results were contained in a 2D map of changes. PC classification can also be done by traditional machine learning methods, such as the Random Forest (RF) algorithm, which is largely used in remote sensing. As an example, after extracting ground points and applying a region growing algorithm to retrieve each separated object, Roynard et al. (2016) classified each remaining object through a RF algorithm trained on several geometric and histogram-based features.

In opposition to post-classification methods, *pre-classification* methods first highlight changes and then characterize them. As an example in an urban environment, Xu et al. (2015a) first established an octree from one of the two PCs, and then directly extracted changes in the other PC by identifying corresponding missing leaf nodes. Clustering of changed points was performed to remove noise and separate the various changes. Finally, the remaining clusters were classified according to fixed rules concerning the area, height, or roughness. Recently, Huang et al. (2022b) also proposed a pre-classification framework with a first change detection step through occupancy-based spatial difference identification. They further refine their change identification, using semantics provided by a deep

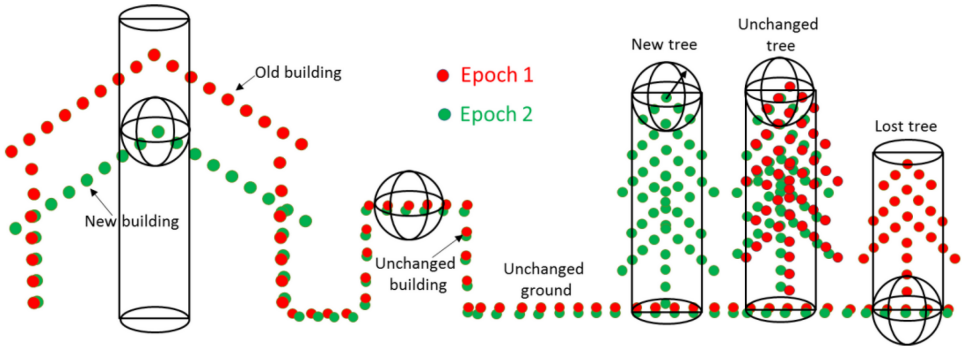


Figure 2.1: ***Stability* features of changed and unchanged objects.** Source: Figure from Tran et al. (2018)

network.

Finally, the technique in Tran et al. (2018) combines classification and change detection into a single step. To do so, the authors extracted features related to point distribution, terrain elevation, the multi-target capability of LiDAR and a feature (called stability) combining points of both PCs to detect changes. The *Stability* feature is defined as the ratio of the number of points in the spherical neighborhood to the number of points in the vertical cylindrical (oriented along the vertical axis) neighborhood in the other PC. Thus, in each point of the current PC, *Stability* is the ratio between the 3D (n_{3D}) and 2D (n_{2D}) neighborhood in the other PC:

$$Stability = \frac{n_{3D}}{n_{2D}} \times 100 \tag{2.1}$$

Notice that looking only at the number of points in the 3D neighborhood of each point of both PCs is enough to retrieve changes on isolated buildings and trees. However, in dense tree areas or when different objects are closed to each other, the 3D spherical neighborhood may still contain points coming from some other unchanged entity. Thus, taking the ratio with the 2D neighborhood is a way to take into account unchanged points and to obtain an indicator of change and the instability of the object. Thereby, the ratio will be near 100% if there is no change, and it tends to 0% if changes occur. On vegetation, we expect the *Stability* value to be lower. An illustration of the *Stability* feature is given in Figure 2.1. Then, an RF algorithm is trained on these features to obtain a supervised classification of changes into different classes. The single step, consisting in classifying directly multi-class changes, avoids propagating errors through the multiple stages.

While numerous works dealing with feature extraction, object detection, and segmen-

tation in 3D PCs are available, the change detection issue still remains largely unexplored with deep learning (Shi et al., 2020a). Apart from the works of Zhang et al. (2018b) and Zhang et al. (2019) using 2.5D DSMs, another deep architecture has been reported in Ku et al., 2021, namely Siamese Graph Convolutional Network (SiamGCN). This architecture is designed in the context of the SHREC21 track on Change3D dataset. We recall that this dataset was designed for multiple change classification in a complex street environment, i.e., it consists in recognizing the type of change between two PCs centered on an urban furniture, e.g., road signs (see Chapter 1, Section 1.1 for more details). Thus, the expected result is provided at the PC scale corresponding to the multiple change classification task (Figure 9b of the Introduction). Thereby, despite processing directly raw 3D PCs, this architecture does not correspond to the change segmentation task that we aim to tackle in this thesis.

2.2 Experimental comparison of state-of-the-art methods

Until now, five existing reviews/comparison papers dealing with change detection methods based on 3D data are available. The first one, Qin et al. (2016), provides a large review of the state-of-the-art on 3D change detection. Besides a description of available change detection methods for 3D data, it first draws up a list of applications; then different types of 3D data are described; and finally, it identifies the challenges of change detection retrieval in a 3D context. More recently, Okyay et al. (2019) provided an overview of airborne LiDAR change detection methods for Earth science applications. However, this survey does not tackle urban change detection and does not propose any comparison of existing methods on a common dataset. Then, Kharroubi et al. (2022) proposed a review of 3D change detection methods with a focus on deep learning[12]. However, this review does not provide any performance comparison. Very recently, Stilla and Xu (2023) also provided a survey on change detection in urban environment using 3D PCs. Although they carefully describe this problem, they do not provide any quantitative comparison of the methods as well. Finally, to the best of our knowledge, the study in Shirowzhan et al. (2019) is the only one that has proposed a comparative analysis for 3D change detection in an urban area. Five methods were compared regarding two criteria: the ability to

12. Notice that this recent review mostly presents methods that were not available when the benchmark presented in this chapter was conducted.

distinguish between demolished and new buildings, and giving information on the magnitudes of changes. However, no quantitative results are available, and the evaluation was performed on a private dataset. Moreover, this study did not include deep learning methods that today represent the state-of-the-art in remote sensing (Zhu et al., 2017; Ma et al., 2019).

Thus, we aim to propose a comparison between methods from the state-of-the-art considering PCs with various qualities and evaluate the robustness of supervised methods on training sets of various sizes while considering their transfer learning capacities.

2.2.1 Experimental protocol

We evaluated a set of methods relying both on extracted DSMs and PCs. The approaches have been selected as representatives of the current literature and are based on distance computation, machine learning with hand-crafted features or deep learning. As shown in the previous section, three levels of outputs are accessible: at the 3D point level, in 2D pixels or in 2D patches. Associated results can either be binary (changed/unchanged) or multi-class (unchanged, new construction or destruction).

First, we used DSMd (Murakami et al., 1999), C2C (Girardeau-Montaut et al., 2005) and M3C2 (Lague et al., 2013) with empirical thresholds, since these basic methods are easy to use and constitute a good baseline. For the DSMd method, we also present results obtained when choosing thresholds with the help of the popular Otsu algorithm (Otsu, 1979) and considering a morphological filtering (opening), as suggested in Stal et al. (2013), in order us to filter out isolated pixels and to clean up predictions. We also considered the method of Tran et al. (2018) as a representative of classical machine learning on hand-crafted features. As for hand-crafted features, we used the following ones related to:

- point distribution represented by point normals (N_x, N_y, N_z) and information on the distribution of points in the neighborhood (L_λ, P_λ, O_λ)
- height information (Z_{range}, Z_{rank}, nH)
- change information ($Stability$)

Information on the distribution of points contained in the neighborhood are given by the three variables: linearity L_λ, planarity P_λ and omnivariance O_λ. These variables represent respectively the likelihood of a point to belong to a linear (1D), planar (smooth

surface) (2D) or volumetric (3D) neighborhood. These three attributes are common to extract information into 3D PCs. They are computed from the three eigenvalues ($\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$) obtained after applying a Principal Component Analysis (PCA) to a matrix containing 3D coordinates of points contained in the neighborhood. Formulas of L_λ, P_λ and O_λ are given in Equations 2.2 to 2.4:

$$L_\lambda = \frac{\lambda_1 - \lambda_2}{\lambda_1} \tag{2.2}$$

$$P_\lambda = \frac{\lambda_2 - \lambda_3}{\lambda_1} \tag{2.3}$$

$$O_\lambda = \sqrt[3]{\lambda_1 \lambda_2 \lambda_3} \tag{2.4}$$

In practice, if λ_1 is large compared to λ_2 and λ_3, L_λ is near to 1. In this situation, only one eigenvalue is meaningful, i.e., only one principal axis results from the PCA and points are mainly distributed along a single axis. If λ_1 and λ_2 are large regarding λ_3, implying P_λ near to 1, points are spread in a plan defined by eigenvectors corresponding to λ_1 and λ_2. Lastly, O_λ is high if each of the three eigenvalues are of equal importance. This implies the points are scattered along the three axis, i.e., in a 3D volumetric space.

Z_{range} and Z_{rank} give information on the height by providing the maximum height (Z coordinate) difference between points in the neighborhood and the rank of the height of the considered points within the neighborhood. The normalized height nH also completes height information by providing the difference between the height of the considered points and the local DTM (rasterization of the PC at the ground level). Lastly, the *Stability* feature described in Section 2.1.2 (see also Figure 2.1 and Equation 2.1) is given to provide a bi-temporal information on the considered point. For all of these attributes, similarly to Tran et al. (2018), when a neighborhood is required, it is based on the k nearest points except for the *Stability* feature where neighborhoods are spherical and cylindrical based on fixed radius. Thereby, most of hand-crafted features presented in Tran et al. (2018) are used except those using LiDAR's multi-target capability, because our dataset does not contain such information.

Finally, even though deep learning methods are a little tougher to experiment with, since training and validation sets need to be constructed, their growing success in remote sensing make them unavoidable in any benchmark nowadays. Thus, we have implemented the FF CNN and the Siamese CNN architectures of Zhang et al. (2019) and Zhang et al. (2018b). Let us recall that each method has been described in Section 2.1.1 (for DSM-

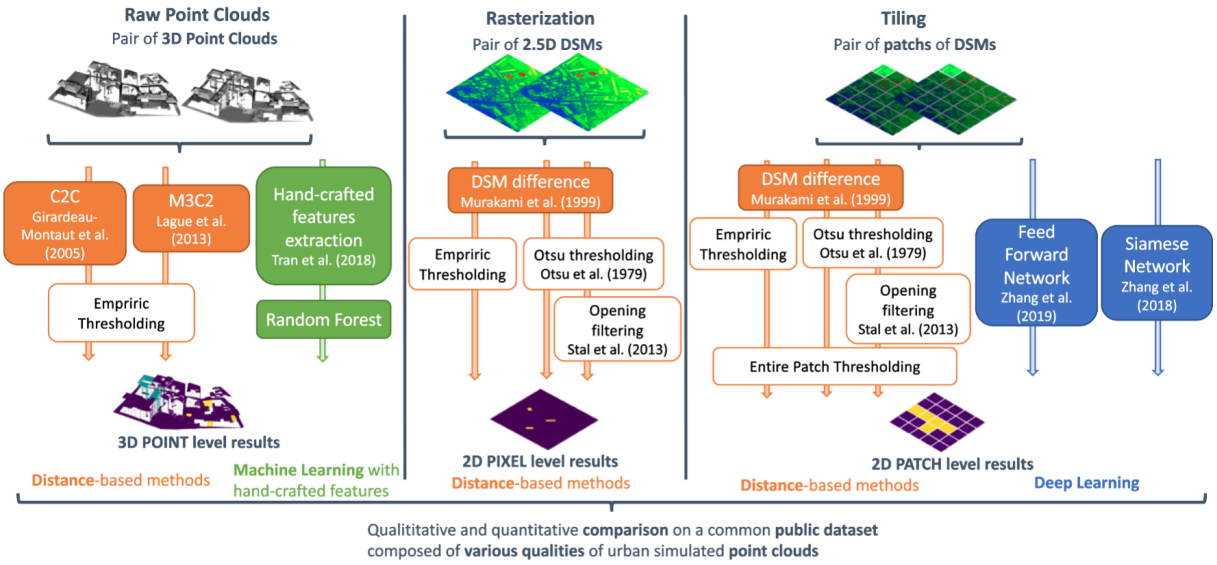


Figure 2.2: **A presentation of the different state-of-the-art methods for change detection and categorization compared in our benchmark**, considering three different levels of information: raw PCs, 2D rasterized PCs and patches of DSMs.

based ones) or Section 2.1.2 (for 3D PCs ones). We provide more technical details about methods and experimental settings in the next section. For the Siamese network, we used the architecture proposed in Zhang et al. (2018b) because this study was conducted only on DSMs, as in our study. In their publications, Zhang et al. (2018b) proposed only binary classification of change. However here, as our dataset contains information on the type of changes, we extended both networks to provide Multi-Class (MC) classifications of the urban changes. Figure 2.2 summarizes the methods from the state-of-the-art considered in our comparative study, organized according to their input data types and processing scales.

As far as the metrics are concerned, let us emphasize that the urban change detection problem usually faces the problem of large class imbalance (i.e., most of the 3D points or 2D pixels are unchanged). Thus, the usual overall accuracy or precision is not an appropriate for assessing the performance of a given method. Indeed, a method can reach up to 99% precision even if every pixel or point is predicted as unchanged. We instead prefer to rely on the IoU for each class as well as the mean of IoU (mIoU) over classes of

change (mIoU_{ch}). The IoU is given by the following equation:

$$IoU = \frac{TP}{TP + FP + FN} \tag{2.5}$$

where TP, TN, FP and FN respectively stand for True Positive, True Negative, False Positive and False Negative. According to the type of output, it was computed for each pixel, point or patch. In the case of binary classification, this corresponds to the IoU over positive class. Otherwise, for a multi-class scenario, this is the average between the IoU of the new construction class and the IoU of the demolition class.

All methods were tested on each sub-dataset of Urb3DCD-V1 presented in Chapter 1, Section 1.3.2. Thus, a first category of tests experimented the capabilities of different methods in various contexts: from high density with not much noise to low density very noisy data as input. Thereby, five tests depending on source, density of points and level of noise, see Section 1.3.2, were carried out for each of the six presented methods. Results are presented in Section 2.2.3.1. For machine and deep learning methods, we conducted other tests to evaluate the influence of the size of the training dataset thanks to sub-datasets 1.a, 1.b and 1.c presented in Table 1.4. Finally, we also aimed to study the behaviors of methods trained on a dataset with a different configuration than the testing set. Thus, for each sub-dataset, we trained machine and deep learning methods before testing them on the other sub-dataset without any re-training process.

Before sharing our results, the next section describes the experimental settings.

2.2.2 Experimental settings

C2C and M3C2 were performed with CloudCompare software (Girardeau-Montaut, 2016). We re-implemented feature extraction in Python with all features of Tran et al. (2018). DTM computations were performed using Point Data Abstraction Library (PDAL) library[13]. We observed that some features are very dependent on the neighboring size (radius or number of neighbors) that is chosen. As our datasets have different densities than the dataset used in the study of Tran et al. (2018), we tested several values of the radius and selected the best one according to each density. Note that the selection is made on the validation set based on the mean of IoU over classes of change. Thus, for all sub-datasets with the density of 0.5 points/m^2, the radius was fixed to 5 m. For the sub-dataset 2 with a density of 10 points/m^2, a radius of 3 m was selected. Notice that

13. `https://pdal.io/en/2.5-maintenance/index.html`, accessed on 27/02/2023.

with the multi-sensor sub-dataset 5, the best results were obtained with a radius of 4 m. When the neighborhood is based on k nearest points, k was set to 10. Training was realized on the training set only.

For deep methods, FF and Siamese network architectures were used similarly to the original papers (Zhang et al., 2019; Zhang et al., 2018b). In their study, Zhang et al. aimed to detect changes between one DSM made from ALS PC data and another one made from photogrammetric data. They presented a few tests with various inputs relying on DSMs only or also on DSMs, and color information coming from RGB orthoimages. Since our dataset only contains information related to the 3D coordinates of points, we kept only configurations where only height information was given. In the case of the FF network, DSMs were provided as inputs through two different channels. For the Siamese network, each DSM was fed into one branch. In our context, the two inputs were quite similar (both were DSMs and came from the same type of sensor); thus, we decided to share weights between the two Siamese branches and to consider a Euclidean distance computation to gather branches, similarly to Zhang et al. (2018b). When training the deep networks, weights were randomly initialized and all networks were trained from scratch. Batch size was set to 128. For the Siamese network, we used Stochastic Gradient Descent (SGD) with momentum as an optimizer with an initial learning rate of 0.003 and a momentum of 0.9. Unlike Zhang et al. (2019), we obtained better results with an Adam optimizer with a learning rate of 0.001 for the FF network, so we kept this configuration for the results presented further on. Notice that for both networks, a few tests have been conducted to select the optimal hyper-parameters, i.e., providing the best results on our datasets. Hyper-parameters selection is made on the validation set. The same configurations were kept for all sub-datasets. Implementation was done in Python with PyTorch.

For all methods relying on DSMs as inputs, we used a spatial resolution of 0.3 m. As comparisons are made between results obtained on the different sub-datasets of Urb3DCD-V1 presented in Chapter 1, Section 1.3.2, we decided to derive DSM resolution from the PC density of sub-dataset 2 (10 points/m^2). Thus, for all data of lower resolution, interpolation was performed when there were no corresponding points in a given cell (pixel). A coarser resolution could have been chosen, but this would have implied less precise results for 2D pixels and also degraded DSM resolution, which can be obtained with sub-dataset 2. Notice that we ran a test of the DSMd method at a coarser resolution (1.40 m) on sub-dataset 1, but this did not lead to better results.

Concerning thresholds, we chose $-3\,\text{m}$ and $3\,\text{m}$ for MC classification, corresponding to one little floor, since the dataset contains changes from building construction or demolition. The opening filter was using a $10\,\text{pixels} \times 10\,\text{pixels}$ kernel size. Empiric thresholds for C2C and M3C2 methods varied according to the dataset; we tried several values and systematically selected the best one for each sub-dataset.

In order to derive 2D ground truth from 3D PC labeling, we used a majority voting strategy. Each pixel was given the most frequent label within its corresponding points. When a label was required as the patch scale, we followed the same process as in Zhang et al. (2019), and marked a patch as changed if the ratio of changed pixels was greater than 10%. For multi-class patch labeling, when a patch was considered as changed, we selected the label according to the majority of pixels in each class of change.

Inspired by Zhang et al. (2019), patch selection was done with a half-overlap sampling. Classes were largely imbalanced; indeed, there were more unchanged areas than changed ones. This imbalance is also visible when looking at the number of patches in each class. We thus relied on data augmentation only for patches of change, as in Zhang et al. (2019). A similar kind of data augmentation was applied, i.e., vertical and horizontal flips, as well as rotations with angles of 90°, 180° and 270°. Finally, patches at the edge of the acquisition could contain some empty pixels, since all patches were square but the acquisition area was not. Thus, another threshold was fixed at 90% non-empty pixels to consider a patch as valid. Patch size was set to $100\,\text{pixels} \times 100\,\text{pixels}$, corresponding to $30\,\text{m} \times 30\,\text{m}$ on the ground.

2.2.3 Results

We now report the experimental results obtained using the methodology described in Sections 2.2.1 and 2.2.2. First, we assess the performances of the different methods on the various sub-datasets to evaluate their behavior in various acquisition conditions (e.g., density, noise, etc.). We then study the influence of the training set size and the ability of each method to cope with domain adaptation.

2.2.3.1 Experiments with various acquisition configurations

Let us recall that results come with different levels depending on the method. All methods dealing with DSMs furnished results in 2D. First, DSMd led to the per-2D-pixel results presented in Table 2.1. This table summarizes results for all sub-datasets with different

ways of selecting thresholds (via an empirical choice or Otsu algorithm, as mentioned in the Section 2.2.1). It also contains the results of the DSMd method with an opening operation and the Otsu algorithm. For each sub-dataset and type of classification (Binary or MC), bold results indicate the best performing methods for this scenario. Following these quantitative results, Figures 2.3–2.7 give a zoomed view of the testing set for each of the five sub-datasets. These figures contain DSMs extracted from the original PCs, the 2D corresponding ground truth and the results for the three versions of DSMd. We also provide each result with an error map (g, h, i). One can notice the difference in terms of quality of DSMs among different sub-datasets. In particular, Figures 2.4(a, b) and 2.7b correspond to DSMs of high resolution with low-noise PCs. DSMs of Figure 2.3(a, b) corresponding to low density but not noisy ALS data (sub-dataset 1.b) are a slightly less precise. Finally, Figures 2.5(a, b), 2.6(a, b) and 2.7a were extracted from low density and noisy PCs, and thereby seem more inaccurate and blurry.

While the initial results with empirical thresholding are not convincing, we can observe a significant improvement when using the Otsu algorithm to select thresholds (values of about $-9\,\mathrm{m}$ and $9\,\mathrm{m}$). The results were further improved when an opening filter was added to remove isolated pixels that do not correspond to real changes. Visual assessment confirmed the utility of the opening filter (f, i) compared to the initial results with empirically-set thresholds (d, g) or using the Otsu algorithm (e, h), and its ability to remove isolated pixels (Figures 2.3–2.7). Unsurprisingly, the best results were achieved on the second sub-dataset corresponding to pairs of PCs with high density and a low level of noise. Notice that the gap in quality between DSMd with a filtering operation and DSMd with the Otsu algorithm was higher for sub-datasets containing noisy data (sub-datasets 3, 4 and 5). Indeed, Table 2.1 reports a gain of about 12 points of mean of IoU between DSMd+Otsu and DSMd+Otsu+Opening methods for sub-datasets 1.b and 2 with low levels of noise, whereas the opening filter increased this metric by 17–22 points for datasets that contain noisy PCs. Indeed, noisy data led to numerous false detection instances in isolated parts of the DSM, which could be eliminated through this morphological operation. This is visible in Figures 2.5(d, g) and 2.6(d, g), where isolated pixels marked as new construction or demolition can be seen. In addition to these isolated pixels, almost all building edges were highlighted with empirical thresholds, especially because of the noise on vertical facades that particularly interferes with the extraction of the DSM. Indeed, such noise makes the facade not straight and difficult to distinguish precisely from an above point of view, as highlighted in sub-datasets 3, 4 and

5 (Figures 2.5–2.7). Notice that with Otsu thresholding, it remained visible (at a lower level though).

When looking at Table 2.1, one can notice that even if the worst results were obtained for sub-datasets 3 and 5 without the filtering operation, the scores became comparable with sub-dataset 1.b when adding the filtering step. Thus, even in cases of noisy data (for one or both PCs), we could achieve similar performance as with a low-noise setup thanks to the morphological operation. However, results in the multi-sensor scenario (sub-dataset 5) remained lower than with sub-dataset 2, even though one of the PCs of sub-dataset 5 has the same quality as sub-dataset 2. The visual assessment of the effects of the opening operation on sub-datasets 3 and 5 (Figures 2.5(f, i) and 2.7(f, i)) shows quite similar results. True changes are visible, but there are still quite a few instances of false change detections.

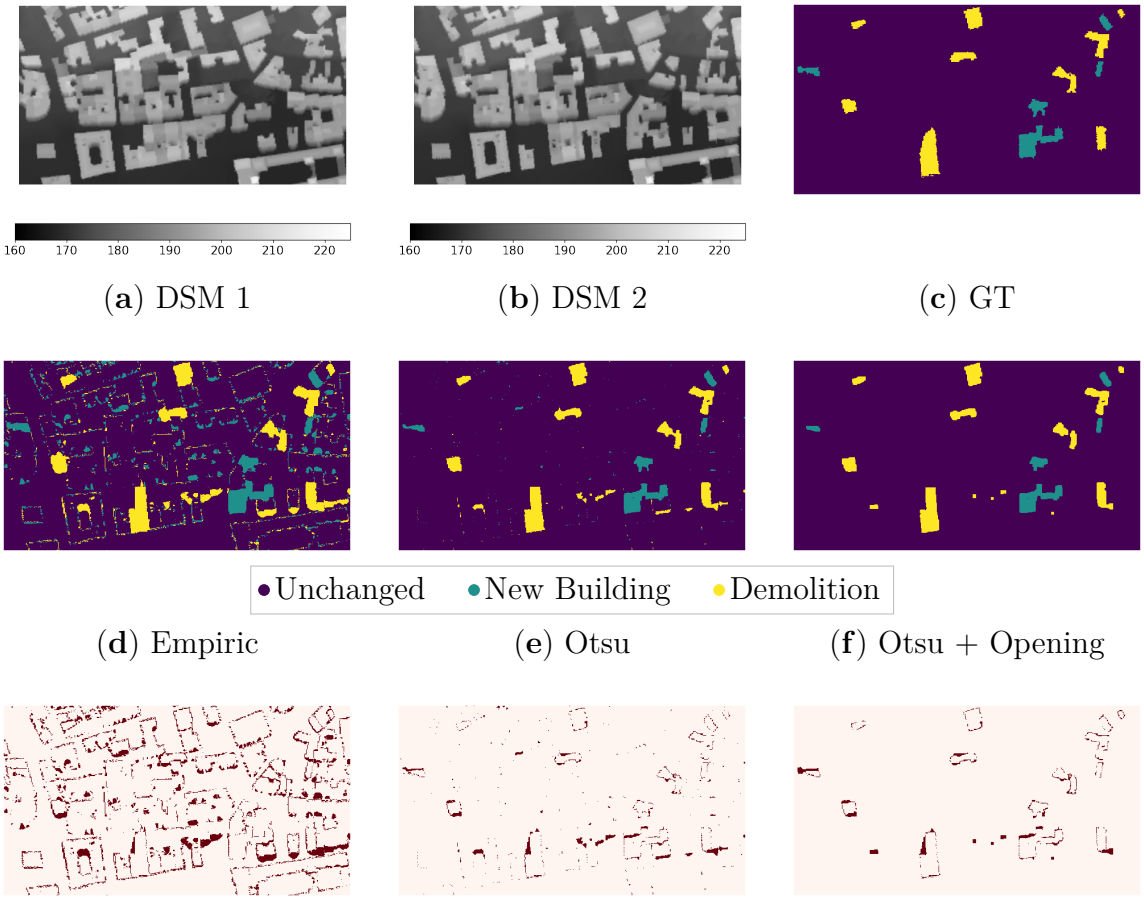
Surprisingly, the fourth sub-dataset obtained a high mean of IoU on classes of change according to Table 2.1. This is particularly visible for DSMd with the Otsu algorithm and the opening filter. This sub-dataset is composed of a pair of PCs with low density ($0.5\,\text{points/m}^2$) and high noise, as the sub-dataset 3. However, the range of scanning angles during acquisition and the overlap between flight tracks were both halved (see Table 1.4). Using this configuration, we aimed to mimic photogrammetric PCs acquired thanks to satellite images. These images have a top-down point of view, leading to fewer shadows. Indeed, the higher the scanning angle and the higher the building, the more shadows in the PCs. In the context of DSM extraction, a building shadow implies empty pixels. In this study, we filled them through interpolation, but this may lead to errors or at least a lack of precision in these areas. Moreover, the flight plan was not the same for the acquisitions of the pair of PCs; hence, shadows are not the same in each PC. Thus, high difference values can be obtained even in unchanged areas. For a qualitative illustration, refer to Figure 2.4(d, e), where a band of pixels marked as changed is always visible on the same side of every building. Indeed, we can also see the difference in DSMs: the north sides of buildings seem more blurry than the other sides in DSM 1 (Figure 2.4a), due to a lack of information in the area just near the facade because of building shadows. In comparison, as the flight plan was not the same for DSM 2 (Figure 2.4b), building edges are more distinct at the bottom of the image. Some misclassifications could be corrected by the opening filter, but only up to a certain extent (i.e., not the larger ones). Finally, sub-dataset 4 contains fewer building shadows, leading to easier identification of true changes. This explains the higher results achieved on this photogrammetric, look-alike

sub-dataset 4 and reported in Table 2.1. Note that the edges of the buildings have also been highlighted in the figure 2.6(d, e) because of the noisiness of the photogrammetric, look-alike sub-dataset 4. This differs from sub-dataset 3, where both effects are visible on buildings' edges: some edges are continuous in one class of change corresponding to building shadows, and predictions of other edges are more heterogeneous. Notice that heterogeneous parts are easily cleaned with the opening operation, leading to a significant difference of results between sub-datasets 3 and 4 in Table 2.1.

Table 2.2 presents results at the 2D patch level for deep learning methods. When comparing both FF and Siamese networks, we can see that better results were obtained with the FF architecture. The same trend was reported in Zhang et al. (2019) when only the DSM information was given to the networks. However, in these original works (Zhang et al., 2018b; Zhang et al., 2019), results were only binary. We propose here to extend these approaches to deal with MC classification. For FF architectures, the quality of MC results is close to the binary case, with only 3–5 points less, depending on the sub-dataset. On the contrary, the Siamese architecture leads to a higher gap, with a 7–10% loss between binary and MC classification. Thereby, Siamese architecture seems less efficient in this context. Better results were again obtained for the high density sub-dataset 2; they were rather similar for sub-datasets 1.b, 3 and 5. Here again, identification of true changed patches seemed to be easier for sub-dataset 4, which showed better results than sub-datasets 1.b, 3 and 5.

For the sake of comparison, we have adapted at the patch level the DSMd method and its variant, i.e., considering Otsu thresholding and morphological filtering. To do so, we have followed the patch-wise strategy introduced in previous works and reported in Section 2.2.2. Namely, we divided with half-overlap sampling the different DSMd results into patches in order to extract the same patches as those that are extracted for deep learning methods. To retrieve labels at patch level, we set a threshold on the percentage of changed pixels according to each DSMd result. For MC classification, when a patch was identified as changed, we have the label of the most prominent class of change. Thresholds of validity and changed pixels were set based on the setup used with deep learning methods (90% and 10%, respectively). Results are given in Table 2.2. We can see that DSMd with the Otsu algorithm and opening filter overtook deep methods for all sub-datasets. Let us note that the coarseness of the analysis (patch instead of pixel) hinders the interest of a visual assessment.

Finally, Table 2.3 gives results at the 3D point level. Notice that the C2C method

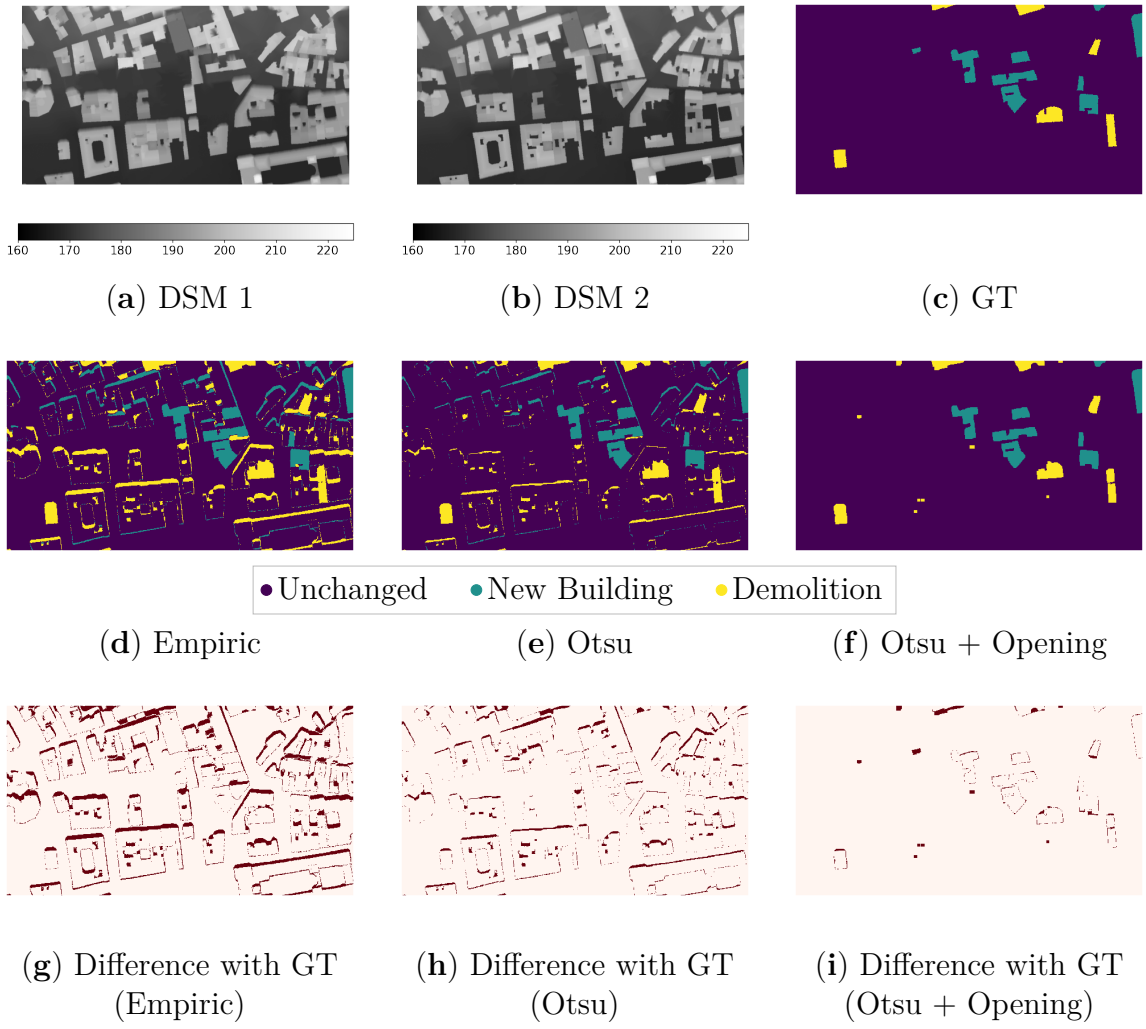


(**a**) DSM 1 (**b**) DSM 2 (**c**) GT

(**d**) Empiric (**e**) Otsu (**f**) Otsu + Opening

(**g**) Difference with GT (Empiric) (**h**) Difference with GT (Otsu) (**i**) Difference with GT (Otsu + Opening)

Figure 2.3: **Results at the 2D pixel level for sub-dataset 1.b (ALS, low density):** (**a**,**b**) DSMs with their elevations in meters and (**c**) ground truth (GT). Results of DSMd with empirical thresholding (**d**), thresholds from the Otsu algorithm (**e**) and post-processing with an opening filter (**f**). Respective errors maps (**g**–**i**) highlight the differences (in red) from the ground truth.



(**a**) DSM 1 (**b**) DSM 2 (**c**) GT

(**d**) Empiric (**e**) Otsu (**f**) Otsu + Opening

(**g**) Difference with GT (Empiric) (**h**) Difference with GT (Otsu) (**i**) Difference with GT (Otsu + Opening)

Figure 2.4: **Results at the 2D pixel level for sub-dataset 2 (ALS High density):** (**a**,**b**) DSMs with their elevations in meters and (**c**) ground truth (GT). Results of DSMd with empirical thresholding (**d**), thresholds from the Otsu algorithm (**e**) and post-processing with an opening filter (**f**). Respective errors maps (**g**–**i**) highlight the differences (in red) from the ground truth.

provides unsigned results, which makes impossible the distinction between positive and negative changes; thus, only binary results are provided. Here again, one method surpassed the others for each kind of sub-dataset. Indeed, the machine learning method of Tran et al. (2018), using the RF algorithm trained on hand-crafted features with a *Stability* feature, had more precise results. Notice that M3C2 did not seem to have interesting results in the context of quite low density PCs: the means of IoU for classes of change were higher than 50% only for the sub-dataset with the higher density. As far as binary results are concerned, C2C seems more suitable if no labeled training set is available. The RF algorithm with hand-crafted features has quite similar results for binary and MC classification. Let us emphasize that Tran et al. (2018) designed their method and especially the feature extraction for a MC classification scenario.

For the sake of qualitative assessment at the 3D point level, Figures 2.8 and 2.9 provide PCs for both dates; the associated ground truth according to the change; and the results provided by C2C, M3C2 and the RF method with hand-crafted features. Firstly, one should outline the difference of quality between all sub-datasets. Indeed, Figure 2.8(a,b) presents low-noise PCs at low density, (a) corresponding to the first sub-dataset, and high density, (b) corresponding to the second sub-dataset. Figure 2.9(a–c) shows PCs of noisy sub-datasets. In Figure 2.9c, the difference between the noisy, low-density PC and the precise, high-density PC is substantial.

From a quantitative point of view, the same tendency is visible in Table 2.3. C2C binary results do not seem so far from ground truth in changed areas; however, it led to some false positive predictions on building facades. M3C2 led to even more false positives, but also it did not highlight changes in some areas. This is particularly visible in Figure 2.8b, where a lot of points lying on unchanged facades are identified as changed, conversely to points on the facade of the new building. Moreover, in the same example, some points were even identified as destruction instead of construction. For all sub-datasets, the RF method (Tran et al., 2018) seemed to bring quite accurate results, except for the ground points near buildings; those points were identified as deconstructed. This is visible in Figures 2.8a and 2.9(a,c), corresponding to the building shadow effects already mentioned. When looking directly at the PCs of both dates, it is particularly visible in Figure 2.8a that all points assigned to a deconstruction label at ground level which are not true deconstruction, are points that do not exist in the first PC (PC 1).

Notice that no confusion between the two classes of change seems to have arisen in the RF method. More specifically, this method seems to have learned that deconstruction

Method		Sub-Datasets 1 ALS Low Dens. Bin.	1 ALS Low Dens. MC	2 ALS High Dens. Bin.	2 ALS High Dens. MC	3 ALS High Noise Bin.	3 ALS High Noise MC	4 Photogrammetry Bin.	4 Photogrammetry MC	5 Multi-Sensor Bin.	5 Multi-Sensor MC
DSMd + Empiric thresholding	Murakami et al. (1999)	37.11	36.89	50.93	50.72	30.34	29.97	30.85	30.73	33.65	33.16
DSMd + Otsu	Murakami et al. (1999)	59.36	59.14	67.08	66.96	53.15	52.67	54.81	54.77	53.15	52.61
DSMd + Otsu + Opening	Murakami et al. (1999)	**72.21**	**71.92**	**79.28**	**79.09**	**70.67**	**70.34**	**76.92**	**76.89**	**71.89**	**70.99**

Table 2.1: **Results of different methods on all sub-datasets at 2D pixel level.** Means of IoU over classes of change are given (%). Bin. stands for binary classification results; MC for multi-class classification results; Dens. for density. For each type of results, the best values are shown in bold.

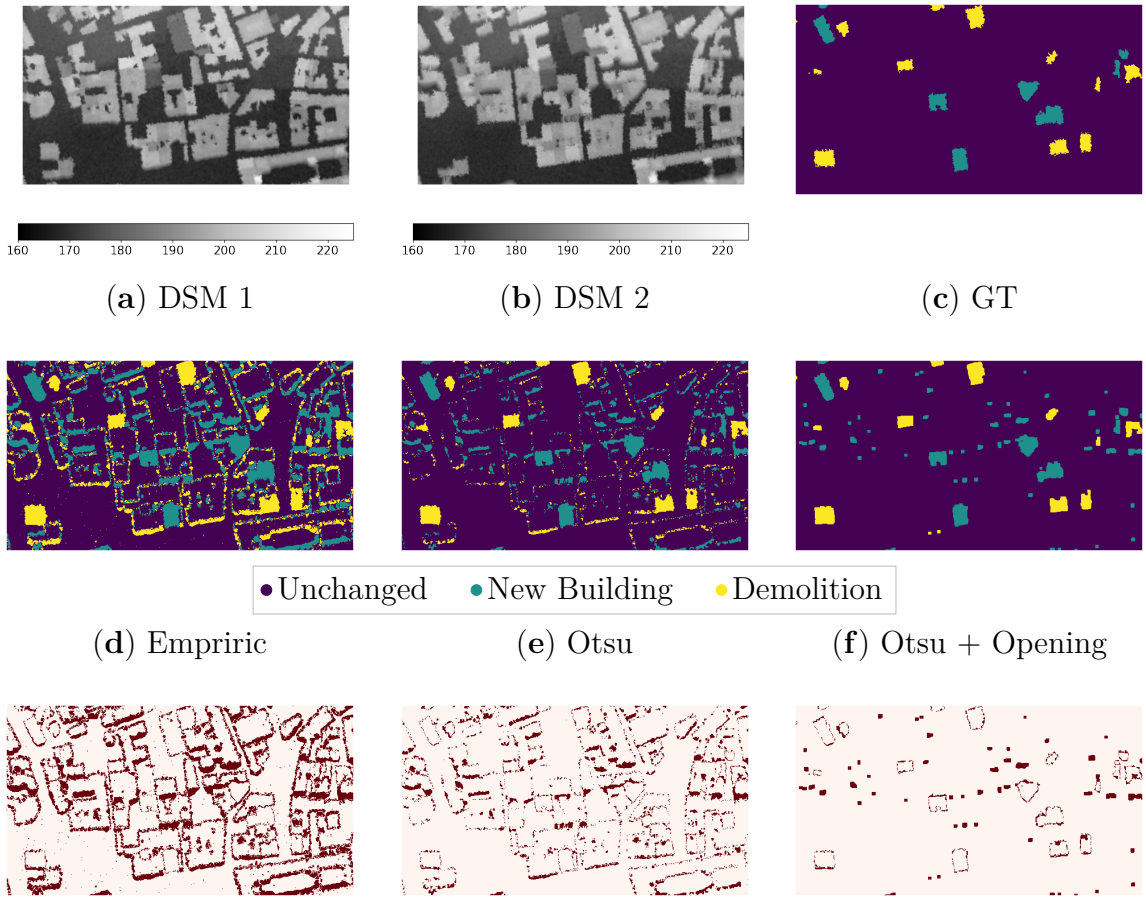
Method		Sub-Datasets 1 ALS Low Dens. Bin.	1 ALS Low Dens. MC	2 ALS High Dens. Bin.	2 ALS High Dens. MC	3 ALS High Noise Bin.	3 ALS High Noise MC	4 Photogrammetry Bin.	4 Photogrammetry MC	5 Multi-Sensor Bin.	5 Multi-Sensor MC
DSMd + Empiric thresholding	Murakami et al. (1999)	36.13	34.59	50.06	48.78	30.93	29.90	31.11	30.76	32.79	31.86
DSMd + Otsu	Murakami et al. (1999)	62.44	60.40	76.82	75.23	58.46	56.59	72.04	71.00	56.60	55.02
DSMd + Otsu + Opening	Murakami et al. (1999)	**82.27**	**80.22**	**88.39**	**86.62**	**83.47**	**80.71**	**88.51**	**86.83**	**82.22**	**79.59**
FF Network	Zhang et al. (2019)	77.64	75.47	84.63	79.66	79.40	76.37	82.54	79.95	79.73	76.77
Siamese Network	Zhang et al. (2018b)	71.58	64.16	79.97	72.18	78.33	71.36	79.03	69.50	75.27	67.41

Table 2.2: **Results of different methods on all sub-datasets at the 2D patch level.** Means of IoU over classes of change are given (%). Bin. stands for binary classification results; MC for multi-class classification results and Dens. for density. For each type of results, the best values are shown in bold.

Method		Sub-Datasets 1 ALS Low Dens. Bin.	1 ALS Low Dens. MC	2 ALS High Dens. Bin.	2 ALS High Dens. MC	3 ALS High Noise Bin.	3 ALS High Noise MC	4 Photogrammetry Bin.	4 Photogrammetry MC	5 Multi-Sensor Bin.	5 Multi-Sensor MC
C2C	Girardeau-Montaut et al. (2005)	49.59	-	61.65	-	49.22	-	54.75	-	49.76	-
M3C2	Lague et al. (2013)	33.50	29.87	59.85	53.73	41.37	38.72	36.60	35.01	40.61	37.78
RF (with *Stability* feat.)	Tran et al. (2018)	**63.70**	**63.41**	**70.48**	**70.12**	**58.78**	**58.83**	**68.46**	**68.87**	**63.65**	**63.64**

Table 2.3: **Results of different methods on all sub-datasets at the 3D point level.** Means of IoU over classes of change are given (%). Bin. stands for binary classification results; MC for multi-class classification results and Dens. for density. For each type of results, the best values are shown in bold.

is always at ground level; this is not the case for the M3C2 method. Indeed, since only thresholding was applied, some deconstruction parts could appear on facades. Finally, point predictions on roofs were not so far from ground truth for all methods, though M3C2 results were the least accurate.

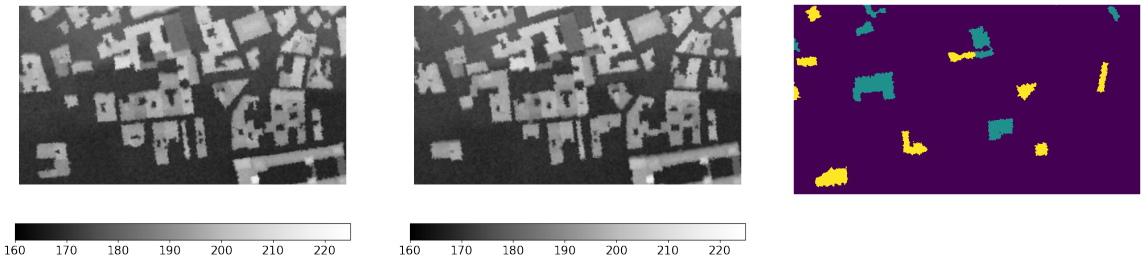


(**a**) DSM 1 (**b**) DSM 2 (**c**) GT

(**d**) Empriric (**e**) Otsu (**f**) Otsu + Opening

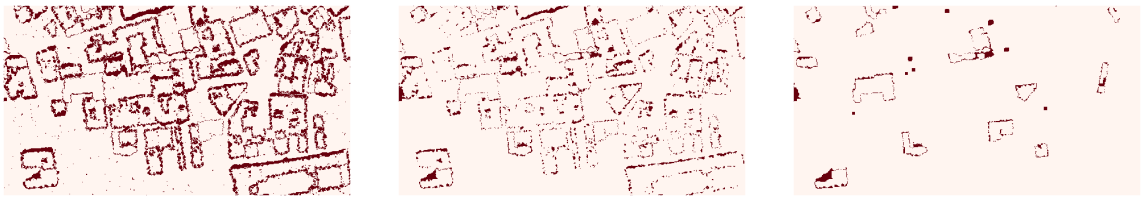
(**g**) Difference with GT (Empiric) (**h**) Difference with GT (Otsu) (**i**) Difference with GT (Otsu + Opening)

Figure 2.5: **Results at the 2D pixel level for sub-dataset 3 (ALS, low density, high noise):** (**a**,**b**) DSMs with their levels of elevation in meters and (**c**) ground truth (GT). Results of DSMd with empirical thresholding (**d**), thresholds from the Otsu algorithm (**e**) and post-processing with an opening filter (**f**). Respective errors maps (**g**–**i**) highlight the differences (in red) from the ground truth.



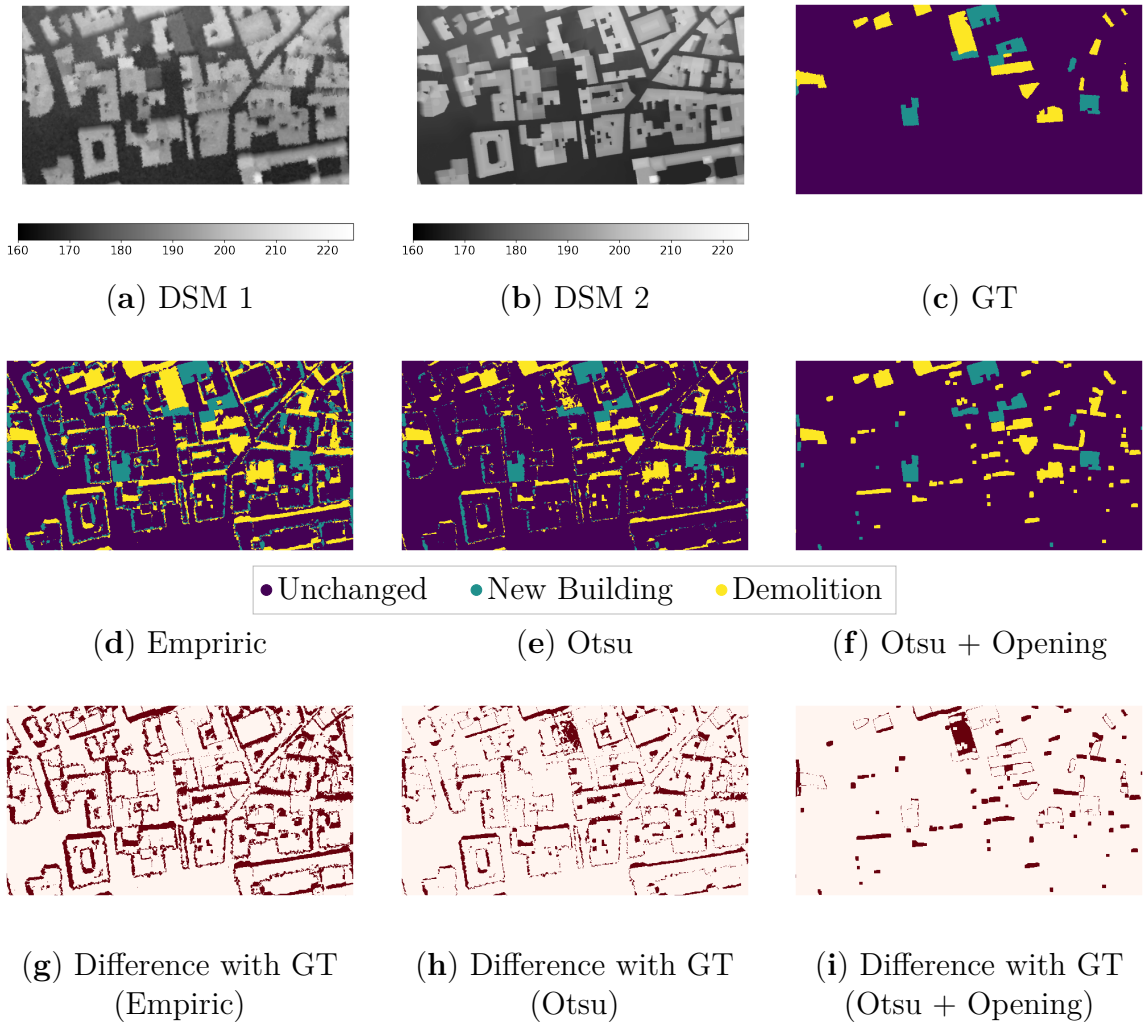
(**a**) DSM 1 (**b**) DSM 2 (**c**) GT

(**d**) Empriric (**e**) Otsu (**f**) Otsu + Opening



(**g**) Difference with GT (Empiric) (**h**) Difference with GT (Otsu) (**i**) Difference with GT (Otsu + Opening)

Figure 2.6: **Results at the 2D pixel level for sub-dataset 4 (photogrammetric):** (**a**,**b**) DSMs with their elevation levels in meters and (**c**) ground truth (GT). Results of DSMd with empirical thresholding (**d**), thresholds from the Otsu algorithm (**e**) and post-processing with an opening filter (**f**). Respective errors maps (**g**–**i**) highlight the differences (in red) from the ground truth.



(**a**) DSM 1 (**b**) DSM 2 (**c**) GT

(**d**) Empriric (**e**) Otsu (**f**) Otsu + Opening

(**g**) Difference with GT (Empiric) (**h**) Difference with GT (Otsu) (**i**) Difference with GT (Otsu + Opening)

Figure 2.7: **Results at the 2D pixel level for sub-dataset 5 (Multi-sensor):** (**a**,**b**) DSMs with their elevation levels in meters and (**c**) ground truth (GT). Results of DSMd with empirical thresholding (**d**), thresholds from the Otsu algorithm (**e**) and post-processing with an opening filter (**f**). Respective errors maps (**g**–**i**) highlight the differences (in red) from the ground truth.

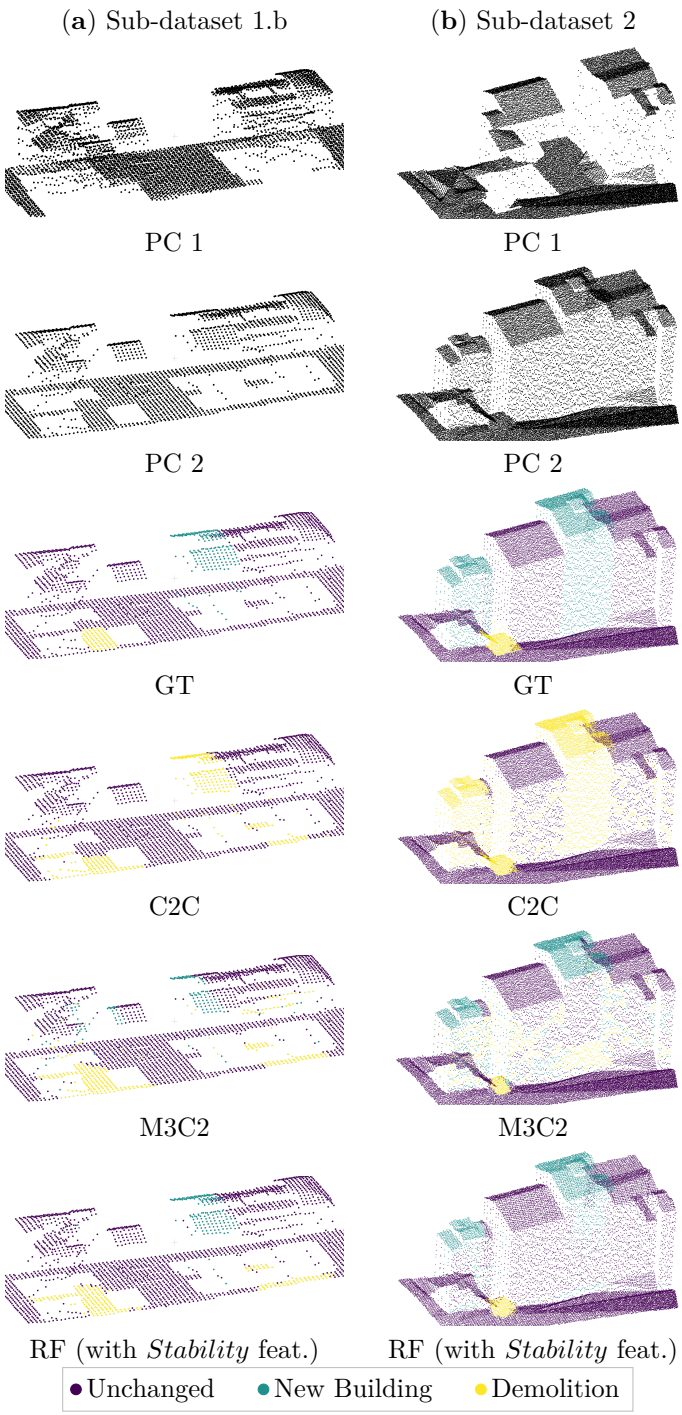


Figure 2.8: **Results on PCs for methods at 3D point level for sub-datasets 1.b (a) and 2 (b)**. Both PCs corresponding to both dates are given. The corresponding ground truth (GT) indicates changes in the second PC (PC 2) compared to the previous PC (PC 1). C2C results are only binary: purple corresponds to unchanged parts and yellow to changed areas.

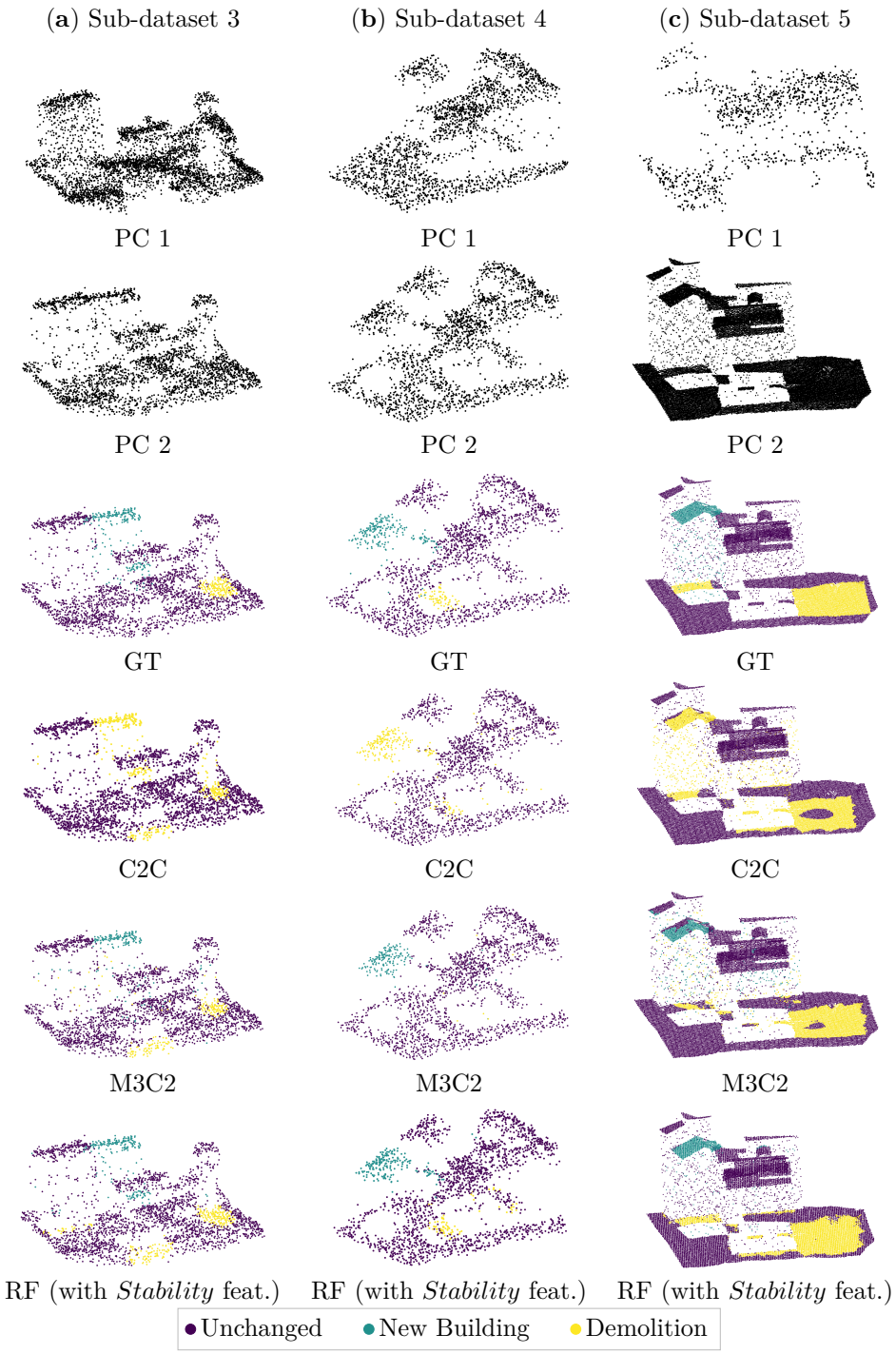


Figure 2.9: **Results on PCs for methods at the 3D point level for sub-datasets 3 (a), 4 (b) and 5 (c)**. Both PCs corresponding to both dates are given. The corresponding ground truth (GT) indicates changes in the second PC (PC 2) compared to the previous PC (PC 1). C2C results are only binary: purple corresponds to unchanged parts and yellow to changed areas.

2.2.3.2 Experiments with various training configurations

Having quantitatively and qualitatively assessed the performances of the different methods when applied to various acquisition conditions, as demonstrated in our pool of sub-datasets, we now focus on the generalization capacities of supervised methods. We especially study their behaviors when trained on datasets of various sizes or having different characteristics. Indeed, most often, no annotated data are available for the target areas, and/or such areas are not covered by the same sensors (or the same acquisition conditions) as the training areas. Thus, it is crucial to assess the ability of techniques trained on synthetic data to perform well on various datasets.

Results reported in Table 2.4 analyze the influence of the training set size. Indeed, the three sub-datasets were of similar type (ALS with low density and low noise); the difference lays in the size of the set by varying the number of pairs of simulated PCs (1, 10 and 50). Note that the validation set was the same for the three tests. Unsurprisingly, the best results were obtained with the largest training set (1.c), which is consistent with common observations using neural networks or RF. However, in the latter case using hand-crafted features, the improvement was limited compared with the case of a neural network: a bit less than 6% using RF between sub-datasets 1.a and 1.c, compared to 27% and 19% with FF method (binary and MC), and 20% and 11% with Siamese network (binary and MC), respectively. One can notice that for the smallest training set (1.a), the Siamese network led to better binary results than the FF network. Contrary to the RF method, the gap between sub-datasets 1.a and 1.b was larger than between sub-datasets 1.b and 1.c.

Finally, one can notice that DSMd with the Otsu thresholding and opening filtering was still a bit better than deep networks at the 2D patch level trained on the larger dataset (see Table 2.2).

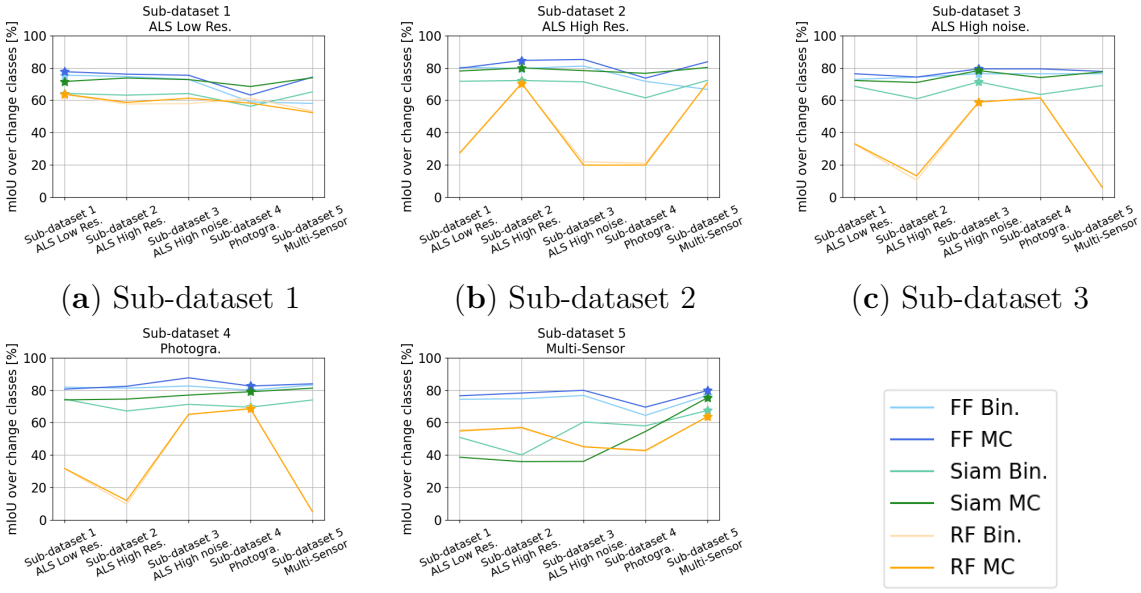
Figure 2.10 provides clues to assess the transfer capacity, through training (and validating) on a dataset with different characteristics than the testing set. Here, we consider each transfer learning scenario with various training configurations related to all five sub-datasets discussed above. Notice that the trained models were directly applied to the testing set, without re-training. In each line of Figure 2.10, the 'star' corresponds to the scenario where training and testing sets were similar, i.e., without transfer learning (and this can be seen as the reference). Let us recall that methods based on neural networks lead to results on patches, whereas RF methods lead to results on PCs. Thus, a direct comparison between deep learning and the RF methods is not possible. However, it is

Sub-Datasets			**Method at 3D Point Level** RF (with *Stability* Feat.) Tran et al. (2018)		**Methods at 2D Patch Level** FF Zhang et al. (2019)		Siamese Zhang et al. (2018b)	
			Bin.	**MC**	**Bin.**	**MC**	**Bin.**	**MC**
1	ALS Low Dens.	a	61.51	61.34	57.52	58.43	58.62	55.07
		b	63.70	63.41	77.64	75.47	71.58	64.16
		c	**67.28**	**67.10**	**81.98**	**77.28**	**78.85**	**66.58**

Table 2.4: **Results of different supervised methods with varying training set sizes** (training sets of sub-dataset 1.a, 1.b and 1.c contained, respectively, 1, 10 and 50 simulations). Means of IoU over classes of change are given (%) at point level for the RF method, and at patch level for both FF and Siamese networks. Bin. stands for binary classification results, MC for multi-class classification results, Dens. for density. For each type of results, the best values are shown in bold.

still interesting to outline that the RF method with hand-crafted features seems to be the more impacted by the nature of training sets. Depending on the configurations of training and testing sets, great variability in terms of results can be observed. For example, when transferring from a noisy photogrammetric type of PC (sub-dataset 4) to noise-free ALS data (sub-dataset 1, 2) or multi-sensor data (sub-dataset 5), results go down from 68.87% for the reference test to, respectively, 31.43%, 9.56% and 5.1% when transferring to sub-dataset 1, 2 and 5; see Figure 2.10d. As expected, the more different the data, the more the RF method struggles to retrieve changes. Indeed, both noisy datasets (sub-datasets 3 and 4) produce similar outcomes. When looking at Figure 2.10b, it is worth noting that the results of tests on the multi-sensor sub-dataset are nearly equivalent to the reference test results corresponding to tests on high-density ALS data. Indeed, the second part of the multi-sensor sub-dataset has a similar configuration to sub-dataset 2. Thus, the extracted hand-crafted features are likely to be similar. However, as can be seen in Figure 2.10, a larger difference of results appears when training on sub-dataset 5 and testing on sub-dataset 2. Finally, let us outline that training on sub-dataset 1 leads to a lower difference among results when transferring to another sub-dataset (Figure 2.10a). Sub-dataset 1 has the same density as sub-datasets 3 and 4, and the first instance of the sub-dataset 5, but is not noisy like sub-dataset 2 and the second instance of the sub-dataset 5.

When looking at deep learning methods, results on transfer learning are quite similar to those obtained without transfer leaning. However, the Siamese architecture seems to be more affected by the transfer learning scenario, considering training on the multi-sensor



(**a**) Sub-dataset 1 (**b**) Sub-dataset 2 (**c**) Sub-dataset 3

(**d**) Sub-dataset 4 (**e**) Sub-dataset 5

Figure 2.10: **Results of different tests of transfer learning.** Each sub-figure presents results when training on the training set of a specific sub-dataset and testing on other sub-datasets. Results marked with stars correspond to reference tests without transfer. Means of IoU over classes of change are given (%). Bin. stands for binary classification results, MC for multi-class classification results, Dens. for density.

dataset and testing on the other sub-dataset. Let us also notice that results issued from transferring to the sub-dataset 4 are often lower. In this case, as seen in Section 2.2.3.1, the characteristics of sub-dataset 4 differ significantly from those of the others, since it corresponds to a photogrammetry-like scenario. Even if change detection on DSMs seems easier for sub-dataset 4 (see results in Tables 2.1 and 2.2), the very different characteristics with respect to the other sub-datasets used for the test make these transfer learning scenarios achieving lower results. More generally, we can observe deep neural network approaches exhibited relatively good generalization properties. This is very encouraging, since one can easily learn on simulated data to classify changes in real PCs. Conversely, when considering the RF method with hand-crafted features (Tran et al., 2018), it seems better to train on a smaller dataset than on datasets with different characteristics.

2.3 Discussion

Complementary to the experimental results presented in the previous section, we provide here an in-depth discussion addressing the change detection methods, the dataset used in this study and the simulation tool that has been employed.

2.3.1 Accuracy of methods

In this study, we have compared methods producing results at different levels on various kinds of 3D change datasets. For the DSMd method, even if the Otsu thresholding substantially improves the results, we observed that changes below 9 m cannot be retrieved. Even if such a situation rarely occurs in our datasets, it could still be problematic in practice, e.g., in low residential zones. One example of low building changes better retrieved with empirical thresholds (set at -3 m and 3 m) can be seen in Figure 2.7.

While deep learning have been shown to generally surpass traditional ones in remote sensing, this tendency was not confirmed in our study. Indeed, existing deep learning methods for 3D change detection only give patchwise results, leading to no fine change object boundaries (Dai et al., 2020). The best results obtained at 2D patch level were those provided by the traditional DSMd coupled with Otsu thresholding and the opening filter. Zhang et al. (2019) also compared their results on DSMd at the 2D patch level, but they assigned a label to each patch of DSMd according to the average height difference (AHD) in the patch. Conversely, we first extracted labels at the pixel level before applying majority vote filtering to produce results at the patch level. However, we have experimentally observed that both strategies (AHD vs. majority vote) were actually leading to the same results. In addition, Zhang et al. (2019) also applied the Otsu thresholding algorithm, but they did not consider a morphological operation to filter out isolated pixels. One other non-negligible difference from our study is that our dataset does not contain vegetation. In such a simple situation, the DSMd method enhanced with a filtering operation and a selection of adequate thresholds with the Otsu algorithm would lead to satisfying results. One can assume the deep networks also benefited from this simpler setup.

Let us recall that Zhang et al. (2018b) reported in the binary case an IoU over the class of change of 67%. However, a direct comparison with our results is not possible, since our dataset is different. Still, it is worth noting that quantitative results showed similar ranges. Similarly to Zhang et al. (2018b), we used a Siamese network with shared

weights as both input data or DSMs. This was also the case for sub-dataset 5 on multi-sensor data, even through the two PCs (from which DSMs have been extracted) have different characteristics. Notice that for this multi-sensor sub-dataset, we also tried a pseudo-Siamese network with unshared weights, as is commonly done when inputs have dissimilarities (Mou et al., 2017; Touati et al., 2020a), even if both inputs remain DSMs. We did not observe convincing results, probably because the pseudo-Siamese networks come with higher complexity (more parameters), thereby requiring larger training sets.

When focusing on methods dealing directly with the raw PCs, we observed a huge gap between traditional approaches and machine learning-based ones. However, one should notice that C2C and M3C2 methods were not especially designed for urban change detection. More particularly, M3C2 was presented in a study using TLS PCs with tremendously higher density than ours. While Lague et al. (2013) sought changes at the centimetric scale within PCs of millimetric resolution (point spacing was about 10 mm horizontally and 5 mm vertically) acquired at a distance of 50 m, our point spacing in sub-dataset 2 with the higher density was about 20 cm to 30 cm, and we focused on changes on a metric scale. Their method is thus possibly inadequate for our settings. Both C2C and M3C2 are very sensitive, and the optimal thresholds are specific to each dataset (conversely to DSMd, where the threshold value is the same for all datasets). Automatic threshold selection, with using the Otsu algorithm, did not lead to satisfying results.

2.3.2 Specificities with regard to the datasets

Our experimental results allowed us to compare different methods on a same dataset and in various conditions. We have also highlighted inherent difficulties of each kind of sub-dataset. The first sub-dataset has a low density, and although it embeds low noise, it contains lots of building shadows and hidden facades due to the acquisition settings. Therefore, both PCs do not contain the same hidden parts and building shadows, making a direct comparison difficult. The second sub-dataset with high density and low noise seems to be the easiest one for change detection and categorization. Indeed, all methods performed best on this sub-dataset. The third sub-dataset is composed of PCs with low density and high noise, and includes some shadows due to the different flight tracks of LiDAR during the acquisition of both PCs. This sub-dataset generally led to results worse than with the fourth sub-dataset, which is also composed of low-density, noisy PCs, but with a lower scanning angle, leading to fewer shadows and to more similar hidden parts between PCs. As already discussed in the previous section, it has a significant

impact on all change detection methods. With this fourth sub-dataset, we aimed to mimic photogrammetric PCs extracted from satellites images. A possibility to enhance the realism would be to adapt the registration error from the acquisition sensor (Nuth and Kääb, 2011), but this remains quite specific to each sensor. Finally, the fifth sub-dataset is composed of multi-sensor data: the first PC is from a low-density, noisy ALS, and the second PC has a high density and low noise. Results achieved on this multi-sensor sub-dataset are similar to those obtained on the first sub-dataset composed of low density but not noisy PCs.

Completion of all experiments led us to conclude that density seems to have more of an impact than noise on the final results. However, since changes in the dataset occurred at building scale (new construction or destruction) and noise was set to be about 1 m, it seems rational that the main changes were retrieved. The main difficulties remain at the buildings' edges, since in this situation, noise makes it difficult to distinct boundaries of changed objects.

Finally, even though Urb3DCD-V1 may appear too simple because it only contains ground and buildings, results show that there are some difficulties in dealing with dense urban areas where building shadows and hidden parts are frequent. These shadows and hidden parts are frequent in real 3D data and greatly challenge the comparison between two point clouds (Czerniawski et al., 2021). Moreover, one can observe that, even for the sub-dataset with high density and low noise, all methods were still far from attaining 100% for their means of IoU over classes of change. Indeed, the best methods obtained about 80% at the 2D pixel level, 90% at the 2D patch level and 70% at the 3D point level. Thereby, this calls for new change detection and categorization methods, and indicates as well the possible difficulties future methods need to surpass with our dataset.

2.3.3 Generation of simulated data

Let us recall that our dataset is made of artificial PCs generated thanks to a novel simulator of multi-temporal 3D urban PCs developed in this thesis. The annotation is done automatically by the simulator, thus avoiding any time-consuming manual annotation. The new construction label corresponds to all points located on a new building. For the demolition class of change, we proceeded by extracting the convex hull of the building footprint on the ground. In our 3D models, the majority of buildings are convex, but in some isolated cases this could lead to some small difference with the actual building footprints. Nevertheless, such a case remains very rare with respect to the large number

of annotated points, so it does not have a significant influence on the quantitative results that have been reported.

Results presented in Section 2.2.3.2 concerning the size of the training set also show some additional interest of our simulator. Indeed, up to 50 simulations of PC pairs have been performed in order to generate the larger sub-dataset 1.c. All these simulations have been made over the same area but, for each generated PC, we randomly selected both the buildings to be updated and the flight tracks to be followed, thus leading to a high variability in the dataset, as illustrated in Figure 1.12. Though each simulated pair may not be considered as a totally new data (since similar buildings could be randomly chosen several times), this process can be seen as data augmentation, allowing us to significantly increase the results, as shown in previous section (e.g., up to 24% for the binary FF network between the smaller (1.a) and the larger (1.c) sub-dataset). Finally, it should be noted that even if the same buildings can be randomly chosen several times, the resulting 3D points lying on the building will not be at the exact same coordinates because of the random flight plan and noise.

2.4 Conclusion

This chapter discussed about 3D change detection methods. First, a review of existing methods given in Section 2.1 divides the literature into two categories: DSMs-based and PCs-based strategies. Then, we have proposed an experimental comparison of different methods of change detection and categorization in an urban environment.

We have experimented and **assessed six different methods for change detection and categorization using either PCs rasterizations in 2D DSMs, or by directly coping with the 3D PCs**. More precisely, we have compared **traditional distance-based methods** such as DSMd with different types of thresholding and a filtering operation (Murakami et al., 1999; Otsu, 1979; Stal et al., 2013), C2C (Girardeau-Montaut et al., 2005) and M3C2 (Lague et al., 2013); with a **machine learning** technique, a random forest fed **with hand-crafted features** (Tran et al., 2018); and with **deep learning** architectures through feed forward (Zhang et al., 2019) and Siamese (Zhang et al., 2018b) networks. These methods provide different results: 2D pixels, 2D patches and 3D points. For the supervised methods, we have also studied the capacity of transfer learning and the influence of the training set size. All experiments have been performed on each sub-dataset of Urb3DCD-V1 described in Chapter 1.

The results of this evaluation showed that **the simulated dataset (Urb3DCD-V1)** in dense urban areas is a **challenge for all existing methods**. **The remaining issues concern the management of point clouds with low densities and the global understanding of the scene in case of occlusions in the point clouds.** For 3D point scale results, traditional machine learning is better than distance-based methods. Existing deep learning methods only provide results at the scale of patches extracted from DSMs, which is much less accurate in terms of results. While deep learning methods seem to be more suitable for transfer learning than machine learning methods, these experiments highlighted the need for deep learning methods that are based directly on raw 3D point clouds. Thereby, the following of thesis will focus on developing original 3D change detection and categorization methods able to understand the 3D context even at a low density or multi-sensor configuration.

Chapter 3

Supervised change detection

Contents

As stated in Chapter 2, existing methods are still limited to efficiently process 3D PC change detection. While deep learning provides interesting results in remote sensing images (Zhu et al., 2017; Ma et al., 2019) or 3D PCs object detection and segmentation (Qi et al., 2017b; Shi et al., 2019; Thomas et al., 2019; Guo et al., 2020), to the best of our knowledge, there is no deep learning method for the multiple change segmentation task dealing directly with raw 3D PCs. Thereby, in this chapter, we aim to tackle this task by proposing the first deep architectures[14] able to deal with multiple change segmentation from 3D PCs, providing results at point scale.

Section 3.1 recalls some state-of-the-art solutions for deep learning on 3D PCs data, and provides as well an overview of deep learning for change detection in 2D images. Inspired by these developments, we build a network based on a Siamese architecture with 3D kernel point convolutions enabling to process directly raw 3D PCs. We extend this network in two versions (described in Section 3.2), named Siamese Kernel Point Convolution (KPConv) for change segmentation and Siamese KPConv Cls variant for change classification in 3D point clouds. Results and discussions are given in Section 3.3. Finally, we propose to further improve change segmentation results by considering hand-crafted features as input to the network and introducing evolutions of Siamese KPConv network [15] in Section 3.4.

3.1 Related work

In this related work section, we will put a focus on deep learning. First, we will see how deep learning has been used for 3D PCs understanding. Then, we will describe how to perform change detection into 2D images using deep learning-based methods.

3.1.1 Deep learning for 3D point clouds

Recent years have seen an increasing interest in developing deep learning frameworks dealing with 3D PCs. Specific PC characteristics (sparsity, irregularity of the distribution of points, continuity, etc.) require in fact particular attention in order to define adapted networks and associated operations.

14. These models were presented at ISPRS Congress 2021 (de Gélis et al., 2021a) and further detailed in a paper published in ISPRS Journal of Photogrammetry and Remote Sensing in 2023 (de Gélis et al., 2023d).

15. These enhancements of Siamese KPConv are submitted for publication and a preprint version is available (de Gélis et al., 2023b).

To this end, existing techniques can be distinguished into three categories, namely projection-based, discretization-based or point-based methods. Projection-based methods consist in projecting 3D PCs onto regular 2D grids (rasters, spheres, etc.) in order to apply traditional existing 2D approaches (Boulch et al., 2018; Wu et al., 2018; Guiotte et al., 2020). In a similar spirit, discretization-based methods also transform 3D PCs into discrete representation in 3D voxels (Tchapmi et al., 2017; Rethage et al., 2018). This rasterization process, whatever the dimensions of the output (2D or 3D), brings severe issues such as loss of information through the aggregation of multiple points into a single cell, and possible empty cells (especially in the 3D case) due to the regular sampling. While being popular in the early years, they are now most often neglected in favor of pure 3D PC approaches.

Conversely, point-based methods are appealing since they avoid rasterization or discretization steps and their aforementioned drawbacks. As reported in a recent survey (Guo et al., 2020), they have become the most popular strategy to deal with 3D PCs. To this end, PointNet (Qi et al., 2017a) allows to learn per-point features using shared Multi-Layer Perceptron (MLP) and global features using symmetrical pooling function to deal with 3D PCs characteristics (orderless and unstructured). A simplified illustration of PointNet is given in Figure 3.1. Further improved by Qi et al., 2017b to group points hierarchically and learn features at different scales, PointNet is still the basis of numerous works in deep learning for 3D PCs (Lang et al., 2019; Shi et al., 2019). However, such a popular framework shows its limitations when applied in a remote sensing context, where large PCs could be acquired through ALS surveys (Landrieu and Simonovsky, 2018) and where no prior assumption can reasonably be made regarding the scene size (in terms of exact number of points).

Alternative point-based strategies have then been introduced to counter weaknesses of PointNet and its variants. Most often, they rely on a specific definition of the convolution operator, and/or on the underlying graph representation. While the latter has led to various successful frameworks (Landrieu and Simonovsky, 2018; Wang et al., 2019a; Wang et al., 2019b; Lin et al., 2021), but requires an initial mapping of the PC into a graph structure, the former has the advantage of being more natural for transferring existing deep learning know-how for 2D images, including the well-explored problem of change detection in remote sensing.

Among graph-based methods, Landrieu and Simonovsky (2018) propose to transform the PC into a superpoint graph. This graph is obtained by performing a geometric

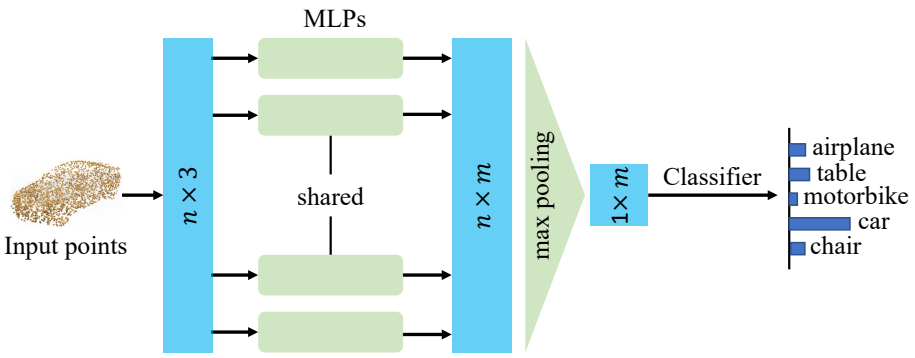


Figure 3.1: **A simplified architecture of PointNet** (Qi et al., 2017a) for point cloud object classification, where parameters n and m denote point number and feature dimension, respectively. Source: Figure from Xiao et al. (2023).

partition, i.e., a division of the PC into simple yet meaningful shape. Each of these shapes is associated to a superpoint. Then, a graph is formed using superpoints as nodes of the graph and edges represent the spatial adjacency between superpoints. The semantic information is extracted through specific edge convolutions applied on the superpoint graph. As it transforms the PC to a smaller set of superpoints, this method is well-suited for large scale PCs. Other graph-based methods, such as Wang et al. (2019a) and Wang et al. (2019b), rely on a simpler graph constructed by linking points from a k-Nearest Neighbors (kNN) neighborhood. Neighbor points are searched first in the spatial space, then, after each convolutional layer, in the current feature space to modify the area of influence of the convolution. Here, a vertex represents each point. Then, different edge convolution operators have been defined according to methods relying on diverse ideas such as MLP (Wang et al., 2019b), attention (Wang et al., 2019a; Zhiheng and Ning, 2019), similarity computation (Lin et al., 2021), focusing on local geometric characteristics (Zhou et al., 2021a), lightweight operators (Landrieu and Simonovsky, 2018), etc.

Let us now focus on the point convolution, which is defined for a point $x \in \mathbb{R}^3$ as:

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathbb{R}^3} g(x_i - x) f_i \tag{3.1}$$

where x_i is a point in the entire set of points $\mathcal{P} \in \mathbb{R}^{N \times 3}$, f_i its corresponding features in the feature set $\mathcal{F} \in \mathbb{R}^{N \times D}$, with D the number of input features and g is the kernel function associated with the correlation. A crucial issue remains in the definition of the kernel function g. A first category is based on MLP and fully connects all points together, i.e.,

the support of g is $\mathbb{R}^3$ (Wang et al., 2018; Li et al., 2018b; Hermosilla et al., 2018; Boulch, 2020), while another family relies on geometric kernels and connects points only on a local neighborhood, i.e., the support of g is a ball $\mathcal{B}^3_{R,x_i} = \{x \in \mathbb{R}^3 \text{ s.t. } \|x - x_i\| \leq R\}$, $R \in \mathbb{R}$ is the neighborhood size. Convolutions involving MLP are more complex, require more trainable parameters and exhibit limited performances. Geometric kernels can be defined with linear functions on kNN (Groh et al., 2018), polynomial functions (Xu et al., 2018), weights in voxels (Hua et al., 2018) or even kernel points (Atzmon et al., 2018; Thomas et al., 2019). Among these convolutions, Kernel Point Convolution (KPConv) (Thomas et al., 2019) achieved very good results on segmentation and classification tasks, even on large urban ALS datasets (Varney et al., 2020). KPConv also outperforms numerous other traditional deep learning methods such as PointNet++ (Qi et al., 2017b) or graph-based methods.

We now recall the main ideas from KPConv, and refer the interested reader to the original paper (Thomas et al., 2019) for more details. The kernel function g defined in KPConv makes it possible to apply different weights to different areas inside the ball $\mathcal{B}^3_{R,x_i}$ of radius R centered on a point x_i of the PC. These weights are defined for all points $\widetilde{x}_k$ inside the ball. These K $\widetilde{x}_k$ points are called kernel points. This domain definition inside a specific area ensures robustness to density variation, which is an interesting property compared to kernel functions based on kNN. Let $\{W_k \mid k \leq K\} \subset \mathbb{R}^{D_{in} \times D_{out}}$ be the associated weight matrices that map features of all kernel points $\widetilde{x}_k$ from dimension D_{in} to D_{out}. Thus, the kernel function g is defined as follows for any centered neighbors $y_i = x_i - x$ with $x \in \mathcal{B}^3_{R,x_i}$:

$$g(y) = \sum_{k \leq K} h(y, \widetilde{x}_k) W_k \tag{3.2}$$

where h is the correlation function between $\widetilde{x}_k$ and y, as defined by equation (3.3). This correlation function makes it possible to define how each kernel point impacts the convolution results. Basically, it should be higher when $\widetilde{x}_k$ is closer to y depending on the influence distance of the kernel points σ:

$$h(y, \widetilde{x}_k) = \max\left(0, 1 - \frac{\|y - \widetilde{x}_k\|}{\sigma}\right) \tag{3.3}$$

Notice that the positions of kernel points are crucial to define KPConv. Thomas et al., 2019 proposed two versions of their convolution with rigid or deformable kernels. In the rigid case, kernel points are distributed in order to be as far as possible from each

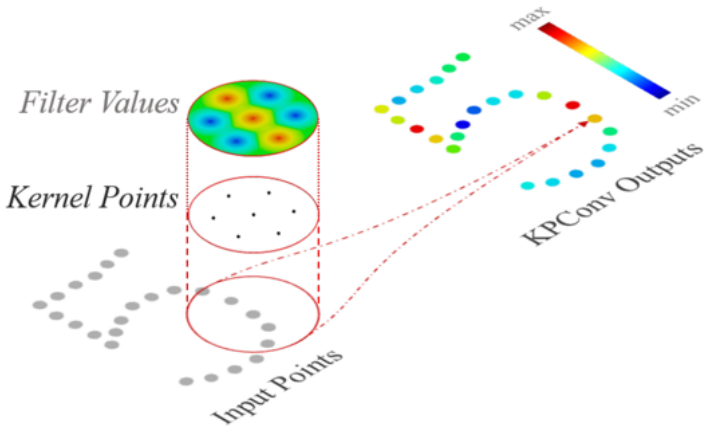


Figure 3.2: **Illustration of Kernel Point convolution on 2D points.** Source: Figure from Thomas et al. (2019).

other. In the deformable case, positions of kernel points are adapted to the PC. In fact, a local shift of each kernel point is learned by the network to adapt to the scene. In practice, deformable kernels considerably increase the number of training parameters and give even worst results than rigid kernels in outdoor scenes where the variability is lower (Thomas et al., 2019). An illustration of KPConv is given in Figure 3.2. These convolutions were included into deep networks, named Kernel Point – Convolutional Neural Network (KP-CNN) and Kernel Point – Fully Convolutional Neural Network (KP-FCNN) to respectively tackle classification and semantic segmentation of 3D PCs. These networks are presented in Figure 3.3.

3.1.2 Deep learning for change detection in 2D images

With the rise of satellite and aerial imagery, several methods have been developed in order to highlight and categorize changes in multi-temporal 2D images. In particular, these past years have seen a lot of deep supervised methods to tackle this issue (Shi et al., 2020a; Jiang et al., 2022a; Shafique et al., 2022) providing some relevant results. Existing deep networks for change detection can be mainly classified into two categories: single-stream and double-stream methods. Both are illustrated in Figure 3.4. Single-stream methods consist of a single branch network where either the difference of images is taken into account (Geng et al., 2017) or data are stacked to create a multi-channel input composed of both images (Gong et al., 2015; Lei et al., 2019; Li et al., 2019b). Although efficient, these early fusion (EF) networks are limited when using heterogeneous data that

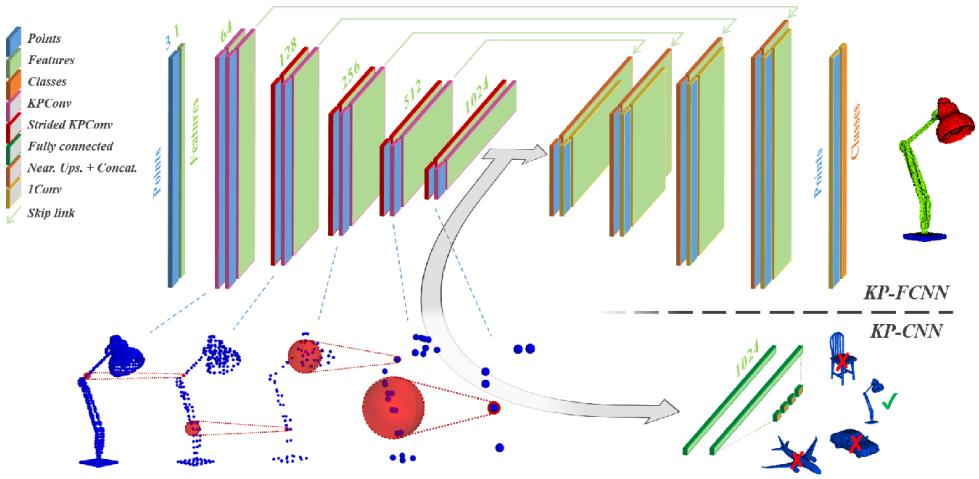
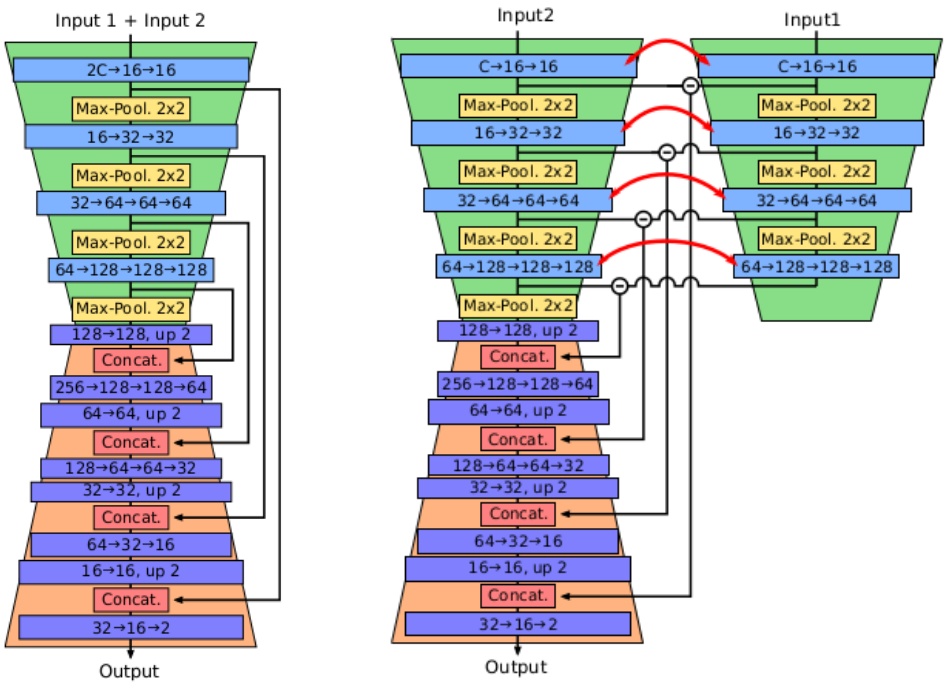


Figure 3.3: **3D deep networks based on KPConv**: KP-CNN and KP-FCNN for classification and semantic segmentation respectively. Source: Figure from Thomas et al. (2019).

cannot be directly fused. For the latter challenge, double-stream networks that involve two distinct branches for both images are more suitable.

Firstly developed in computer vision (Chopra et al., 2005; Zagoruyko and Komodakis, 2015), Siamese networks belong to the double-stream architecture family. Siamese networks are parts of reference architectures for change detection or similarity computations between two inputs. As such, they have been largely used for remote sensing applications (Zhan et al., 2017; Lefèvre et al., 2017; He et al., 2018; Shi et al., 2020a), and they provide reliable results even with significantly heterogeneous inputs, such as optical and SAR images (Mou et al., 2017). In particular, a Siamese network encoder part is composed of two similar branches extracting features from input data, which will then be fed into a decision-maker component to highlight changes. Thus, each input image is given separately to a branch of the encoder acting as a feature extractor. Usually, the two branches of the encoder share exactly the same architecture. However, their weights may be shared for pure Siamese networks (Zhan et al., 2017; Hedjam et al., 2019; Jiang et al., 2020) or unshared in pseudo-Siamese networks (Touati et al., 2020a; Xu et al., 2020). The latter lead to more flexibility to deal with data of various sources even though the number of trainable parameters is higher, yielding more complexity during the training stage (Dong et al., 2018). In addition, in order to classify changes, one can also use deep Siamese fully convolutional network (FCN). As in a conventional Siamese network, the encoder part is composed of two branches. Each branch is a succession of traditional convolution and



(**a**) Single-stream (FC-EF) (**b**) Double-stream (FC-Siam-diff)

Figure 3.4: **Illustration of single-stream** Fully-Convolutional Early Fusion (**a**) **and double-stream** Fully-Convolutional Siamese with difference (**b**) **methods.** Source: Figures from Daudt et al. (2018).

pooling layers in order to extract information at several scales. A particularity of Siamese FCN remains in concatenating or fusing at each pooling step the difference between extracted features of the two encoder branches to the corresponding scale in the decoder part (Daudt et al., 2018). Finally, Siamese FCNs are inspired by the U-Net architecture (Ronneberger et al., 2015) with skip connections between the encoder and decoder. However, in Siamese FCNs, skip links come from a fusion (by concatenation or differentiation) of information provided by each branch of the encoder part. An illustration of Siamese FCN is given in Figure 3.4b. Siamese networks remain the basis of numerous works in change detection.

It has been shown that taking into account the multi-scale properties of remote sensing images is important for change detection (Yu et al., 2019). First, the focus can be put on multi-scales features by using convolutions with different receptive fields (Zhang et al., 2018a; Wang et al., 2020b; Chen et al., 2022a; Song et al., 2021; Jiang et al., 2022b) or by using other back-bones such as U-Net++ (Zhou et al., 2018) that allows to emphasize

deep connections between each scale of the network and to produce multi-scale outputs that are usually all combined in the loss function (Li et al., 2020; Fang et al., 2021). Note that using U-Net++ back-bone also yields interesting results for single-stream network (Peng et al., 2019). Furthermore, driven by the idea that each layer of a network produces a feature map at different scale, the combination of outputs from different layers can be used to merge multi-scale information. In Mao et al. (2022), they are gathered to feed a transformer. The outputs of each layer of the decoder can be used by directly applying a classifier on them and merging the results of these classifiers (Zhou et al., 2023). Fang et al. (2021) uses an attention module on the concatenation of the outputs of each layer of the decoder to select and focus on more effective information. Chen et al. (2022a) also use attention modules to combine multi-scale information. In particular, the multi-scale features are not only combined through multi-output fusion, but also through multiple attention mechanisms to fuse the multiple scales at each layer. They further introduce selective kernel convolutions that are convolutions with adapted receptive fields (e.g., kernel size and dilation rate) to extract information according to different scales. Multi-scale information is also assessed thanks to adapted convolution module with different kernel sizes (from 1×1 to 9×9) (Xiang et al., 2021; Jiang et al., 2022b). Song et al. (2021) combine a special multi-scale convolution and cross-scale global context fusion to get information from different scales. Notice that studies using different kernel sizes join studies using dilated convolutions with different rates (Zhang et al., 2018a; Wang et al., 2020b; Chen et al., 2022a) with the common goal of varying the receptive field of the convolutions. Bao et al. (2020) propose to combine pixel-level and patch-level information. Lei et al. (2022) focus more on multi-scale boundary extraction to enhance building change detection.

Recent studies have also shown that data fusion is a crucial step in change detection. Paying more attention on how to fuse information coming from the two network inputs can improve change detection results. It has for example been demonstrated that multi-temporal fusion leads to better results when it is performed at multiple scales (Daudt et al., 2018; Chen et al., 2019a; Zhang and Shi, 2020; Zheng et al., 2023). While Daudt et al. (2018) propose to merge information from both branches either by concatenation or differentiation of features, other studies propose more advanced fusion modules. For example, Song et al. (2021) propose a network based on the three results of addition, subtraction, and concatenation of features at multiple scales. Jiang et al. (2022b) take a step aside from the traditional Siamese network with one input for each branch, by

proposing to take in one branch the concatenation of the images and in the other the difference forming two sub-networks with different properties. At the output of each layer, the features of the two branches are summed and then concatenated at the corresponding scale in the decoder thanks to skip connections. Yin et al. (2023) propose to embed some fusion modules relying on multi-scale features difference aggregation and attention on concatenation of bi-temporal features. Note that according to Peng et al. (2020), taking into account both concatenation and difference of input images is more efficient even in single-stream methods. As with multi-scale consideration, another category of methods uses attention mechanism to help the network focus on the most important features for multi-temporal information fusion (Jiang et al., 2020; Chen et al., 2021b; Song et al., 2021; Chen et al., 2022a).

Finally, while the majority of studies use a traditional classification loss function such as cross-entropy (Jiang et al., 2022a), it is also possible to adapt the loss function to the change detection task. In particular, a contrastive loss function enabling to push away features of the changed pixels and get closer features of the unchanged pixels is sometimes used (Zhan et al., 2017; Wang et al., 2020b; Wang et al., 2023). Zhang et al. (2018a) further improve change detection by using a triplet loss function instead of a contrastive loss function to take advantage of greater spatial relationship between pixel.

3.2 Siamese KPConv: 3D change detection with deep learning

The following section describes the proposed methods for change detection between bi-temporal 3D PCs whether at PC or points scale levels (see Figure 9 in the general Introduction). Based on the literature of change detection in 2D images and on the state-of-the-art in deep learning for processing 3D PCs, we propose a Siamese FCN with Kernel Point Convolution (KPConv). In fact, standard 2D convolution involved in Siamese FCN (Daudt et al., 2018) is not directly suitable for 3D PCs. We therefore combine Siamese FCN with specific 3D PC convolutions, namely KPConv (Thomas et al., 2019). Indeed, as pointed out in the Section 3.1.1, KPConv is chosen because of its high performances against the state-of-the-art and its intrinsic compatibility with the Siamese framework. We recall the appealing properties of KPConv over the well-established PointNet in our change detection context, i.e., its ability to scale to large datasets and to deal with different numbers of points from each of the input PCs.

3.2.1 Siamese KPConv network

To extend the Siamese principle to 3D PCs, we propose here to embed the KPConv in a deep Siamese network, as presented in Figure 3.5. We detail here the different parts of our architecture. Both input PCs will pass through encoders consisting of a stack of five layers containing two convolutional blocks, the first one being "strided" except for the first block.

Convolutions are performed here with KPConv presented in Section 3.1.1. To mimic 2D "strided" convolutions, "strided" KPConv operations reduce the number of points to compute features at different scales. At each layer j, the cell size dl_j corresponding to the minimum distance between two consecutive points is recursively defined as $dl_j = 2 \times dl_{j-1}$. As for the first layer, dl_0 is set according to the dataset density and the level of detail in the changes we aim to retrieve. KPConv radius R for convolutions also depends on the layer and is set to $R_j = 2.5 \times dl_j$. The decoder part is composed of a stack of five layers holding a nearest upsampling and concatenation stage and a unary convolution. The unary convolution behaves like a fully connected layer. We can observe that encoder and decoder architectures are very similar to KP-FCNN used for semantic segmentation (Thomas et al., 2019).

Equivalently to a typical FCN with skip connections, the network enables the passing of information between intermediate layers of the encoder and the decoder. In Siamese networks however, a strategy should be used to fuse data coming from both encoders. Daudt et al. (2018) showed that a difference of features coming from both encoder layers gives better results for change detection. The same conclusion is made in SiamGCN (Ku et al., 2021): the difference of features leads to more accurate results than concatenating both sets of features into the decoder part. Inspired by these results, we concatenate the difference of extracted features associated with the corresponding encoding scale (see Figure 3.5). In practice, computing such feature difference is not obvious, since PCs do not contain the same number of points and are not defined at the same positions, even in non-changed areas. To cope with this issue, we compare each point of the second PC with its spatially closest point in the first PC. Thus, for two PCs $\mathcal{P}_1$ and $\mathcal{P}_2$, with their corresponding features $\mathcal{F}_1$ and $\mathcal{F}_2$, the feature difference $\ominus$ is computed between features $f_{2i} \in \mathcal{F}_2$ of each point $x_{2i} \in \mathcal{P}_2$ of the second PC and features $f_{1j} \in \mathcal{F}_1$ of the nearest point $x_{1j} \in \mathcal{P}_1$. Thereby:

$$(\mathcal{P}_1, \mathcal{F}_1) \ominus (\mathcal{P}_2, \mathcal{F}_2) = f_{2i} - f_{1j|j=\arg\min(\|x_{2i} - x_{1j}\|)} \tag{3.4}$$

Within the encoder, "strided" convolutions sub-sample PCs at each layer, leading us to perform nearest neighbor computation for the feature difference each time the PC is sub-sampled.

Let us observe that while both our Siamese KPConv network and the original KP-FCNN share the principle of embedding KPConv into a deep neural network, they significantly differ to address their respective tasks: semantic segmentation for KP-FCNN vs. multiple change segmentation for our Siamese KPConv. Indeed, our model relies on two encoders enabling to take two different PCs as input, before fusing the encoded information through some subtraction layers.

The network takes as input the 3D point coordinates and, similarly to state-of-the-art deep models for 3D PCs, is also flexible to any supplementary input features such as RGB information, LiDAR intensity, etc. In practice, literature reports that there is no systematic gain when using color information (Boulch, 2020). Fusion of color and geometric information can lead to better results but remains an open problem (especially when they come from two different data sources) (Widyaningrum et al., 2021). Since this question is out-of-scope of our study, we simply recommend following the usual practice in the field (characterize each point by the geometric coordinates X,Y,Z and any available supplementary features such as RGB, number of echoes, etc.) as early done by the authors of PointNet (Qi et al., 2017a). Supplementary features can still be easily added as inputs by modifying the input dimension of weight matrix of kernel points of the first layer.

Let us note that beyond possible supplementary features, one constant feature is always kept to encode the geometry of the PC for the network (as illustrated by the depth of the input feature map (in green) in the Figure 3.5). We refer the reader to the original KPConv paper (Thomas et al., 2019) for more details.

We propose two versions of this network: encoder with shared or unshared weights (the latter being equivalent to a pseudo-Siamese network). Let us notice that even if weights are not shared in two encoders of the Pseudo-Siamese version, other hyper-parameters remain similar. Both will be evaluated in Section 3.3.3. Usually, pseudo-Siamese networks are used when data to be compared come with different characteristics.

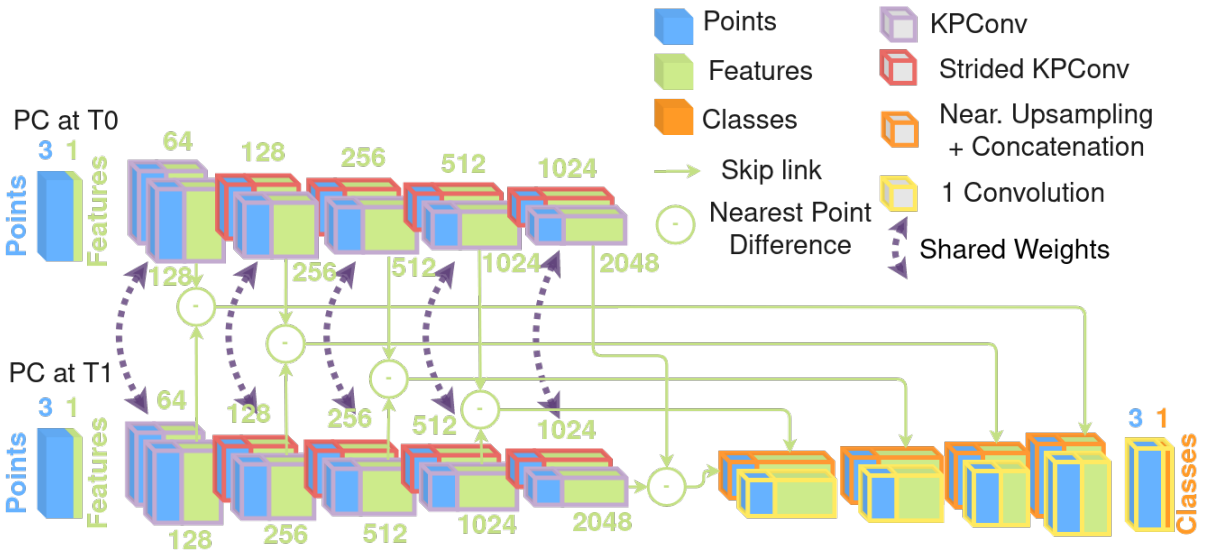


Figure 3.5: **Our Siamese KPConv network architecture.** The Pseudo-Siamese version of the network is the same without shared weights symbolized by dotted purple arrows. Links between successive layers are omitted for the sake of concision.

3.2.2 Siamese KPConv network for classification of change at PCs scale

In order to compare our proposed method to the state-of-the-art which remains limited to PCs change classification, we built a second version of Siamese KPConv dedicated to this task (see Figure 9b in the general Introduction), henceforth referred as Siamese KPConv Cls. The architecture is presented in Figure 3.6. It is composed of the same encoder part as in Siamese KPConv network, except that a fully connected layer has been added at the end of the last layer. Then, features coming from the last layer of each encoder are fused through a difference based on nearest neighbor (as in the previous architecture), that will feed the input of a fully connected layer. A global average pooling is done in order to downscale to the global PC scale. Finally, after a last fully connected layer, PC change classification results are obtained.

Notice that several configurations of this network have been empirically tested to select the best architecture in terms of number of layers or parameters at each layer. Let us note that while SiamGCN contains an average pooling layer after each encoder (i.e., before the feature difference), we have experimentally observed that for our Siamese KPConv Cls, applying the average pooling after the feature difference was leading to better results.

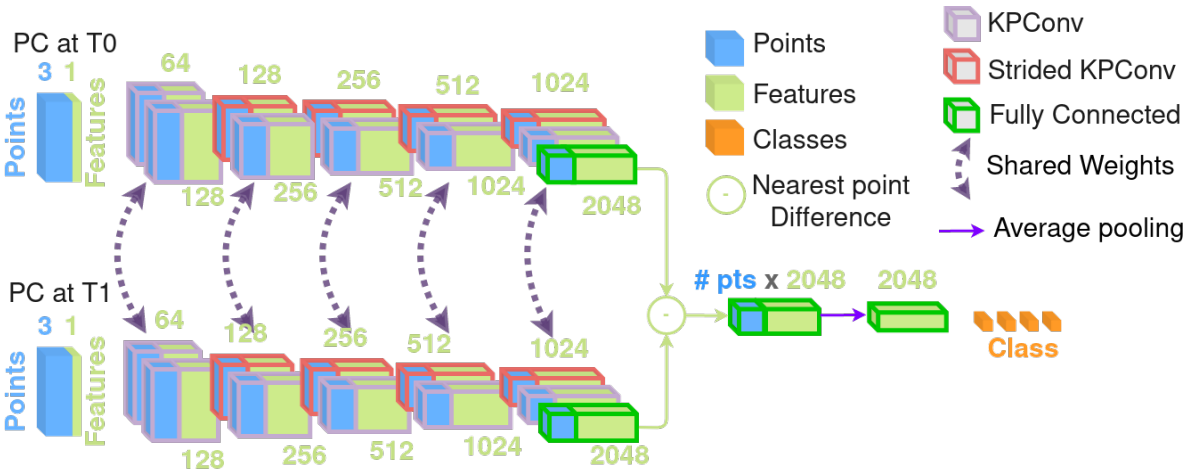


Figure 3.6: **Our Siamese KPConv Cls network architecture for classification.**

3.3 Experimental assessment

In the following section, we present the experimental results of our methods on both simulated and real datasets. Before describing them into detail, let us first introduce the experimental protocol.

3.3.1 Experimental protocol

To compare our approach with typical change detection techniques, we first compare our method for change segmentation with a traditional machine learning approach based on the RF model and trained using handcrafted features proposed by Tran et al., 2018. We consider this technique as representative of the state-of-the-art since it obtains the best results for change detection at 3D point level on Urb3DCD-V1 dataset, as demonstrated in Chapter 2. We recall that we re-implemented the computation of all features of Tran et al., 2018 except those using LiDAR's multi-target capability because Urb3DCD datasets do not contain such information. As mentioned above, to the best of our knowledge, there is no deep learning method for change detection operating directly on 3D PCs. Nevertheless, we have designed two deep learning baselines illustrating the current performances of existing networks for change detection. Inspired by the work on 2D images of Daudt et al., 2018 or on 2.5D DSMs (Zhang et al., 2019), we consider DSMs extracted from our PCs as input 2D matrices to train a fully connected Siamese network (DSM-Siamese) and a fully connected network with early fusion (DSM-FC-EF). These networks rely on

usual 2D convolutions performed on 2D rasterization of PCs. Architectures are similar to those presented in Daudt et al., 2018. DSM-Siamese decoder relies on features difference to gather information from both encoder branches. 2D results can be straightforwardly propagated back to original 3D PCs to be compared with pure methods dealing with raw 3D PCs. Finally, our proposed method as well as the DSM-Siamese one are both tested using Siamese and Pseudo-Siamese networks, i.e., with shared or unshared weights respectively, between Siamese branches. To evaluate the variability of our results, all tests have been conducted at least three times and we report average and standard variation of performances. This strategy will be followed in all experiments presented in this thesis.

These experiments are carried on Urb3DCD-V2 LiDAR low density and multi-sensor sub-datasets, as well as on AHN-CD datasets. We recall that these datasets were presented in Chapter 1.

Concerning change classification, we propose to compare our Siamese KPConv Cls with SiamGCN network (Ku et al., 2021). This Siamese network relies on graph convolution, in particular edge convolution (EdgeConv) operator based on MLP (Wang et al., 2019b). From input PCs, graphs are constructed from the kNN connections. Thus, points form graph vertices and edges are set according to kNN relationships. Conversely to our method, the merging of the two branches of the Siamese network is done after a max-pooling operation, thus it does not imply a point-to-point substraction of features. This is an important difference with our Siamese KPConv Cls architecture. Finally, our method is also compared on the Change3D dataset to Point Cloud Change Detection with Hierarchical Histograms (PoChaDeHH) and Hybrid Graph Inception Change Detection (HGI-CD) algorithms. These two methods competed with SiamGCN in SHREC21 challenge (Ku et al., 2021). PoChaDeHH is a fully handcrafted method based on histogram clustering. HGI-CD relies on both handcrafted and learned-based features. The learned-based part relies on Graph Convolution Network (GCN). The handcrafted parts of these two methods were specifically designed for the Change3D dataset experiments and as such, cannot be applied to any other dataset, including our Urb3DCDV2-Cls. Comparison with HGI-CD and PoChaDeHH is then limited to the Change3D dataset.

Since in change detection and categorization datasets are in general largely imbalanced (i.e., most data belong to the unchanged class despite this class not being the most interesting one), we prefer to discard the overall accuracy or precision scores that are not very indicative of method performance in such settings. We therefore select the mean of accuracy (mAcc) and the mean of IoU over classes of changes (mIoU_{ch}) for reliable

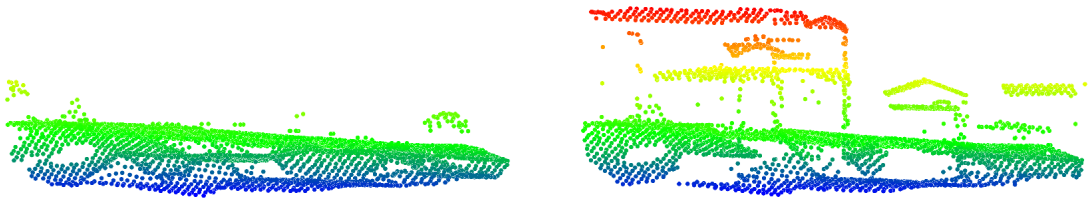
quantitative assessment of the different methods. IoU formula is indicated in Equation 2.5, and the accuracy is given in the following formula:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.5}$$

where TP, TN, FP and FN respectively stand for True Positive, True Negative, False Positive and False Negative. The IoU is also reported for each class of change.

3.3.2 Experimental settings

Similarly to the segmentation task in KPConv experiments, we do not feed entire PCs to the network for computational reasons. Indeed, the PCs are too large to be processed as a whole. Thus, Thomas et al., 2019 have proposed to divide their dataset into small spherical sub-clouds. In the context of urban PC change detection, we prefer to use cylinders aligned to the vertical axis rather than spheres since the vertical direction is of a different nature compared to the two horizontal ones. By doing so, we also avoid empty sub-clouds (note that the centers of the cylinders are the same for both dates). Indeed, if a sphere is centered at the top of a building that does not include any ground point, and if this building is demolished, the sphere obtained in the second PC will be free of points and this will disturb the training of the network. To illustrate this, examples of two input cylinders are given in Figure 3.7. Let us consider a point in the center of the building's roof (center of Figure 3.7b). In this case, the corresponding sphere at the second date could have been empty. Indeed, depending on the radius, no ground points would have been visible. By taking cylinders, we ensure that each of the sub-clouds contains ground. At testing, cylinders are chosen regularly with some overlap to ensure that all points are seen at least once by the network. For points seen several times, predicted probabilities are averaged to decide the final label, similarly to voting schemes. It should be outlined that classes are largely imbalanced in the change detection problem. As a matter of fact, the unchanged area represents up to 98% of points according to datasets. Thus, during training, the centers of the cylinders are chosen thanks to a weighted random drawing. Weights are set as a function of dataset balance, in order to set the probability higher for smaller classes. This allows our network to regularly observe changes during the training phase. Moreover, we perform data augmentation through both random rotation around the vertical axis for each selected cylinder and random Gaussian noise at point scale. Notice that a random rotation angle is selected for each pair of cylinders and to keep



(**a**) First cylinder (**b**) Second cylinder

Figure 3.7: **Example of input cylinders with changes between the first and the second cylinders** (buildings have been added). The two input PCs (**a**-**b**) are colorized based on their relative elevation.

valid the registration, the same rotation angle is applied both PCs in the pair.

As mentioned in Section 3.2, a first sub-sampling rate (dl_0) has to be chosen to design the network. In practice, this has been set to 1 m for experiments on a simulated dataset. We have empirically set the radius of cylinders to 50 m, following the recommendation of KPConv authors who set the radius to $50 \times dl_0$. For the real dataset AHN-CD, as density is higher than in Urb3DCD-V2 datasets, we set dl_0 to 0.5 m, implying cylinders of 25 m in radius. According to our experiments, a compromise should be made to use cylinders as large as possible to take into account enough context and the sub-sampling rate, to avoid losing too many available points.

Parameter settings (summarized in Table 3.1) have been largely influenced by the original KPConv proposed values. We thus use a SGD with a momentum of 0.98, to minimize a point-wise negative log-likelihood (NLL) loss, given by the following equation:

$$\ell_{NLL}(y_t, y_p) = -(y_t \log(y_p) + (1 - y_t) \log(1 - y_p)) \tag{3.6}$$

where y_t and y_p correspond to the target label and the predicted label, respectively. As prediction is expected at point scale on the second PC, the loss is applied for each point $x_{2i} \in \mathcal{P}_2$ and its corresponding predicted (y_{2i_p}) and ground truth (y_{2i_t}) labels. A batch size of 10 is used. The initial learning rate is set to 10^{-2} and scheduled to decrease exponentially.

Unlike KP-FCNN, we included a probability dropout of 0.5 in the last classification layer. Conversely, no dropout is used for Siamese KPConv Cls. In addition, in order to prevent overfitting, we set a L_2 loss regularization balanced by a coefficient of 10^{-6}.

Concerning Kernel Point Convolution, experiments were conducted with rigid kernels of 25 points. Finally, and as already indicated, input cylinders are randomly chosen in training. Thus, the number of input cylinders is another hyper-parameter to be set. After experimenting with several configurations, the best results were obtained when 6,000 pairs of cylinders were seen by the network per epoch, which corresponds to 600 optimizing steps with a batch size of 10. As for the validation, 3,000 and 500 pairs are used for Urb3DCD-V2 and AHN-CD datasets, respectively.

Concerning change classification task, experiments are performed on Urb3DCD-Cls and Change3D. Regarding Change3D, only 3D coordinates of the center of objects of interest are given, further preparation of the dataset is left to the user. In particular, the authors suggest extracting a vertical cylinder centered on the point of interest. In our experiments, we decided to extract vertical cylinders of 3 m in radius, as done in the pre-processing step of the SiamGCN deep learning method. Thus, concerning Urb3DCD-Cls and Change3D, inputs are already cylinders of 15 m and 3 m in radius. Thereby, only the first sub-sampling rate should be chosen to run experiments with our Siamese KPConv Cls architecture. It has been set respectively to 0.3 m and 0.06 m for Urb3DCD-Cls and Change3D dataset. As a matter of fact, the scales of changes to retrieve are different and dl_0 has to be adapted to expected changes. Regarding Siamese KPConv Cls, the same training parameters as for Siamese KPConv are used except for the learning rate set to 10^{-3}, as done in object classification experiment by Thomas et al. (2019) in their KPConv study. As the results are expected at pairs of PCs scale, the NLL loss is computed for each pair of PCs target and prediction labels. As datasets for classification are smaller, Siamese KPConv Cls network is trained on 1,000 examples of pairs of cylinders per epoch. Batch size is also set to 10 for the classification task. For experiments over the Change3D dataset, RGB information is needed to distinguish the class 'color change'. Thereby, for these experiments, our Siamese KPConv Cls network takes as input both RGB information and 3D coordinates.

The whole development is implemented in PyTorch and relies on KPConv implementation available in Torch-Points3D (Chaton et al., 2020). Concerning the nearest point feature difference (Equation 3.4), the nearest point is determined thanks to the kNN implementation available in PyTorch Geometric, which is graphics processing unit (GPU) compliant for faster computation.

In order to compare with more traditional machine learning techniques, we also perform a RF for change detection. For this comparison, let us remark that some features are

	Optimizer	Initial LR	LR scheduler	Dropout	Loss	Batch size	Convolution
Siamese KPConv	SGD	10^{-2}	Exponential	Yes	NLL	10	Rigid KPConv
Siamese KPConv Cls	SGD	10^{-3}	Exponential	No	NLL	10	Rigid KPConv
SiamGCN	Adam	10^{-3}	Step	No	NLL	16	EdgeConv
DSM-based DL	Adam	10^{-3}	Exponential	No	NLL	32	2D convolution

Table 3.1: **Summary of training parameters for deep learning methods.** Notice that training parameters for DSM-based deep learning methods are all the same for the three different networks experimented. LR stands for learning rate.

dependent on the neighboring radius size. To choose this radius we tested several values. We then set it to 5 m, 4 m and 3 m respectively for the low density simulated dataset, the MS simulated dataset and AHN-CD.

Concerning deep learning methods dealing with DSM, the same architectures as Daudt et al., 2018 are set up. DSM resolution is set to 0.5 m for Urb3DCD-V2 and 0.3 m for AHN-CD.

A summary of parameters according to methods and datasets is given in Table 3.2.

Finally, regarding comparisons on the classification task, results for PoChaDeHH and HGI-CD are directly taken from the publication of Ku et al., 2021. For SiamGCN, experiments have been done using the implementation provided by the authors, training parameters are given also in Table 3.1. Let us note that the difference between our results and the original SiamGCN paper (Ku et al., 2021) come from the methodology used for training/validation/testing data splitting. Indeed, we initially failed to reproduce their results. We thus investigated their code and found that all the training, validation and testing steps were carried out on the same set. This is obviously a severe flaw from a machine learning perspective[16]. In our experiments, we have followed a scientifically rigorous strategy, i.e., put apart the testing set and divide the train set into training and validation splits according to the ratio 80/20% as it was also done by authors of HGI-CD. Notice that for fair comparison, the same training and validation split is used for the training of Siamese KPConv Cls.

Concerning all experiments using deep learning, a single GPU (Nvidia Tesla V100 SXM2 16 GB) is used to perform training and inferences.

16. We have contacted the authors, as well as the Editor-in-Chief of *Computer & Graphics*, and the publication of a corrigendum is in progress.

Dataset	Deep learning 3D Siamese KPConv (Cls) Input cylinders radius (m)	dl_0 (m)	Input samples Training	Validation	Deep learning 2D DSM resolution (m)	RF Neighboring radius (m)
Urb3DCD-V2-1	50	1	6,000	3,000	0.5	5
Urb3DCD-V2-2	50	1	6,000	3,000	0.5	4
AHN-CD	25	0.5	6,000	500	0.3	3
Urb3DCD-Cls	15	0.3	1,000	all	-	-
Change3D	3	0.06	1,000	all	-	-

Table 3.2: **Summary of input parameters** according to the five datasets and the three families of methods.

3.3.3 Results

3.3.3.1 Semantic change results on synthetic datasets

Quantitative results concerning the Urb3DCD-V2 dataset for low density LiDAR are presented in Table 3.3 and 3.5 and qualitative results are shown in Figures 3.8 and 3.9. As can be observed, the Siamese KPConv method with both shared or unshared weights for encoders largely improves global results when looking at mAcc or mIoU_{ch}. Indeed, an enhancement of about 30% of mIoU_{ch} can be seen between the traditional machine learning method with handcrafted features and our proposed method. Also, focusing on deep learning-based methods in Table 3.3, we can notice that the direct processing of 3D PCs instead of rasterizing data into DSM highly improves scores. When looking at DSM-based methods, the Siamese and the FC with early fusion are quite comparable though the Siamese is not very stable. Let us emphasize that, when rasterizing PCs into DSM, the size of the training set is considerably diminished, from one label per point to one label per cell/pixel (a 2D pixel gathers multiple 3D points). Since DSM-based networks rely on smaller training sets, we assume they are more prone to overfitting. Pseudo-Siamese networks have more trainable parameters than their Siamese counterpart because the two encoders need to be trained, thereby there might be not enough data to train them properly. This leads to high variation observed within the results.

Concerning per-class performance in Table 3.5, very high scores are reached by our method, especially on 'new buildings', 'new vegetation', 'mobile objects' and 'unchanged' classes as can also be seen in Figure 3.8f. In particular, results are very impressive for mobile objects. To explain this, with an average density of 0.5 points/m^2, mobile objects are represented by only a few 3D points. This leads to very low scores for DSM-based methods since the rasterization process implies a loss of information that is even more

visible on small objects. The vegetation growth class seems to be the hardest to predict for all methods. This is logical, since this category is more related to an evolution than an abrupt change. Furthermore, on vegetation, points are not regularly distributed on the surface of the objects, as LiDAR can penetrate the foliage of trees. More generally, our results (Figures 3.8f-3.9f) are consistent with the ground truth (Figures 3.8c-3.9c). Conversely, the RF method (Figures 3.8d-3.9d) gives less convincing results with several confusions between classes, e.g., in the foreground low building it mixes new building and new vegetation classes in Figure 3.8d (see the ellipse showing the region of interest). In Figure 3.9, some occlusions are shown. While they are very common in processing 3D PCs data especially in dense urban areas, they remain an important challenge for change detection methods. Indeed, as can be observed when comparing both PCs (Figure 3.9(a-b)), hidden facades are not in the same location between the two epochs because of different positions of the sensor during the acquisition. When looking at results of different methods, deep learning based approaches bring better results in these particular areas while the RF algorithm on handcrafted features mix with new building class the building facades that appear only in the second PC (because of occlusion) (see the ellipse showing the region of interest in Figure 3.9d). As our method learns deep features from raw 3D PCs, it seems to be able to characterize objects as a whole. This ability probably comes from the different scales (or network layers) in the feature extraction process. DSM-based methods also provide accurate results in hidden facades. Indeed, the prediction is made only on roofs of buildings by definition of DSM, so predicting no change on the roof leads to the whole facade below to be marked as unchanged as well in the 3D re-projection step. However, DSM-based methods face some problems with occlusions due to building shadows (as no point is acquired resulting in empty pixels in the rasterization) that are generally filled using an interpolation (thus implying imprecision in building edges). When looking at qualitative results of DSM-FC-EF method (Figure 3.9e), one can observe that small roofs details are confused with mobile objects. Indeed, this method rather identifies cars than roofs probably because these details are similar to cars on this low density dataset.

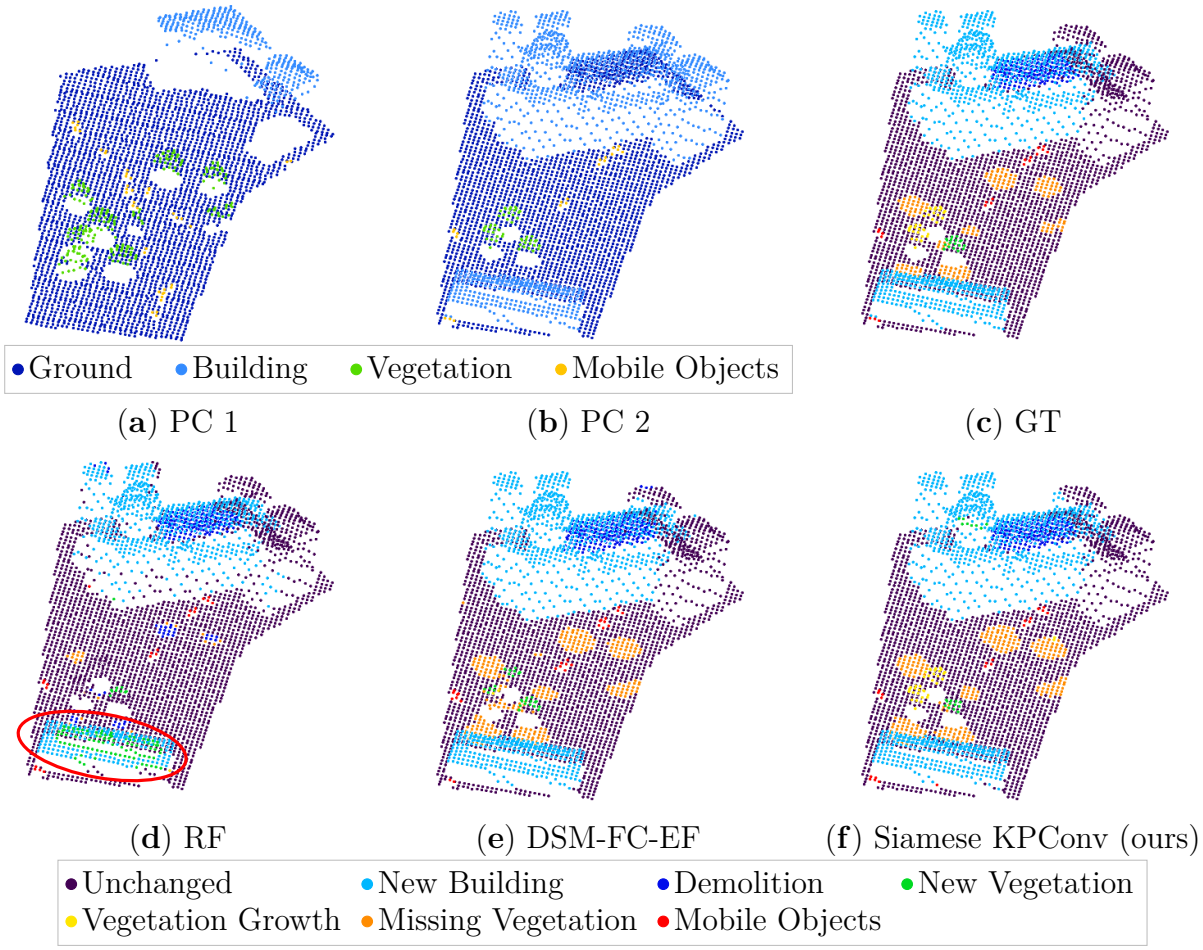


Figure 3.8: **Visual change detection results on Urb3DCD-V2 low density LiDAR sub-dataset:** (**a-b**) the two input point clouds; (**c**) ground truth (GT): simulated changes; (**d**) RF (Tran et al., 2018) results; (**e**) DSM-FC-EF (adaptation of Daudt et al., 2018 FC-EF to DSM inspired by Zhang et al., 2019 works) results; (**f**) our results with Siamese KPConv. Region of interest specifically discussed in the text is highlighted with an ellipse.

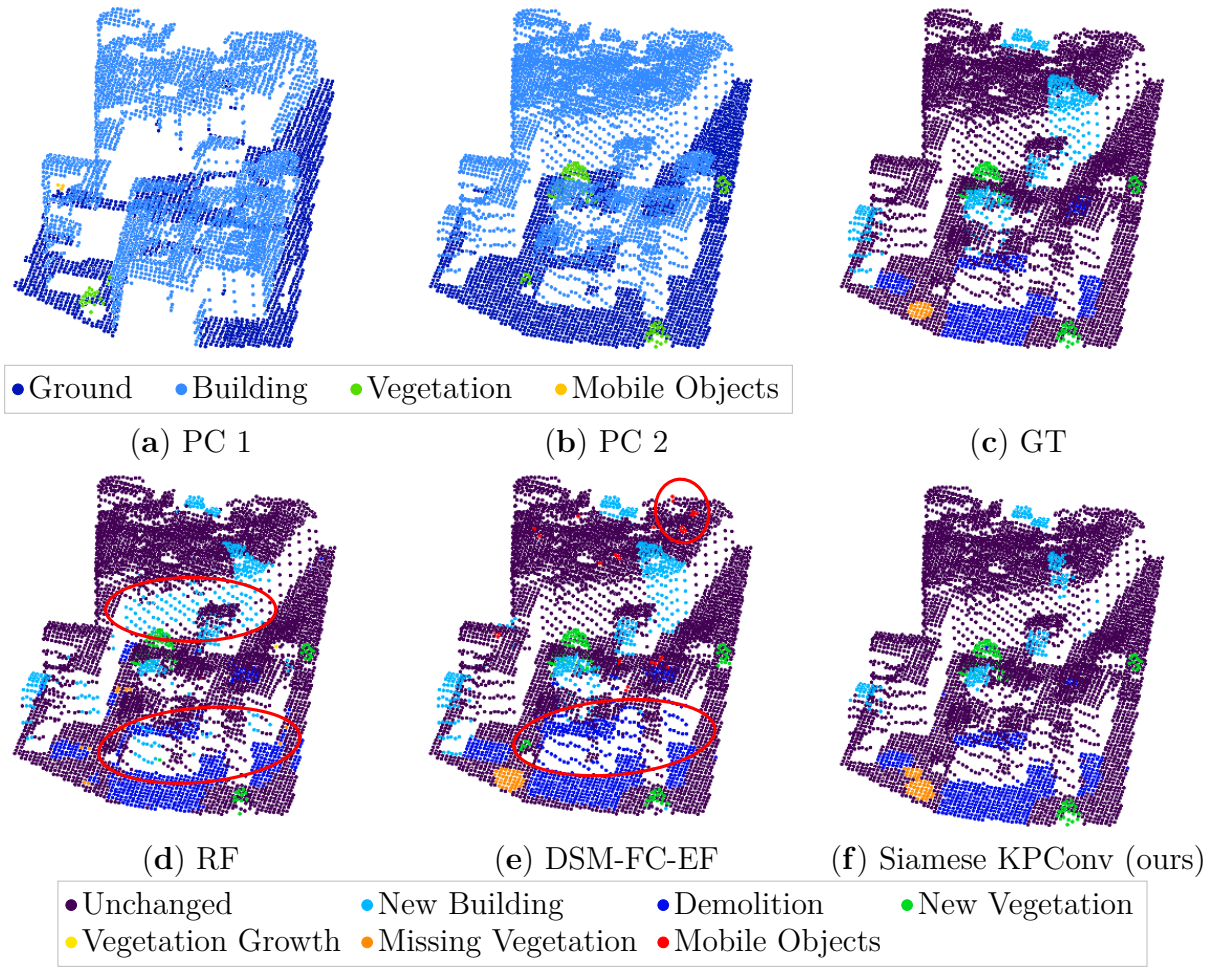


Figure 3.9: **Visual change detection results on Urb3DCD-V2 low density LiDAR sub-dataset in an area containing occlusions:** (**a**-**b**) the two input point clouds; (**c**) ground truth (GT): simulated changes; (**d**) RF (Tran et al., 2018) results; (**e**) DSM-FC-EF (adaptation of Daudt et al., 2018 FC-EF to DSM inspired by Zhang et al., 2019 works) results; (**f**) our results with Siamese KPConv. Regions of interest specifically discussed in the text are highlighted with ellipses.

Method	mAcc	mIoU_{ch}
Siamese KPConv (ours)	91.21 ± 0.68	**80.12** ± 0.02
Pseudo-Siamese KPConv (ours)	**91.31** ± 2.34	77.80 ± 1.69
DSM-Siamese	80.91 ± 5.29	57.41 ± 3.77
DSM-Pseudo-Siamese	75.17 ± 10.03	55.30 ± 8.17
DSM-FC-EF	81.47 ± 0.55	56.98 ± 0.79
RF (Tran et al., 2018)	65.82 ± 0.05	52.37 ± 0.10

Table 3.3: **General results in % on Urb3DCD-V2 low density LiDAR dataset.** DSM-based methods are adaptation of Daudt et al., 2018 networks to DSM inspired by Zhang et al., 2019 works.

Quantitative results for the MS dataset are presented in Tables 3.4 and 3.6. The Siamese KPConv method with shared weights does not outperform state-of-the-art as much as the pseudo-Siamese KPConv does. This was expected, since even if both pieces of data are 3D PCs, they embed very different characteristics. Thus, unshared weights allow each branch of the encoder to specialize in extracting features from one type of sensor. Even for the unshared weights configuration of our method (Pseudo-Siamese KPConv), the results are lower than for the previous dataset. However, the same gap between methods can be seen: Pseudo-Siamese KPConv still improves mIoU_{ch} of about 30% versus the RF method. Among DSM-based methods, early fusion obtains the best results. This is consistent with results obtained in Chapter 2, especially for the MS sub-dataset. As described previously, the number of ground truth labels in the DSM and 3D PCs databases are substantially different because of the rasterization process. Hence, probably due to overfitting problems, it explains why DSM-Pseudo-Siamese is worse than DSM-Siamese even in the MS configuration, conversely to Siamese KPConv results. Furthermore, we believe that in 3D PCs the difference of sensor is more visible than in 2D rasterization. Indeed, in DSMs most differences are seen at edges of buildings which are very distinct in the noiseless DSM while blurry in the noisy DSM. Even if original PCs are very different in terms of quality, they are converted to more similar 2D data during the rasterization process since the same grid size is chosen. Still, the noise present in the first point cloud leads to a noisy DSM, especially on the building edges. Overall, the high similarity between the two input DSMs makes relevant the use of DSM-Siamese with shared weights. Thus, similar filters (and similar weights) can be used to identify changes among pairs of DSMs, conversely to pairs of PCs. Let us note that when applied on 2D images, pseudo-Siamese networks are used mostly in case of pairs of images coming from different sensors, e.g., change detection between optical and SAR inputs (Touati et al., 2020a; Zhou et al., 2021b).

It is worth noting that the quality of data seems to impact less DSM based results when comparing low density and MS results in Tables 3.3 and 3.4, which is in our mind not so surprising because the rasterization process tends to smooth original data by fusing several points into a single pixel.

Concerning per-class results, the same trend as for the low density LiDAR dataset is observed in Table 3.6. When looking at qualitative results in Figures 3.10 and 3.11, the missing vegetation class is almost always mixed with demolition in RF results (3.10d-3.11d). Changed objects boundaries are not precise in DSM-FC-EF results (3.10e-3.11e)

due to the rasterization process. Despite the difference of quality between the two input PCs (3.10(a-b)-3.11(a-b)), our method seems capable of retrieving and classifying changes correctly, even for challenging classes such as vegetation growth. Looking at occlusions visible in Figure 3.11, we can draw the same conclusion as already made on the Urb3CDCD-V2 low density dataset.

Regarding computation time, Siamese KPConv takes about one day train on these synthetic datasets with 6,000 cylinders as input. The inference on the entire testing set is about 4 minutes for the low density sub-dataset and 10 minutes for the MS sub-dataset.

Method	mAcc	mIoU_{ch}
Siamese KPConv (ours)	73.24 ± 5.70	58.55 ± 4.86
Pseudo-Siamese KPConv (ours)	**87.86** ± 0.94	**74.48** ± 0.59
DSM-Siamese	69.91 ± 6.18	49.14 ± 4.92
DSM-Pseudo-Siamese	66.50 ± 10.82	46.60 ± 10.13
DSM-FC-EF	81.25 ± 1.86	55.59 ± 1.84
RF (Tran et al., 2018)	62.20 ± 0.02	46.81 ± 0.01

Table 3.4: **General results in % on Urb3DCD-V2 MS dataset.** DSM-based methods are adaptation of Daudt et al., 2018 networks to DSM inspired by Zhang et al., 2019 works.

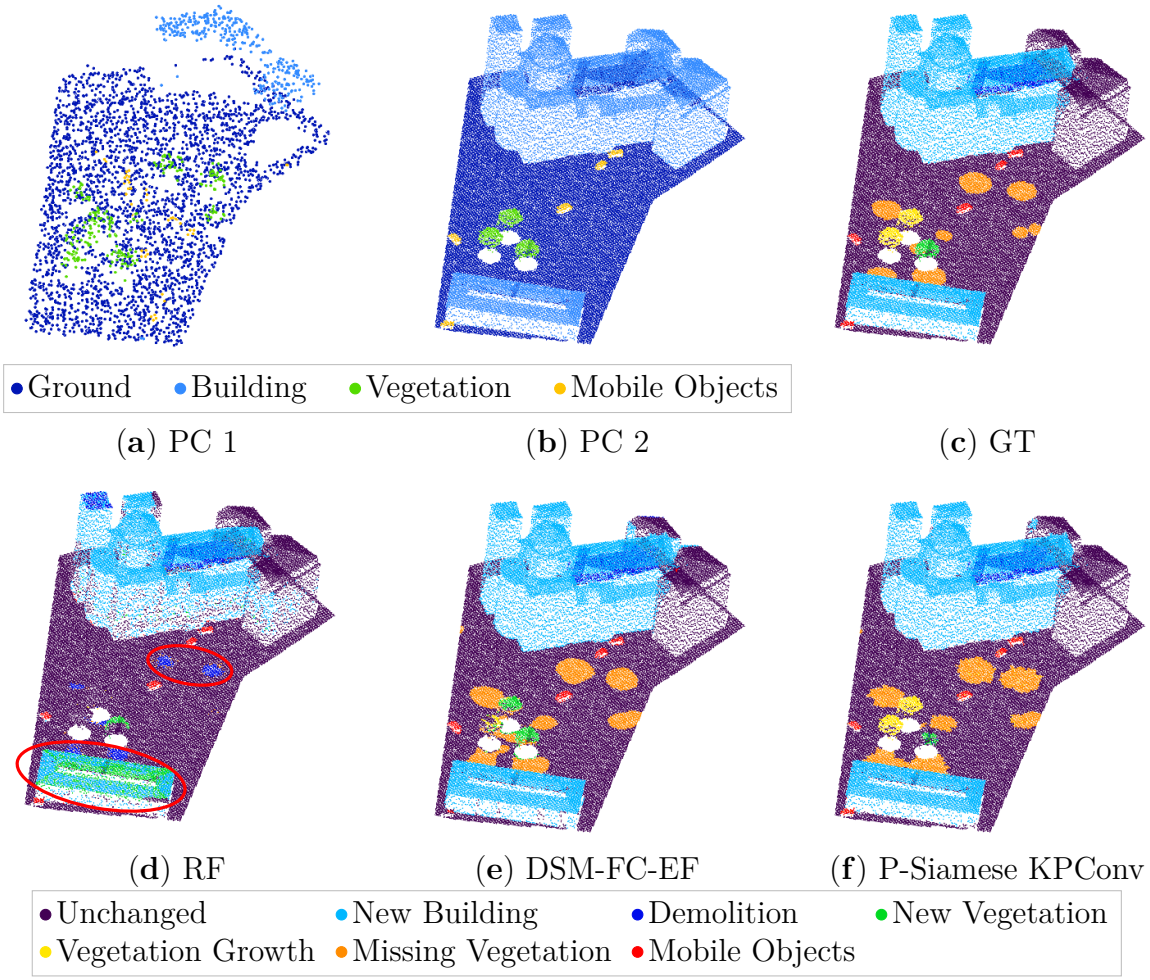


Figure 3.10: **Visual change detection results on Urb3DCD-V2 MS sub-dataset:** (**a**-**b**) the two input point clouds; (**c**) ground truth (GT): simulated changes; (**d**) RF (Tran et al., 2018) results; (**e**) DSM-FC-EF (adaptation of Daudt et al., 2018 FC-EF to DSM inspired by Zhang et al., 2019 works) results; (**f**) our results with Pseudo-Siamese KPConv. Regions of interest specifically discussed in the text are highlighted with ellipses.

Method	Per class IoU (%)						
	Unchanged	New building	Demolition	New veg.	Veg. growth	Missing veg.	Mobile Object
Siamese KPConv	**95.82** ± 0.48	86.68 ± 0.47	**78.66** ± 0.47	**93.16** ± 0.27	**65.17** ± 1.37	**65.46** ± 0.93	**91.55** ± 0.60
Pseudo-Siamese KPConv	95.20 ± 0.18	86.23 ± 1.37	76.08 ± 0.54	92.98 ± 0.95	55.96 ± 9.41	63.50 ± 1.41	91.88 ± 0.71
DSM-Siamese	93.21 ± 0.11	86.14 ± 0.65	69.85 ± 1.46	70.69 ± 1.35	8.92 ± 15.46	60.71 ± 0.74	8.14 ± 5.42
DSM-Pseudo-Siamese	93.44 ± 0.23	84.65 ± 2.05	68.41 ± 1.77	70.38 ± 4.98	15.42 ± 13.81	59.77 ± 3.32	33.15 ± 29.12
DSM-FC-EF	94.39 ± 0.12	**91.23** ± 0.31	71.15 ± 0.99	68.56 ± 3.92	1.89 ± 2.82	62.34 ± 1.23	46.70 ± 3.49
RF	92.72 ± 0.01	73.16 ± 0.02	64.60 ± 0.06	75.17 ± 0.06	19.78 ± 0.30	7.78 ± 0.02	73.71 ± 0.63

Table 3.5: **Per-class IoU scores on Urb3DCD-V2 low density LiDAR dataset.** DSM-based methods are adaptation of Daudt et al., 2018 networks to DSM inspired by Zhang et al., 2019 works. Results are given in %. Veg. stands for vegetation.

Method	Per class IoU (%)						
	Unchanged	New building	Demolition	New veg.	Veg. growth	Missing veg.	Mobile Object
Siamese KPConv	91.68 ± 1.38	55.90 ± 15.65	66.80 ± 0.44	70.94 ± 11.07	42.50 ± 4.88	48.43 ± 4.35	66.74 ± 2.39
Pseudo-Siamese KPConv	**95.52** ± 0.19	83.34 ± 2.21	**76.22** ± 1.08	**85.76** ± 0.50	**59.35** ± 1.00	57.55 ± 0.89	**81.98** ± 0.87
DSM-Siamese	92.85 ± 0.11	87.08 ± 0.70	66.10 ± 0.59	67.47 ± 2.69	1.78 ± 3.09	58.93 ± 0.82	13.51 ± 23.39
DSM-Pseudo-Siamese	93.10 ± 0.42	84.73 ± 1.74	63.33 ± 5.59	62.82 ± 9.71	13.49 ± 10.71	35.22 ± 24.53	20.02 ± 20.42
DSM-FC-EF	93.99 ± 0.12	**90.57** ± 0.61	71.15 ± 1.22	58.74 ± 0.76	6.31 ± 4.49	**62.82** ± 0.68	43.96 ± 4.84
RF	91.59 ± 0.00	68.96 ± 0.01	58.78 ± 0.02	72.65 ± 0.03	13.88 ± 0.09	4.26 ± 0.00	62.31 ± 0.07

Table 3.6: **Per-class IoU scores on Urb3DCD-V2 MS dataset.** DSM-based methods are adaptation of Daudt et al., 2018 networks to DSM inspired by Zhang et al., 2019 works. Results are given in %. Veg. stands for vegetation.

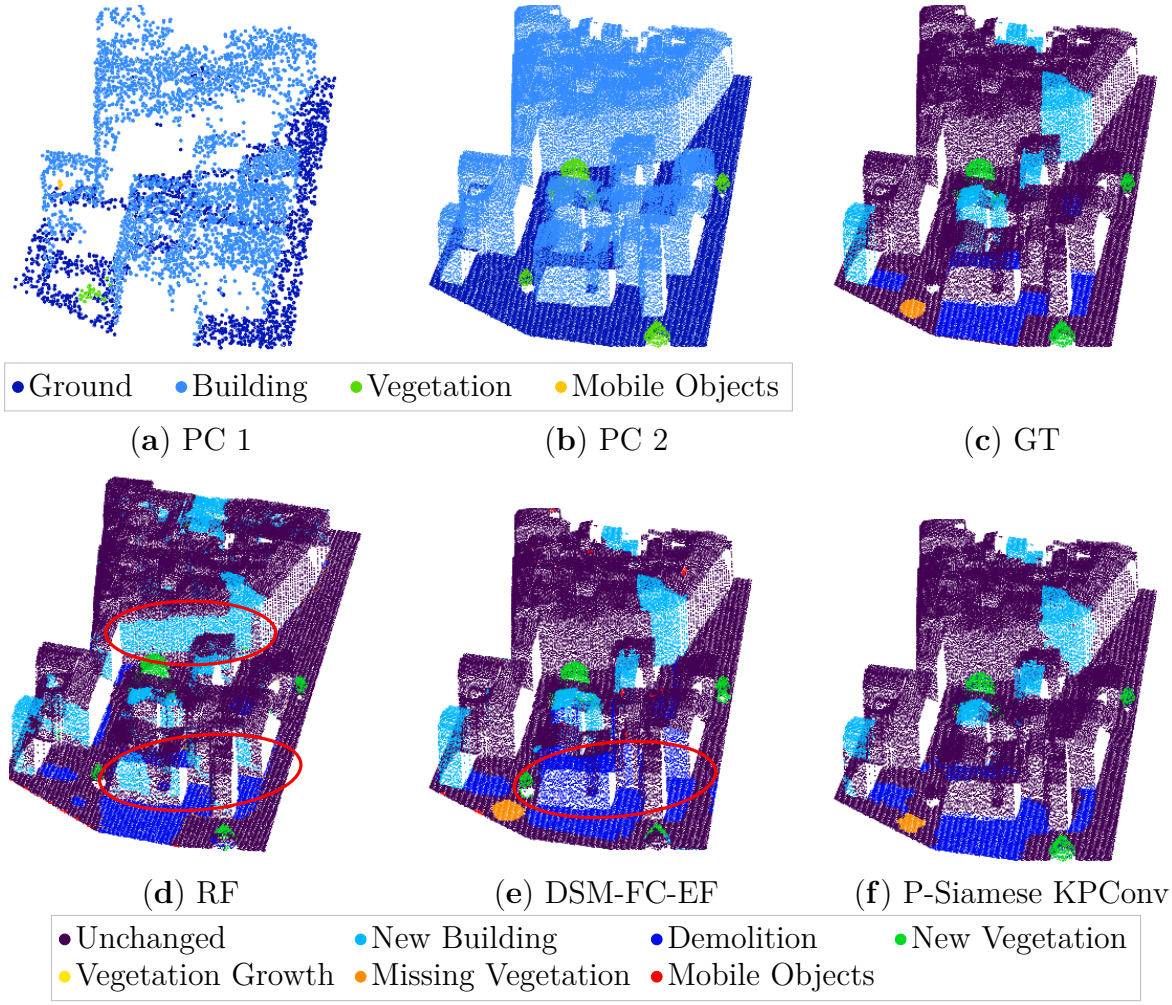


Figure 3.11: **Visual change detection results on Urb3DCD-V2 MS sub-dataset in an area containing occlusions:** (**a**-**b**) the two input point clouds; (**c**) ground truth (GT): simulated changes; (**d**) RF (Tran et al., 2018) results; (**e**) DSM-FC-EF (adaptation of Daudt et al., 2018 FC-EF to DSM inspired by Zhang et al., 2019 works) results; (**f**) our results with Pseudo-Siamese KPConv. Regions of interest specifically discussed in the text are highlighted with ellipses.

Method	mAcc	mIoU_{ch}
Siamese KPConv (ours)	81.86 ± 0.72	**59.93** ± 0.14
Pseudo-Siamese KPConv (ours)	**84.44** ± 1.24	52.32 ± 4.31
DSM-Siamese	62.85 ± 1.13	33.18 ± 3.56
DSM-Pseudo-Siamese	67.04 ± 0.77	41.40 ± 0.62
DSM-FC-EF	74.98 ± 0.80	44.73 ± 2.16
RF (Tran et al., 2018)	50.11 ± 0.01	28.56 ± 0.02

Table 3.7: **General results on AHN-CD dataset** given in %. DSM-based methods are adaptation of Daudt et al., 2018 networks to DSM inspired by Zhang et al., 2019 works.

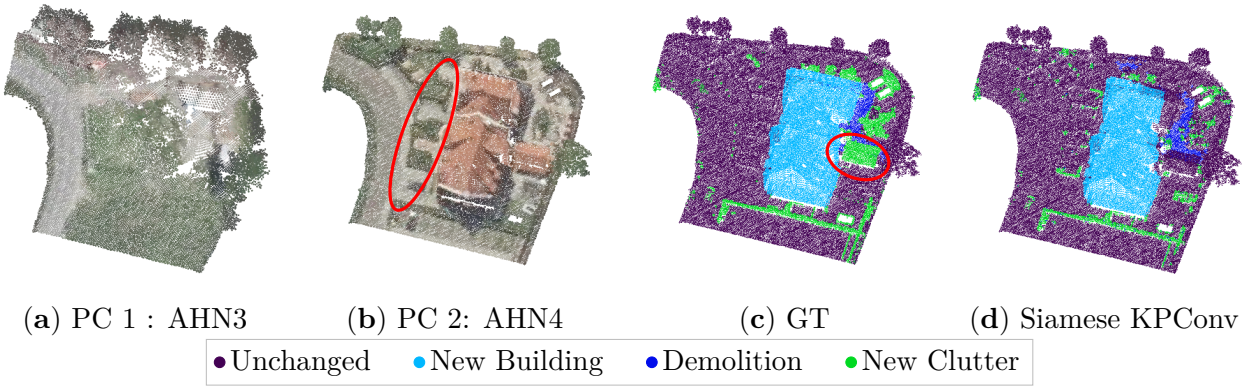
Method	Per class IoU (%)			
	Unchanged	New building	Demolition	New clutter
Siamese KPConv (ours)	**95.94** ± 0.06	**83.19** ± 1.54	**56.05** ± 1.74	**40.53** ± 0.56
Pseudo-Siamese KPConv (ours)	92.96 ± 1.34	76.54 ± 11.39	43.67 ± 1.88	36.76 ± 2.95
DSM-Siamese	88.58 ± 2.53	60.95 ± 5.54	18.04 ± 1.59	20.54 ± 3.59
DSM-Pseudo-Siamese	92.25 ± 0.11	73.26 ± 0.68	22.91 ± 1.82	28.02 ± 0.73
DSM-FC-EF	92.95 ± 1.49	74.21 ± 0.37	33.68 ± 6.84	26.32 ± 0.04
RF (Tran et al., 2018)	93.13 ± 0.00	70.50 ± 0.21	2.04 ± 0.04	13.27 ± 0.02

Table 3.8: **Per class IoU results on AHN-CD dataset.** DSM-based methods are adaptation of Daudt et al., 2018 networks to DSM inspired by Zhang et al., 2019 works.

3.3.3.2 Semantic change results on real dataset

Results on AHN-CD dataset are presented in Table 3.7 and 3.8. As in previous experiments, Siamese KPConv networks provide better results than other methods. A significant gap (around 31% of mIoU_{ch}) between our results and RF persists on this real dataset. Similarly to simulated datasets, the fully convolutional network with early fusion performs better than Siamese networks on DSMs, with lower scores, however, than our method. As can be seen in Figure 3.12, Pseudo-Siamese KPConv predictions (3.12d) are globally similar to the ground truth (3.12c).

As seen in Table 3.7 and 3.8, the scores of all methods are lower with AHN-CD than scores obtained on Urb3DCD-V2 datasets. Despite the fact that it might be more difficult to perform change detection and categorization on these real data, our results seem quite coherent with visible changes when comparing AHN3 and AHN4, as shown in Figure 3.12. In our opinion, the main difficulty comes from change annotation. Indeed, as shown in the quality assessment of AHN-CD in the Chapter 1 (Section 1.2.3), the ground truth is far from being perfect. In Figure 3.13, we illustrate the same problem in the ground



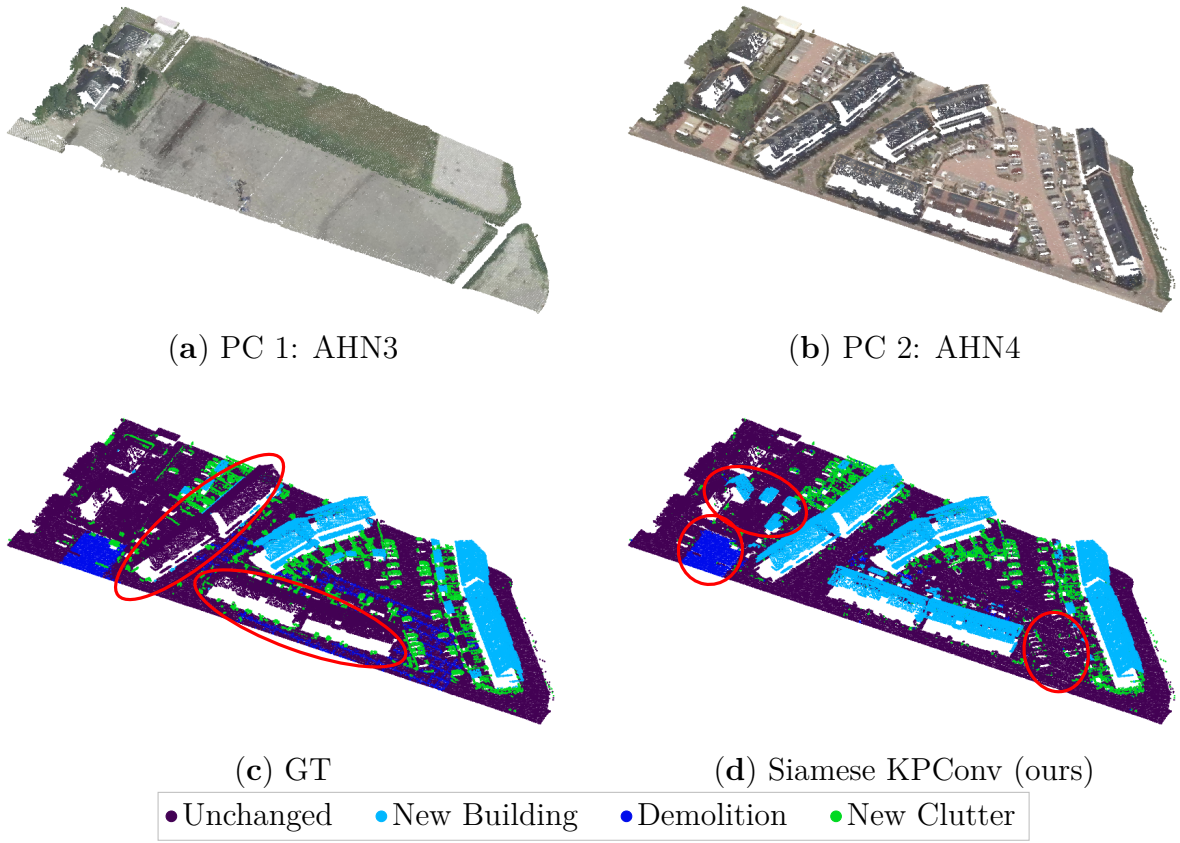
(**a**) PC 1 : AHN3 (**b**) PC 2: AHN4 (**c**) GT (**d**) Siamese KPConv

Figure 3.12: **Qualitative results on AHN-CD dataset.** See the discussion regarding the quality of the ground truth (GT). Regions of interest specifically discussed in the text are highlighted with ellipses.

truth as the one given in the Chapter 1 (Section 1.2.3). On this example, new buildings are omitted because of the presence of the glasshouse in AHN3 PC. As can be seen, this challenging situation is interesting because our method correctly predicted the majority of all new buildings. Similarly, when looking at the garden sheds visible in Figure 3.12 and 3.13, the ground truth marked it as a new clutter or unchanged whereas it is sometimes predicted as new building or even unchanged because of the glasshouse present in the older PC as explained before (Figure 3.13).

Another remark should be made on the demolition class. Indeed, this class is largely underrepresented: it contains only 0.2% of points in the training dataset whereas the 'unchanged', 'new building' and 'new clutter' classes represent 87.83%, 7.84% and 4.41% respectively. This undoubtedly explains the lower scores for demolition, even if we adapted the training stage to alleviate this issue. An example of demolition omitted by our network is visible on the ground replacing the demolished glasshouse of Figure 3.13d (see region of interest on the right side). However, this example might be a difficult situation since in the older PC, the glasshouse was mapped with both points of the ground and on its roof, since the LiDAR signal was partly reflected on the glass surface, and partly passing through it and reflected on the ground. Indeed, the demolition is well predicted in easier configurations such as in the left side of Figure 3.13.

Despite this imperfect annotation, we thought it was interesting to perform some tests on such real data. However, figures should be read with caution and analyzed in comparison to other methods. Nevertheless, let us point out that the visual results of our



(**a**) PC 1: AHN3 (**b**) PC 2: AHN4

(**c**) GT (**d**) Siamese KPConv (ours)

Figure 3.13: **Qualitative results on AHN-CD dataset, illustrating some ground truth (GT) errors contrasting with relevant prediction by our method.** Regions of interest specifically discussed in the text are highlighted with ellipses.

method seem very promising. In particular, the fact that our method provides results in some cases closer to reality than the ground truth, as seen in Figure 3.13, highlights the robustness against mislabeled data. Therefore, it would be interesting to possess a method capable of indicating the confidence level of the prediction, such as Bayesian deep learning methods. Indeed, it has been shown that some errors in the ground truth can be highlighted by looking at the confidence level (Dechesne et al., 2021).

Finally, we now give results of the sub-part of AHN-CD testing set that has been manually annotated, see Chapter 1 (Section 1.2.3) for more details. Results are given in Table 3.9 and 3.10. Again, our methods lead to better results than other state-of-the-art methods based on handcrafted features or DSM. In particular, scores are very satisfying on unchanged, new building and demolition classes. Concerning the new clutter class,

Method	mAcc	mIoU_{ch}
Siamese KPConv (ours)	85.65 ± 1.55	**72.95** ± 2.05
Pseudo-Siamese KPConv (ours)	**87.87** ± 1.89	69.33 ± 1.99
DSM-Siamese	50.87 ± 1.15	30.96 ± 2.48
DSM-Pseudo-Siamese	70.71 ± 5.09	48.85 ± 7.03
DSM-FC-EF	71.47 ± 1.43	45.57 ± 0.98
RF	47.94 ± 0.02	29.45 ± 0.02

Table 3.9: **General results** (given in %) **on the AHN-CD dataset sub-part that has been manually annotated.** DSM-based methods are adaptation of Daudt et al., 2018 networks to DSM inspired by Zhang et al., 2019 works.

Method	Per class IoU			
	Unchanged	New building	Demolition	New clutter
Siamese KPConv (ours)	**89.75** ± 2.18	82.77 ± 5.38	**86.44** ± 0.88	**46.65** ± 0.16
Pseudo-Siamese KPConv (ours)	88.90 ± 1.89	86.93 ± 5.32	84.01 ± 0.87	37.08 ± 2.85
DSM-Siamese	77.10 ± 1.51	76.77 ± 0.79	4.91 ± 8.33	11.20 ± 1.71
DSM-Pseudo-Siamese	78.00 ± 5.09	75.32 ± 8.59	47.46 ± 11.92	23.76 ± 0.56
DSM-FC-EF	70.77 ± 1.13	**90.32** ± 0.61	30.58 ± 1.76	15.81 ± 0.81
RF	78.24 ± 0.00	74.64 ± 0.03	0.00 ± 0.00	13.72 ± 0.06

Table 3.10: **Per class IoU results** (given in %) **on the AHN-CD dataset sub-part that has been manually annotated.** DSM-based methods are adaptation of Daudt et al., 2018 networks to DSM inspired by Zhang et al., 2019 works.

results are less impressive but still better than other methods. However, as stated before, this class is a mix of several types of objects. Notice that these results are obtained with the network trained on the AHN-CD dataset without manual correction of the ground truth. Hence, it demonstrates the robustness of our method to errors in the training database.

To improve change classification results, it would also be interesting to add RGB information or LiDAR intensity, available in the AHN data, as input to the network.

As for computation time, we report an inference time for Siamese KPConv of about 30 minutes in a single GPU computer (Nvidia Tesla V100 SXM2 16 GB) for cylinders of 25 m in radius in the testing area of Figure 1.6. The testing set corresponds to around 27,000 cylinders extracted from the pair of original PCs, i.e., a total of around 34 and 81 million of points for each PC respectively, resulting in about 9 millions points in each PC after the first sub-sampling step. The training stage takes about one day on AHN-CD dataset with 6,000 cylinders in the training set and 500 in the validation set.

Method	mAcc (%)	mIoU (%)
Siamese KPconv Cls (ours)	**88.75** ± 1.59	**80.30** ± 1.58
SiamGCN	76.45 ± 1.14	57.27 ± 0.52

Table 3.11: **Change classification results on Urb3DCD-Cls synthetic dataset.** Results are given in %. Veg. stands for vegetation.

Method	Per class IoU (%)				
	No change	New building	Demolition	New veg.	Veg. removed
Siamese KPconv Cls	**82.10** ± 0.98	**73.65** ± 1.56	**80.50** ± 1.60	**85.81** ± 1.64	**79.45** ± 2.87
SiamGCN	68.63 ± 0.97	61.43 ± 0.79	70.29 ± 1.08	38.31 ± 0.59	47.69 ± 0.92

Table 3.12: **Per class change classification results on Urb3DCD-Cls synthetic dataset.** Results are given in %. Veg. stands for vegetation.

3.3.3.3 Change classification results

Results regarding the change classification task for both synthetic and real datasets are presented in Tables 3.11, 3.12, 3.13 and 3.14, respectively. Our architecture is reaching some quite reliable results for each class of the Urb3DCD-Cls dataset. It also strongly outperforms SiamGCN. When looking at Table 3.14 for the Change3D dataset, results of Siamese KPConv Cls are still higher than other methods except for classes "no change" and "color change". Indeed, on these two classes, the hand-crafted PoChaDeHH method is performing better. As shown in Table 1.2 the "color change" class is underrepresented (3.24% of the training set), surely explaining lower scores of methods requiring a training phase (Siamese KPConv Cls, SiamGCN and HGI-CD). Furthermore, this class is the only one representing colorimetric changes instead of geometric ones. Even if it outperforms other methods on the "change" class, Siamese KPConv Cls leads to an unsatisfactory IoU score and has an important variation over different training runs. This class stands for slight changes in a remaining object, therefore the scale of change is different for this class compared to "removed" or "added" ones where the entire object changes, making the change detection task harder. Finally, when looking at global results (mAcc and mIoU), one can observe that our method outperforms state-of-the-art methods for the change classification task.

Method	mAcc	mIoU
Siamese KPconv Cls (ours)	**49.64** ± 1.35	**34.64** ± 1.18
PoChaDeHH	45.18	30.22
HGI-CD	25.82	17.17
SiamGCN	32.04 ± 6.49	19.18 ± 1.03

Table 3.13: **Change classification results on Change3D real dataset.** PoChaDeHH, HGI-CD, and SiamGCN have been introduced in Ku et al., 2021. For PoChaDeHH and HGI-CD, results are directly taken from the original publication. For SiamGCN, the public code has been used to retrain the model on a valid train/val/test split. Results are given in %.

	Per class IoU (%)				
Method	No change	New building	Demolition	New veg.	Missing veg.
Siamese KPconv Cls	55.35 ± 2.80	**43.41** ± 3.71	**47.93** ± 4.74	**19.85** ± 9.25	6.67 ± 11.55
PoChaDeHH	**61.06**	31.58	40.00	4.17	**14.29**
HGI-CD	55.30	16.28	14.29	0.00	0.00
SiamGCN	42.56 ± 1.78	24.33 ± 0.83	11.27 ± 3.07	14.00 ± 2.19	3.70 ± 4.94

Table 3.14: **Per class change classification results on Change3D real dataset.** PoChaDeHH, HGI-CD, and SiamGCN have been introduced in Ku et al., 2021. For PoChaDeHH and HGI-CD, results are directly taken from the original publication. For SiamGCN, the public code has been used to retrain the model on a valid train/val/test split. Results are given in %.

3.4 Beyond Siamese KPConv

While the results achieved by our Siamese KPConv are promising, especially w.r.t. the state-of-the-art, we believe there is still room for improvement. Thus, in this section, we explore two strategies to further enhance supervised deep learning for PC change detection: on the one hand, we evaluate the added-value of adding hand-crafted features as input along with 3D point coordinates; on the other hand, we design new deep architectures that better encode the change information between the two PCs.

3.4.1 Considering hand-crafted features

While a general trend in deep learning-based method is to give as input the raw data and to let the network combining inputs to obtain discriminating features, it appears interesting to also feed the network with some hand-crafted features designed with scene and data knowledge. Indeed, some specific hand-crafted features have been cautiously designed to perform classification, segmentation, or change detection tasks using machine learning frameworks. As an example, we recall here that the RF methods based on hand-crafted features outperform other distance-based methods on multiple or binary change detection into 3D PCs (see Chapter 2 Section 2.2.3.1). Thus, this raises a question: would giving hand-crafted features in addition to 3D point coordinates help or in the contrary disturb the deep network to extract informative features?

Even if not usual in deep learning, some studies showed that combining deep and hand-crafted features improves final results in computer vision (Nanni et al., 2017) or even in remote sensing (Nijhawan et al., 2019). Hsu and Zhuang (2020) also showed that incorporating hand-crafted features into a deep learning framework allows improving PCs semantic segmentation. In particular, they evaluate the benefice of giving some different types of features in addition to 3D points coordinates for PointNet (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b) 3D deep frameworks. It is shown that depending on the dataset (MLS or ALS), PointNet basic architecture can equalize or even outperform more complex architectures such as PointNet++ or KP-FCNN (Thomas et al., 2019) when input embeds hand-crafted features.

Therefore, in this section, we study whether adding hand-crafted features in Siamese KPConv deep network influences the change segmentation results. We extracted the same hand-crafted features as used in the Random Forest (RF) experiments. These features are selected according to the study of Tran et al. (2018). More specifically, chosen hand-

Method	# of input features		mAcc	mIoU$_{ch}$
Pseudo-Siamese KPConv	0		91.31 ± 2.34	77.80 ± 1.69
	10		**93.58** ± 0.68	**85.01** ± 0.53
Siamese KPConv	0		91.21 ± 0.68	80.12 ± 0.02
	10		**93.65** ± 0.16	**84.82** ± 0.58
	9	w/o *Stability*	91.44 ± 0.47	80.49 ± 0.64
	1	*Stability* only	92.92 ± 0.47	83.80 ± 0.89

Table 3.15: **Comparison of our architectures with different input features on Urb3DCD-V2 low density LiDAR dataset.** Results are given in %. The ten input features are: N_x, N_y, N_z, L_T, P_T, O_T, Z_{range}, Z_{rank}, nH and *Stability*.

crafted features provide information on point distribution (N_x, N_y, N_z, L_T, P_T, O_T), height (Z_{range}, Z_{rank}, nH) and change (*Stability*). These features are described more precisely in Chapter 2 (Section 2.2.1). We recall that Siamese KPConv architecture takes as many input features as desired, by simply modifying the number of inputs of the first layer of the encoders.

Quantitative results are given in Table 3.15. First, we can observe that providing as input hand-crafted features in addition to point coordinates considerably improves both Siamese KPConv and Pseudo-Siamese KPConv. Then, we assessed the importance of the unique change-related hand-crafted feature (*Stability*). As visible, it seems that point distribution and height hand-crafted features have only a slight beneficial impact (+0.37% of mIoU$_{ch}$) on change segmentation results. On the opposite, the *Stability* feature seems to have a major impact (+3.67% of mIoU$_{ch}$) on both metrics mAcc and mIoU$_{ch}$. More specifically, when looking at the per class gain in IoU, the *Stability* feature on its own principally helps for 'new building', 'demolition' and 'missing vegetation' classes (see Figure 3.14).

In conclusion, the Siamese KPConv network seems to be able to derive deep features related to points and their neighborhoods at a single date, indeed, providing features linked to point distribution and height does not bring any significant change on the results. However, even though looking at the results presented in Section 3.3.3, our architecture is able to recover the change on its own, it seems that giving a hand-crafted feature related to the change as input helps the network to focus on the change.

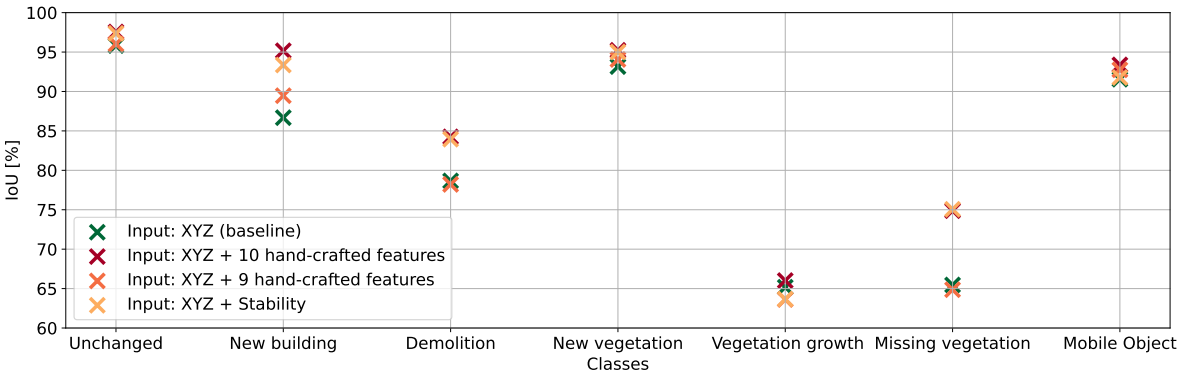


Figure 3.14: **Influence on per class IoU of adding hand-crafted features along with 3D point coordinates as input to Siamese KPConv.** For classes 'new building', 'demolition' and 'missing vegetation', the high disparity in IoU shows that adding hand-crafted features to the input has a greater influence than on classes where results are grouped around a same value.

3.4.2 Siamese KPConv architecture evolutions

Following the observed benefit of adding the *Stability* change-related hand-crafted feature as input along with 3D points coordinates, we explore how to learn this change information through novel deep networks. To do so, we build upon our Siamese KPConv model and propose three original architectures that emphasize change related features (e.g., the nearest point features difference in Siamese KPConv).

A first option is to lighten the network by fusing both PCs information just after the first layer, as illustrated in Figure 3.15. The following layer of the encoder takes as input only the nearest point features difference (noted $\ominus$). Then, for the following layers of the encoder and the decoder, they take as input the output of the previous layer as in a classical FCN. Here, the idea is to evaluate the benefits of dealing with differences earlier in the process. This architecture is named OneConvFusion.

However, mono-date features of the first layer might not be sufficient for accurate change identification. Therefore, we designed the Triplet KPConv network. It contains two encoders to extract mono-date information (as in the Siamese KPConv network) and an additional encoder whose goal is to extract change related features. The "change encoder" takes as input the nearest point difference computed after the first layer of mono-date encoders. Then, the following layers of the change encoder take as input the concatenation of the output features of the previous layer and the result of the nearest

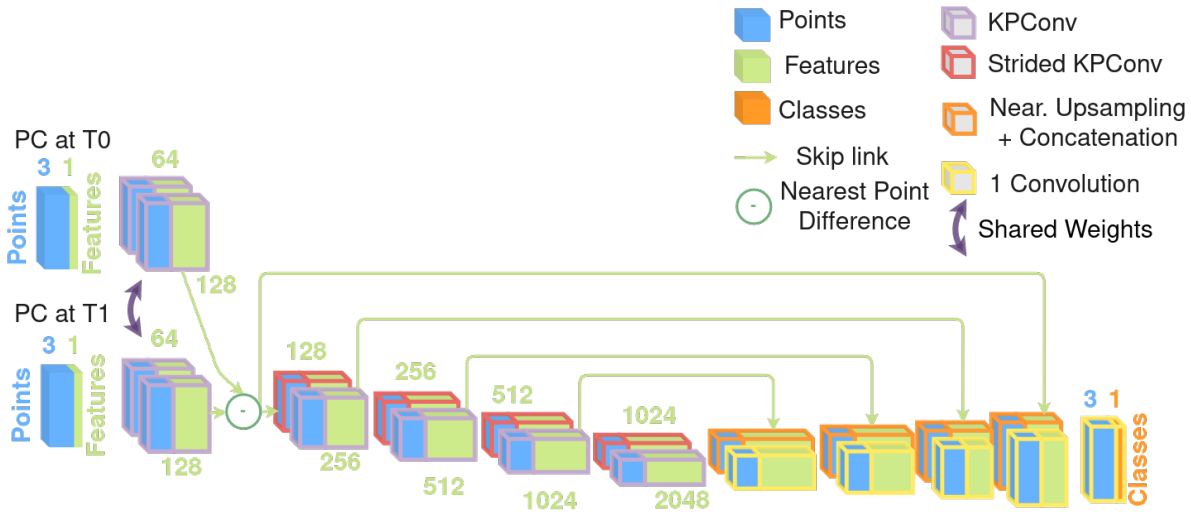


Figure 3.15: **OneConvFusion architecture for 3D PCs change segmentation.**

point features (from mono-date encoder) difference of the corresponding scale. Thereby, multi-scale mono-date information is taken into account as well as multi-scale change information. The decoder uses features extracted by the change encoder as input. Notice that mono-date encoders can share weights or not (leading to pseudo-Triplet KPConv), as for Siamese KPConv and Pseudo-Siamese KPConv. This network is shown in Figure 3.16.

The third version of the architecture is designed to directly fuse mono-date and change features in a same encoder. This network is called Encoder Fusion SiamKPConv. A first encoder extracts mono-date features of the older PC using convolution layers (top of Figure 3.17), as in all previous architectures. Then, as illustrated in the bottom of Figure 3.17, the second encoder is more specific to combine output features from the newer PC and the nearest point difference of features. In particular, each layer of this second encoder takes as input the concatenation of output features of the previous layer and the difference of features from this encoder and the mono-date encoder of the older PC. Thereby, both mono-date and change features can be combined in convolutional layers. As with Triplet KPConv and OneConvFusion architectures, the idea is to encode the differences earlier in the process, but here they are combined with features of the second PC.

These proposed new architectures are in line with recent developments for 2D image change detection concerning data fusion (see Section 3.1.2). Indeed, by convolving change features in the encoder, we expect that the network will put more attention on changes

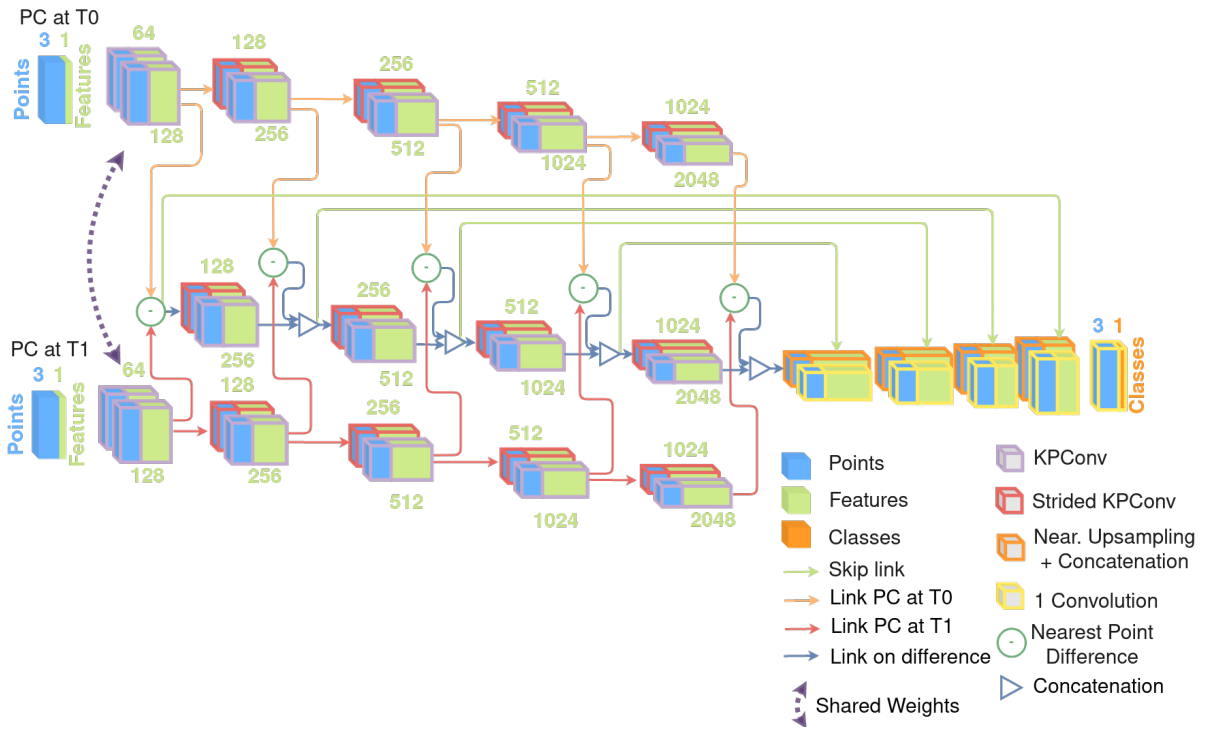


Figure 3.16: **Triplet KPConv architecture for 3D PCs change segmentation.**

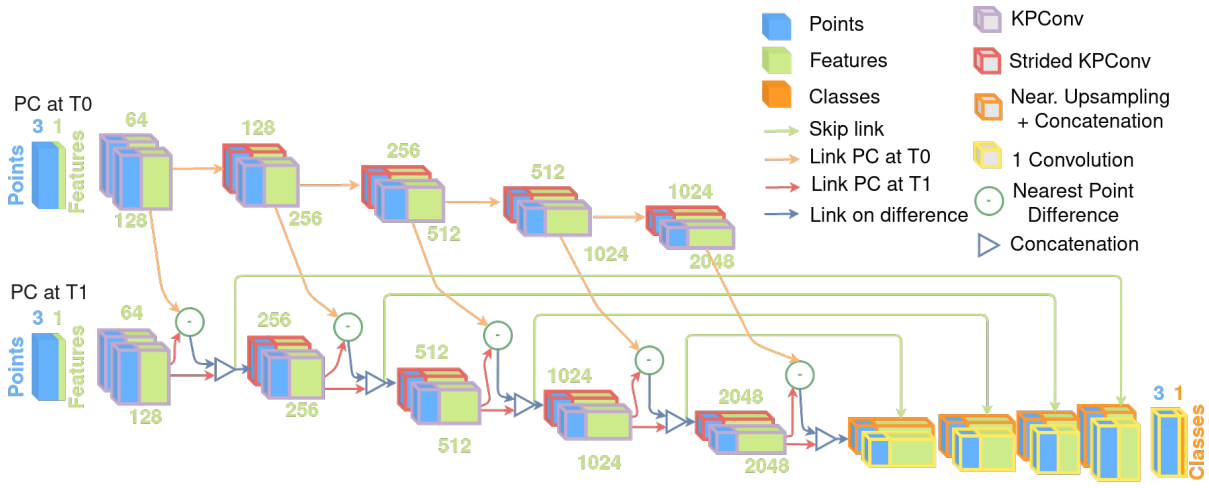


Figure 3.17: **Encoder Fusion SiamKPConv architecture for 3D PCs change segmentation.**

and also better combine multi-scale change features.

3.4.3 Experimental results

In the following sub-section, we evaluate the three architectures on Urb3DCD-V2 LiDAR low density dataset. The same training configurations are used as the one used for Siamese KPConv assessment, as described in Section 3.3.2. In particular, the same first subsampling rate (dl_0), cylinder radius, and learning rate are used.

Quantitative results of the evaluation of the three architectures are presented in Tables 3.16 and 3.17. It is worth noting that each of the three architectures outperforms Siamese KPConv network. In particular, the best architecture is Encoder Fusion SiamKPConv nearly followed by Triplet KPConv, while OneConvFusion is only slightly better (1.5% of mIoU_{ch}) than Siamese KPConv. When looking at per class results (Table 3.17 and Figure 3.18), Encoder Fusion SiamKPConv network provides a significant improvement for 'new building', 'demolition', 'new vegetation', 'missing vegetation' and 'vegetation growth' classes. Qualitative results are shown in Figures 3.19 and 3.20. As can be seen, the three architectures provide very similar results to the ground truth. In Figure 3.20, each of the three Siamese KPConv evolutions show results more accurate than Siamese KPConv in the new building facades. These facades are particularly hard to correctly detect, because in the first PC (Figure 3.20a), the neighbor facade was not visible. Thereby, identifying the new facade in the class 'new building' while neighboring facades are unchanged is not obvious. In particular, the network should understand that if the roof is new, the facade is probably new also. In the same way, if the roof has not changed, the facade also should be identical. Another difference with Siamese KPConv results is visible in Figure 3.19, where a part of the church roof is identified as new vegetation for Siamese KPConv while not for the other architectures. The misclassification is probably due to the shape of the dome roof that looks like a tree in simulated data. Indeed, even if tree models are not totally spherical (in particular the Arbraro software (Diestel, 2003) was used to obtain OBJ models of trees, see Chapter 1 Section 1.3.1), LiDAR simulation on these models render a quite spherical object with only a few points inside the foliage of the tree unlike real LiDAR acquisition. Therefore, aside from the shape, the main way to distinguish between the vegetation and the dome is that trees are generally on the ground. These examples, highlight the fact that the network should be able to understand the PC at multiple scales and predict changes with regard to surrounding objects.

Method	mAcc	mIoU_{ch}
Siamese KPConv	91.21 ± 0.68	80.12 ± 0.02
Siamese KPConv (+10 input features)	93.65 ± 0.16	84.82 ± 0.58
OneConvFusion	92.62 ± 1.10	81.74 ± 1.45
Triplet KPConv	92.94 ± 0.53	84.08 ± 1.20
Encoder Fusion SiamKPConv	**94.23** ± 0.88	**85.19** ± 0.24

Table 3.16: **General results in % of the three Siamese KPConv evolutions on Urb3DCD-V2 low density LiDAR dataset.**

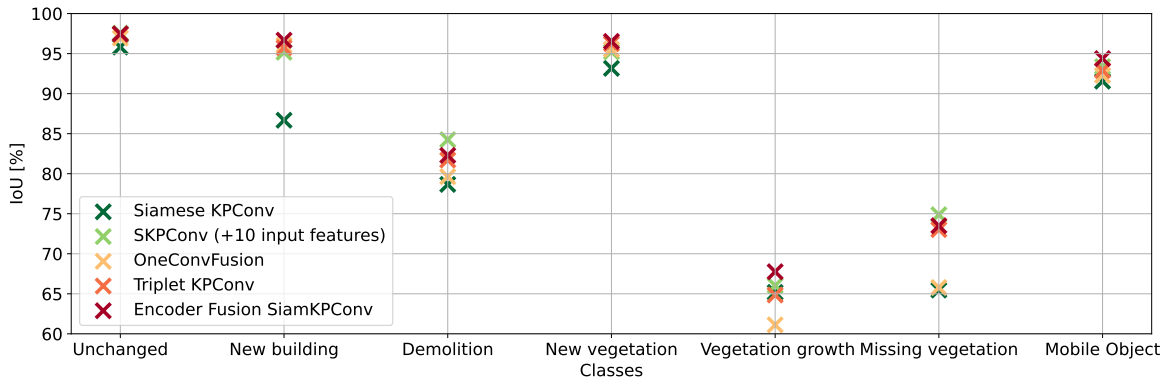


Figure 3.18: **Influence on per class IoU of the three Siamese KPConv evolutions**, namely OneConvFusion, Triplet KPConv and Encoder Fusion SiamKPConv. For comparison purpose, results of Siamese KPConv with 10 hand-crafted input features are also shown.

Method	Per class IoU						
	Unchanged	New building	Demolition	New veg.	Veg. growth	Missing veg.	Mobile Object
Siamese KPConv	95.82 ± 0.48	86.67 ± 0.47	78.66 ± 0.47	93.16 ± 0.27	65.18 ± 1.37	65.46 ± 0.93	91.55 ± 0.60
SKPConv (+10 input feat.)	**97.55** ± 0.11	95.17 ± 0.21	**84.25** ± 0.59	95.23 ± 0.21	66.02 ± 1.33	**74.88** ± 1.03	93.38 ± 0.74
OneConvFusion	96.95 ± 0.34	96.06 ± 0.27	79.63 ± 1.48	95.53 ± 0.77	61.12 ± 2.13	65.79 ± 2.61	92.89 ± 1.95
Triplet KPConv	97.41 ± 0.24	95.73 ± 0.67	81.71 ± 1.47	96.24 ± 0.37	64.85 ± 1.46	73.02 ± 1.18	92.90 ± 2.47
Encoder Fusion SKPConv	97.47 ± 0.04	**96.68** ± 0.30	82.29 ± 0.16	**96.52** ± 0.03	**67.76** ± 1.51	73.50 ± 0.81	**94.37** ± 0.54

Table 3.17: **Per-class IoU scores of the three *Siamese KPConv* evolutions on Urb3DCD-V2 low density LiDAR dataset.** Results are given in %. Veg. stands for vegetation; input feat. for input features; SKPConv for *Siamese KPConv*.

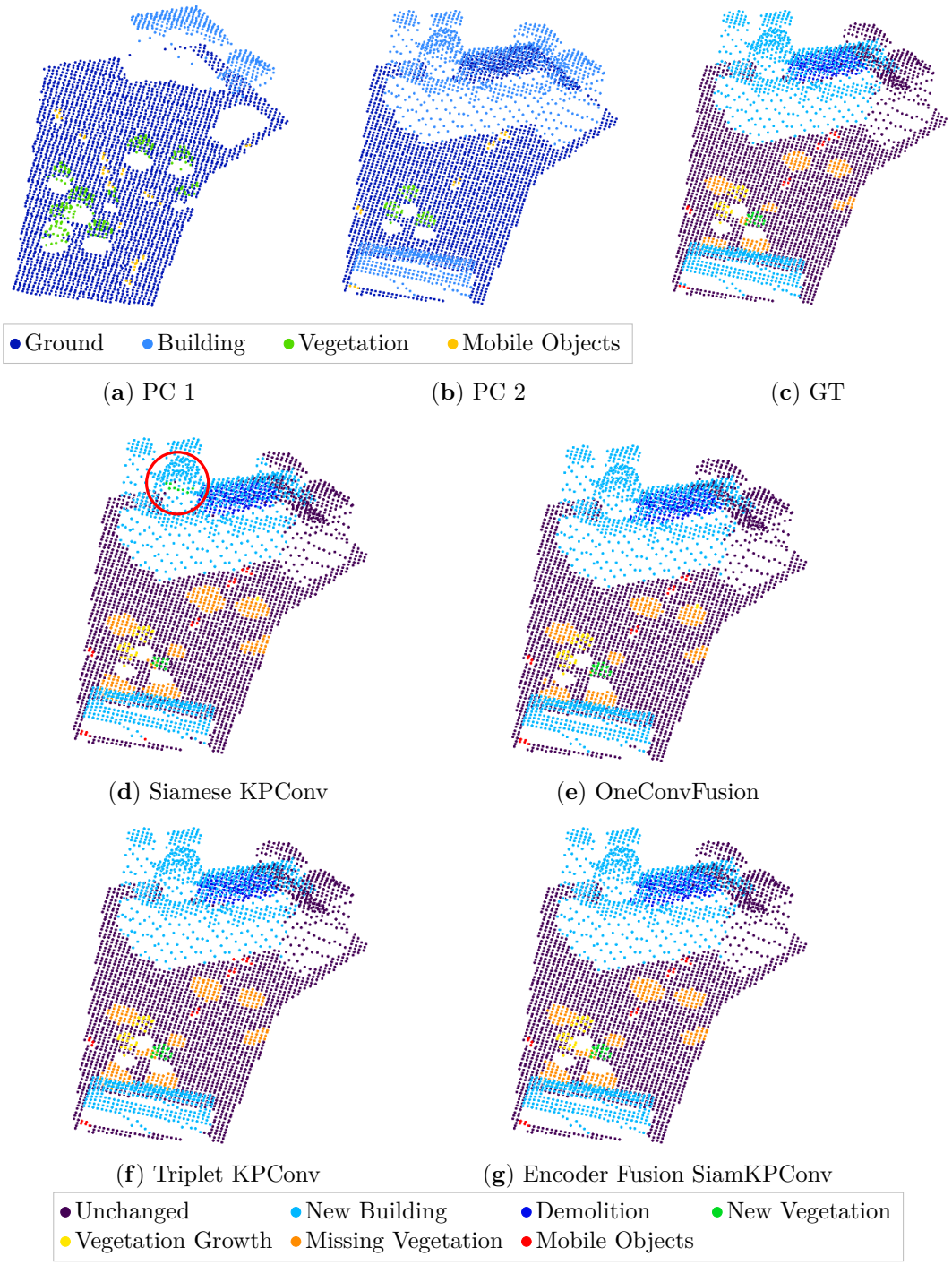


Figure 3.19: **Visual change detection results on Urb3DCD-V2 low density LiDAR sub-dataset:** (**a**-**b**) the two input point clouds; (**c**) ground truth (GT): simulated changes; (**d**) Siamese KPConv results; (**e**) OneConvFusion results; (**f**) Triplet KPConv results; (**g**) Encoder Fusion SiamKPConv results. Region of interest specifically discussed in the text is highlighted with an ellipse.

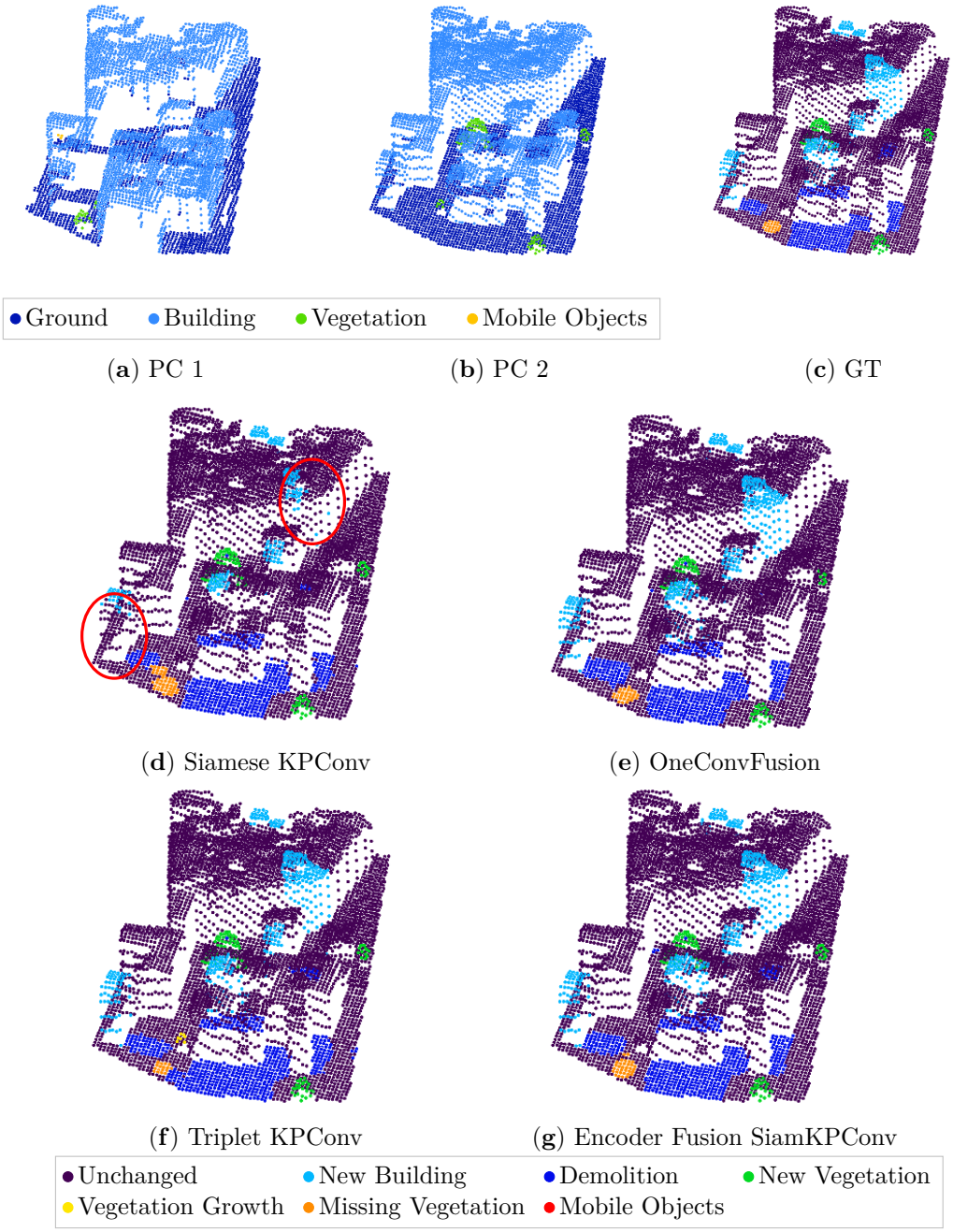


Figure 3.20: **Visual change detection results on Urb3DCD-V2 low density LiDAR sub-dataset in an area containing occlusions:** (**a**-**b**) the two input point clouds; (**c**) ground truth (GT): simulated changes; (**d**) Siamese KPConv results; (**e**) OneConvFusion results; (**f**) Triplet KPConv results; (**g**) Encoder Fusion SiamKPConv results. Regions of interest specifically discussed in the text are highlighted with ellipses.

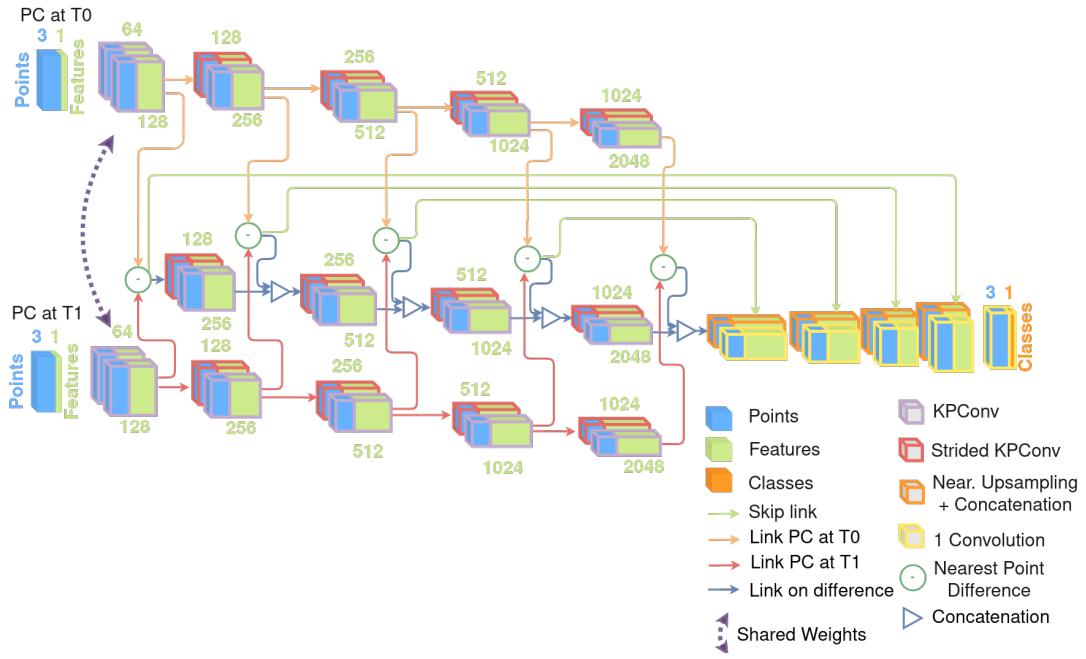


Figure 3.21: **Variant architecture for Triplet KPConv** (Figure 3.16) with skip connection only with the nearest point difference.

3.4.4 Discussion

Variant of Triplet KPConv and Encoder Fusion SiamKPConv networks

Table 3.18 and 3.19 present complementary results for Triplet KPConv and Encoder Fusion SiamKPConv architectures. In particular, we evaluate whether it is better to fuse in the decoder skip connections with the concatenation of features from the precedent layer and the nearest neighbor difference, as assessed in the previous results section, or only the nearest neighbor difference. Illustration of such variant of architectures are presented in Figures 3.21 and 3.22 for Triplet KPConv and Encoder Fusion SiamKPConv respectively. In these figures, skip connections provide only the nearest neighbor difference to the decoder part conversely to architectures in Figures 3.16 and 3.17. For Encoder Fusion SiamKPConv, there is no significant difference of results between the variants. However, a skip connection with only the difference of features worsens Triplet KPConv results of about 2.5% of mIou_{ch}. These results show that convolved difference features are important to fuse in the decoder for Triplet KPConv. We again emphasize the advantage of change-related features obtained by convolution over the simple difference of single-date features for change detection.

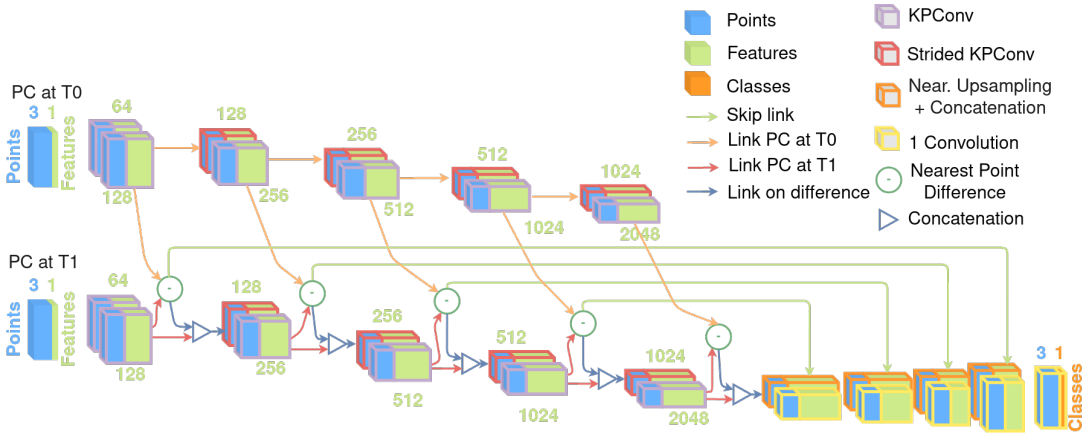


Figure 3.22: **Variant architecture for Encoder Fusion SiamKPConv** (Figure 3.17) with skip connection only with the nearest point difference.

Method	Skip co.	mAcc	mIoU_{ch}
Triplet KPConv	all	92.94 ± 0.53	84.08 ± 1.20
Triplet KPConv	diff.	93.15 ± 0.64	81.54 ± 0.67
Encoder Fusion SKPConv	all	**94.23** ± 0.88	85.19 ± 0.24
Encoder Fusion SKPConv	diff.	93.27 ± 0.58	85.36 ± 0.13
Encoder Fusion SKPConv (+10 input feat.)	all	94.13 ± 0.97	**85.87** ± 1.08
Encoder Fusion SKPConv (+10 input feat.)	diff.	93.80 ± 0.81	85.82 ± 1.10

Table 3.18: **Complementary results in % for Triplet KPConv and Encoder Fusion SiamKPConv on Urb3DCD-V2 low density LiDAR dataset.** For both architectures, we provide results with the difference concatenated with features from the previous layer (all), or, a variant where only the nearest point difference (diff.) are used as content of the skip connections (sk. co.). The first configuration is the one presented in Table 3.16. Results obtained with 10 hand-crafted features as input to the Encoder Fusion network are also provided. Feat. stands for features.

On the importance of learning change information

These results of Siamese KPConv architecture evolutions show the relevance of applying convolution also on the nearest point features difference at multiple scale to obtain change-related features. Lower results of OneConvFusion network exhibit that it is important to keep multi-scale mono-date features in the architecture. Then, the fact that Encoder Fusion SiamKPConv provides better results than the Triplet network shows that combining both mono-date semantic features and change features as input to convolutional layers can extract useful discriminative features for the change segmentation task. Both Triplet KPConv and Encoder Fusion SiamKPConv are closer to results on the benefit of adding hand-crafted features as input to the network. More specifically, Encoder fusion SiamKPConv gets better results than the Siamese KPConv with the 10 hand-crafted features. Finally, we tried to add hand-crafted features as input to Encoder Fusion SiamKPConv network, results are only very slightly improved (less than 1% of mIoU_{ch}, see Tables 3.18 and 3.19). This shows that an architecture more specifically designed for change detection is capable of extracting discriminative features on its own. This is especially true for change-related features such as the *Stability* which is no longer required in Encoder Fusion SiamKPConv architecture.

Method	S. co.	Per class IoU						
		Unchanged	New building	Demolition	New veg.	Veg. growth	Missing veg.	M. O.
Triplet KPConv	all	97.41 ± 0.24	95.73 ± 0.67	81.71 ± 1.47	96.24 ± 0.37	64.85 ± 1.46	73.02 ± 1.18	92.90 ± 2.47
Triplet KPConv	diff.	96.56 ± 0.14	93.67 ± 0.37	79.48 ± 0.97	91.48 ± 5.72	62.50 ± 2.65	65.48 ± 1.36	93.32 ± 1.27
EFSKPConv	all	97.47 ± 0.04	**96.68** ± 0.30	82.29 ± 0.16	**96.52** ± 0.03	67.76 ± 1.51	73.50 ± 0.81	**94.37** ± 0.54
EFSKPConv	diff.	97.63 ± 0.11	96.45 ± 0.19	83.18 ± 0.45	96.51 ± 0.35	68.06 ± 2.02	74.13 ± 1.66	93.83 ± 0.18
EFSKPConv (i. f.)	all	**97.84** ± 0.14	96.53 ± 0.24	**85.64** ± 0.11	96.23 ± 0.52	**68.20** ± 1.35	76.30 ± 1.77	92.33 ± 2.63
EFSKPConv (i. f.)	diff.	97.79 ± 0.14	95.99 ± 0.61	85.38 ± 0.60	96.08 ± 0.07	65.50 ± 3.14	**76.86** ± 2.04	93.11 ± 1.83

Table 3.19: **Per-class complementary IoU scores for the *Triplet KPConv* and the *Encoder Fusion SiamKPConv* networks on Urb3DCD-V2 low density LiDAR dataset.** For both architectures, we provide results with the difference concatenated with features from the previous layer (all), or a variant where only the nearest point difference (diff.) are used as content of the skip connections (sk. co.). The first configuration is the one presented in Table 3.17. Results obtained with 10 hand-crafted features as input to the *Encoder Fusion SiamKPConv* network are also provided. Results are given in %. Veg. stands for vegetation; M.O. for Mobile Object; i. f. for input features; EFSKPConv for Encoder Fusion SiamKPConv; S. co. for skip connection; Diff. for difference.

3.5 Conclusion

In this chapter, we have presented an **original deep neural network**, called **Siamese KPConv**, dedicated to change detection and categorization on 3D point clouds. We build upon successful deep components such as a Siamese network and Kernel Point Convolution to elaborate, to our knowledge, the first deep network able to **cope with pairs of raw 3D point clouds and perform change segmentation task**. We conducted **various experiments in an urban environment using synthetic** (Urb3DCD-V2) and **real** (AHN-CD) **datasets**. For each dataset, **our technique outperforms the state-of-the-art with a significant margin**, around 30% of mean IoU over classes of change. Since the best existing method before our Siamese KPConv relies on traditional machine learning algorithm trained on handcrafted features, as reported in Chapter 2, and there is no deep learning method dealing with change detection and categorization over raw 3D PCs, we have also been inspired by the literature to provide as baselines two different networks (a Siamese and a Fully-Connected network with early fusion) on 2D rasterization of PCs (DSMs). Our method consistently leads to significant improvement, between 15% to 30% in mean of IoU over classes of change when compared with the best deep baseline.

Then, we further analyzed the **benefit of adding hand-crafted features as input to the network along with 3D points coordinates**. This significantly improves change segmentation results (around 4.5% of mean IoU over classes of change). More specifically, the addition of a change-related feature plays a great role in the enhancement of results. Following this study, **we designed three other architectures** for change segmentation (**OneConvFusion**, **Triplet KPConv** and **Encoder Fusion SiamKPConv**) **to emphasize change-related deep features**. All **the three get better results than Siamese KPConv** (from 1.5% to 5% of mean IoU over classes of change). The Encoder Fusion SiamKPConv network, with a specific encoder that fuses change and mono-date features, allows getting rid of the addition of hand-crafted features. This emphasizes the **importance of applying convolution also on features difference**.

Furthermore, we proposed an adapted version of **Siamese KPConv for the change classification task that outperforms state-of-the-art methods including deep learning based networks on both synthetic and real** (Change3D) **datasets**.

Finally, this chapter showed the interest of developing methods directly processing 3D PCs without any rasterization as well as the advantage of using a deep learning-based method over traditional methods with hand-crafted features in a supervised context.

However, and even more when it comes to deep learning, annotations are crucial and moreover difficult to obtain. Thus, in the next chapter, we will see strategies to perform change segmentation without any annotation or with only a few annotations from real datasets.

Chapter 4

Unsupervised change detection

Contents

In the previous chapter, we showed that deep supervised learning can successfully tackle the 3D PCs change segmentation task thanks to our proposed method *Siamese KPConv*. However, the training process needs a large amount of labeled data. In a 3D PCs change segmentation supervised context, training data signifies annotated data at point level. With regard to the difficulty of obtaining 3D point scale change annotations discussed in Chapter 1, the question of the benefits of the deep learning paradigm when no or only a few annotations are available is open. This is addressed in this chapter.

We saw in Chapter 2 that deep learning based methods seem to better transfer knowledge from one source dataset to a target dataset compared to traditional machine learning ones (e.g., Random Forest (RF) algorithm). Thereby, one first approach to perform 3D change segmentation without or with only few labels from the target dataset would be to transfer learning from a labeled source dataset (e.g., our simulated dataset) to the unlabeled target dataset. This is described in Section 4.2.

In Section 4.3, unsupervised binary change segmentation is tackled by comparing deep features from both PCs. To extract discriminating features, two different strategies are tested to train a deep network to effectively characterize the target area. These two strategies relies on transfer learning and self-supervised learning[17]

Finally, in Section 4.4 multiple change segmentation is tackled. Instead of training a network for semantic segmentation and then comparing extracted features to highlight changes, the network is directly trained to predict changes thanks to recent advances on deep clustering[18].

4.1 Related work

Because of the difficulty of getting annotated data, the recent years have seen a growing interest over unsupervised or low-supervised deep learning methods for 2D image understanding (Ericsson et al., 2022; Wang et al., 2022; Berg et al., 2022; Yang et al., 2022) as well as for 3D PCs data (Xiao et al., 2023). More generally, by unsupervised methods, we mean any method that does not rely on human annotation for learning, unlike supervised approaches. We give here a general overview of existing unsupervised learning techniques

17. These methods have been developed in the frame of a mobility in the AI4EO Future lab (Technical University of Munich). A paper has been subsequently submitted for publication in a journal and a preprint version is available (de Gélis et al., 2023e).

18. This method has been submitted for publication in a journal and a preprint version is available (de Gélis et al., 2023c)

for 3D PCs representation learning, as well as for 2D image change detection.

4.1.1 Unsupervised representation learning for 3D point clouds

Most of 3D unsupervised representation learning methods consist in finding pre-text tasks that do not require some human annotation in order to pre-train a network to extract some discriminating features. These unsupervised methods, based on auxiliary tasks, are also called Self-Supervised Learning (SSL) methods. Then, to tackle a downstream task (i.e., the final task such as object classification, semantic segmentation, ...), the pre-trained network is usually fine-tuned with a few labels from the downstream task. As in 2D computer vision (Jing and Tian, 2020), unsupervised representation learning methods for 3D PCs can mainly be divided into two main categories: generation-based and context-based as described in Xiao et al. (2023).

Concerning **generative methods**, as for 2D image processing, we can distinguish between auto-encoders (AEs) and Generative Adversarial Networks (GANs). AEs encode inputs into representation vectors and decode them back to the original input data, thereby the ground truth is the input data itself (Kramer, 1991). This enables to pre-train the encoding part of a network without any labeled data. This typical architecture has been adapted to 3D PC data by using specific:

- losses: point permutation invariant losses (Fan et al., 2017) such as Chamfer distance or Earth Mover's distance;
- representations for 3D PCs: sparse 3D points (Zhao et al., 2019b; Yang et al., 2021), graphs (Yang et al., 2018; Chen et al., 2019b; Gao et al., 2020; Chen et al., 2021a), 3D voxels (Girdhar et al., 2016; Sharma et al., 2016; Hess et al., 2023);
- strategies: by exploring different scales in PCs (Liu et al., 2019) or multiple resolutions (Gadelha et al., 2018; Chen et al., 2021a; Yang et al., 2021), by synthesizing the reconstruction problem in the decoder by the deformation of a 2D grid (Yang et al., 2018; Chen et al., 2021a), by modeling the self-reconstruction problem as a point distribution learning task (Yang et al., 2019; Shi et al., 2020b; Sun et al., 2020), ...

On the other hand, 3D PCs GANs versions have also been proposed. A GAN consists in a generator trained to fool a discriminator by generating data samples as realistic as possible. The role of the discriminator is to distinguish between real and synthesized

data samples (Goodfellow et al., 2020). As far as 3D PCs data are concerned, some 3D voxel-based (Wu et al., 2016), graph-based (Valsesia et al., 2018) and raw 3D PC-based (Achlioptas et al., 2018; Li et al., 2018a) GANs have been developed. Furthermore, generative pre-text task is also often derived in up-sampling task (e.g., via AEs (Remelli et al., 2019) or GANs (Li et al., 2019a)) or PCs completion task (e.g., reconstruction of masked PCs part (Yuan et al., 2018; Wen et al., 2020; Wang et al., 2020a; Mittal et al., 2021)). Finally, even if 3D PCs data generation is well studied through AE or GAN adaptations, it is often not suitable for real large remote sensing PCs. Indeed, most of the state-of-the-art only focuses on synthetic PC objects available in ModelNet (Wu et al., 2015) or ShapeNet (Chang et al., 2015) datasets (Xiao et al., 2023).

The second category of unsupervised methods for representation learning is based on **context information** learning. Here, different pre-text tasks (similarly to 2D) can be designed to learn spatial context structures of PCs such as 3D Jigsaw (Sauder and Sievers, 2019; Alliegro et al., 2021), rotation angle prediction (Poursaeed et al., 2020), 3D distortion recognition (Chen et al., 2021c), etc. Others approaches consist in learning context similarities through contrastive learning. Recently, contrastive learning has emerged as an attractive branch of self-supervised learning in computer vision in general. This involves forcing a network to predict similar features for data belonging to the same class, and dissimilar features for data from different classes. The main question in contrastive learning is how to collect multiple instances of similar and dissimilar data, respectively called positive and negative samples. A first set of methods relies on augmented views of a same object to get the positive example. On the contrary, negative samples are taken from the rest of the dataset. Different studies in the literature use such a data augmentation approach to generate positive views (Sanghi, 2020; Wang et al., 2021). However, these studies are designed for 3D PC objects and not for scene-level experiments. Focusing more on scene-level, Xie et al. (2020) proposed PointContrast which relies on different views (partial scans) of a scene to generate positive and negative point samples by using a nearest neighbor mapping to match the points together. Because of its efficiency on various downstream tasks (classification, semantic segmentation and object detection), PointConstrast is the basis of many other works. For example, Hou et al. (2021) start from PointContrast ideas and introduce some spatial context. In DepthContrast (Zhang et al., 2021c), authors further improve PointContrast idea by proposing a solution with single view data thanks to data augmentation. To overcome the constraints of data augmentation and multi-view data, Mei et al. (2022b) propose to rely on clustering principle.

In the category of learning context similarity, clustering approach consists in grouping similar data thanks to traditional clustering algorithm such as k-means. Further notice that most of the time clustering principle is used along with other pre-text tasks such as contrastive learning (Zhang and Zhu, 2019; Liang et al., 2021; Huang et al., 2022a; Mei et al., 2022a) or self-reconstruction (Hassani and Haley, 2019). Finally, a last set of methods learns context from temporal data such as Red Green Blue – Depth (RGB-D) video or LiDAR sequential data by using contrastive learning (Chen et al., 2022b) or by trying to minimize the mean squared errors between features of two PCs from the same temporal sequence (Huang et al., 2021). The latter is an of *bootstrap your own latent (BYOL)* concept (Grill et al., 2020) to 3D PCs data. Notice that these temporal 3D data are far from 3D PCs data used for change detection in remote sensing as they consist in continuous data frames. Even if there are some scene-level studies in the aforementioned works, none of them are applied on large scale remote sensing PCs. To the best of our knowledge, the study in Zhang et al. (2021a) is the only one to propose an unsupervised learning approach for terrain classification in ALS data. It is entirely based on deep clustering principle on 3D voxelisation of PCs (Zhang et al., 2021a).

Aside from these two main categories, two additional groups of methods can be outlined even though they are less representative of the state-of-the-art. Some other pre-text tasks lie in **learning local descriptors**. For example, Tang et al. (2020) propose to retrieve geometric properties such as the normal in the self-supervision stage. Some other studies rely on rotation invariant local descriptor (Deng et al., 2018) or 3D PCs registration (Jiang et al., 2021).

Finally, **multi-modal** based methods combine 3D PCs with other sources of information such as 2D views of the 3D PC (Jing et al., 2021; Afham et al., 2022) or even text (Chen et al., 2020).

In this section, we saw that a consequent number of studies have been conducted recently to learn unsupervised representation of 3D data. However, to the best of our knowledge, in this unsupervised context, 3D PCs change detection task has not been tackled yet.

4.1.2 2D change detection with low supervised learning

We saw in Chapter 3 (Section 3.1.2) that numerous studies propose specific deep learning-based methods for supervised change detection. Obtaining annotated ground truth is also critical for 2D change detection. Considering the difficulty of data annotation, numerous

deep frameworks have been designed to address the change detection task in an unsupervised way. As highlighted in Shi et al. (2020a), a first category of methods is based on the generation of credible change *pseudo-labels* to train a deep model. Generation of training data relies on various ideas such as combination of unsupervised traditional methods (e.g., Change Vector Analysis (CVA)) (Song et al., 2018; Li et al., 2021; Seydi and Hasanlou, 2021), fuzzy clustering (Gao et al., 2016; Zhan et al., 2018; Zhang et al., 2021b), metric learning (Zhao et al., 2019a) or even unsupervised deep framework, e.g., AEs (Gong et al., 2017) or GANs combined with metric learning (Tang et al., 2021). To counter the class imbalance problem (i.e., changed areas are in minority compared to unchanged ones), additional GANs can be used to enrich changed class pseudo-labels (Zhang et al., 2021b). In general, pixels are classified into three categories: changed, unchanged and uncertain. Only certain pixels are taken into account for the loss computation. Even if it has been shown that final change maps predicted by the deep network are more accurate than the original pseudo-change classification, these methods may somehow be limited by the pseudo-label quality.

Therefore, a second category of methods is based on latent change map generated by deep features. Transfer learning is a common strategy to train the deep model to extract useful features (Saha et al., 2019). However, transfer learning still requires the availability of an annotated source dataset. Therefore, fully unsupervised networks such as AEs (Lv et al., 2018; Bergamasco et al., 2019; Kalinicheva et al., 2019; Touati et al., 2020b; Zheng et al., 2021) or GANs (Niu et al., 2018) are used as well. Self-supervised learning strategies have shown great success recently, including for change detection task. In Saha et al. (2021), the authors take the advantage of the underrepresentation of changed areas and of the multi-sensor configuration to force the network to learn similar features in patches from the same spatial location and different features for two random patches through a contrastive loss. Contrastive learning is also used at super-pixel level (Chen and Bruzzone, 2022) or to separate features from similar and dissimilar patches generated using an unsupervised image segmentation algorithm (Cai et al., 2021). Leenstra et al. (2021) experimented two different pre-text tasks: overlapping and non-overlapping patches discrimination, and minimizing the difference between overlapping patches in the feature space. Notice that the second task seems to bring better change detection results, this is in line with the work of Saha et al. (2021). Dong et al. (2020) make use of the discriminator of a GAN trained to differentiate samples from bi-temporal images. When image time series are available, the prediction of the natural order of images seems to be a

suitable pre-text task for change detection (Saha et al., 2020). Pre-trained models can also be used to generate latent features further transformed in the final change map. Building upon this idea, Saha et al. (2019) propose to adapt the well-known CVA algorithm (Malila, 1980) to deep latent features with Deep Change Vector Analysis (DCVA) method. A deep change magnitude coefficient is computed for each pixel from automatically selected deep features. These pixel-wise coefficients, named the latent change map, are then converted to the final change map through thresholding. Let us also outline that in the literature, different other strategies are experimented to generate the latent change map using features similarity analysis (Zhang et al., 2016; Chen and Bruzzone, 2022), slow features analysis (Du et al., 2019), features distance combined with mutual information metric (Zheng et al., 2021), multi-scale feature map fusion (Li et al., 2022). Thresholding operation is very common to obtain the final change map (Liu et al., 2016; Du et al., 2019; Chen and Bruzzone, 2022; Zheng et al., 2021), but clustering is also used for binary (Zhang et al., 2016; Lv et al., 2018; Touati et al., 2020b) or multi-class change identification (Wu et al., 2021).

4.2 Transfer learning

A first strategy to reduce the need for annotated training data is to use transfer learning. Indeed, by training (and validating) on another dataset containing annotation (called the source dataset), the method might be able to also detect changes on another dataset (called the target dataset). However, because source and target data may be of different nature, a fine-tuning, e.g., retraining of the whole or at least of some layers of the pre-trained network, can be used to adapt more the model to the target dataset specificities (characteristics and/or classes). In the following section, we aim at assessing the transfer capacity of *Siamese KPConv* network, from simulated to real datasets. The goal is to explore the ability of a model trained on a specific dataset to generalize data of various types. In this section, we will assess the transfer learning capacities of our previous models, *Siamese KPConv* and *Pseudo-Siamese KPConv* (defined in Chapter 3 Section 3.2). In particular, two different experiments are proposed: transfer from the Multi-Sensor (MS) to the low-density LiDAR simulated sub-dataset without fine-tuning, and transfer from the simulated dataset to the real AHN-CD dataset with a fine-tuning using only small amounts of training data.

Method	$mIoU_{ch}$	Per class IoU (%)						
		Unch.	New build.	Demol.	New veg.	Veg. growth	Miss. veg.	M.O.
P-Siam KPConv	**59.10**	**92.91**	69.73	**63.71**	**40.88**	**35.80**	**65.69**	**78.79**
DSM-Siamese	37.07	92.08	**74.61**	54.67	39.41	0.43	38.05	15.25
DSM-P-Siam	35.77	91.55	69.36	56.02	36.30	4.76	30.11	17.94
DSM-FC-EF	42.01	92.87	67.11	55.63	33.41	1.14	39.10	29.72
RF	14.48	87.74	54.03	21.91	8.24	0.47	0.02	2.19

Table 4.1: **Transfer learning tests with training on the Urb3DCD-V2 MS sub-dataset and testing on the Urb3DCD-V2 low-density LiDAR dataset.** DSM-based methods are adaptation of Daudt et al. (2018) networks to DSM inspired by Zhang et al. (2019) works. Results are given in %. Unch., build., demol., veg., miss. and M.O. stand for unchanged, building, demolition, vegetation, missing, and mobile object respectively. P-Siam stands for Pseudo-Siamese.

4.2.1 Transfer learning assessment of *Siamese KPConv* network

In Table 4.1, we report the transfer results between a training on Urb3DCD-V2 MS sub-dataset and a test on the low-density LiDAR sub-dataset. Notice that no retraining has been necessary to adapt to the other dataset (since the classes are identical). As expected, results are worse than when the training is performed on a training set containing the same types of PCs as the testing set. However, our Pseudo-Siamese KPConv still gives better results than other methods when observing change classes corresponding to $mIoU_{ch}$. Notice that the generalization capacity is not the same according to the classes. Indeed, low scores are obtained on new vegetation or vegetation growth, whereas missing vegetation obtains very similar results to the without-transfer method. We have not included our *Siamese KPConv* in this comparison since its training on the MS sub-dataset is not reliable (see Table 3.4) and therefore the pre-trained network would lead to non-reliable features. One can note that scores obtained with Pseudo-Siamese KPConv trained on the MS dataset are slightly higher than those obtained when training an RF algorithm directly on the low-density LiDAR dataset. In particular, it allows us to obtain more reliable results than the RF method without transfer for 'unchanged', 'vegetation growth', 'missing vegetation' and 'mobile objects' classes (see Table 3.5). Table 4.1 recalls the poor generalization capacity of the RF method, even though it requires a smaller training set than deep learning methods (see experiments on training set size in Chapter 2, Table 2.4).

4.2.2 Transfer from simulated to real data in a weakly supervised context

The issue of the size of the training set is crucial since automatic data annotation is tricky (see Chapter 1, Section 1.2.3) and manual annotation is time-consuming.

To deal with this issue, an idea would be to pre-train a network on simulated data and then to fine-tune it on a few examples of real data. In order to assess the behavior of our network in such a small training dataset configuration, we trained the network from scratch with different sizes of training sets (symbolized by the number of cylinders given as input) and we compared results with the network pre-trained on a simulated dataset and fine-tuned on real data. The results are depicted in Figure 4.1. For these experiments, input cylinders are randomly chosen among the whole training set according to the class balance before the training, conversely to results shown in Chapter 3 (Section 3.3.3.1), where, for each training epoch, 6,000 cylinders are chosen randomly in the training set according to class balance (see Chapter 3, Section 3.3.2). Chosen cylinders are the same for both training from scratch and transfer learning tests. Notice that classes from Urb3DCD-V2 and AHN-CD are not the same. Therefore, we initialized weights with those issued from the Urb3DCD-V2 pre-training except for the last layer of the network, which gives the final label. This last layer is initialized randomly, as for the whole network when trained from scratch. Pre-trained weights are taken from *Siamese KPConv* with shared weight configuration trained on the sub-dataset Urb3DCD-V2-1 (low-density LiDAR), with input cylinders of 50 m in radius ($dl_0 = 1$ m). Even if the results are slightly higher when weights are not shared, the shared weights configuration provides better generalization capacities according to our experimental observations. Based on this figure, we can make several observations. The proposed fine-tuning strategy allows us to reduce the number of cylinders to 100, to achieve the same score. It should be noticed that our fine-tuning is straightforward, and one could expect better results using domain adaptation or meta-learning (Rußwurm et al., 2020). We have also observed that using more than 100 training cylinders did not improve the results further. This is due to an overfitting situation faced by our training procedure since we do not consider a random drawing for cylinders selection at each epoch, conversely to the process proposed by Thomas et al. (2019) that requires up to 36,000 cylinders in total (considering 60 epochs) and that was followed in Chapter 3. Improving the simulator to generate data closer to real data (in terms of resolution, noise, and classes) would definitely help, but this requires knowing

the target data in advance, which is not realistic in all use cases.

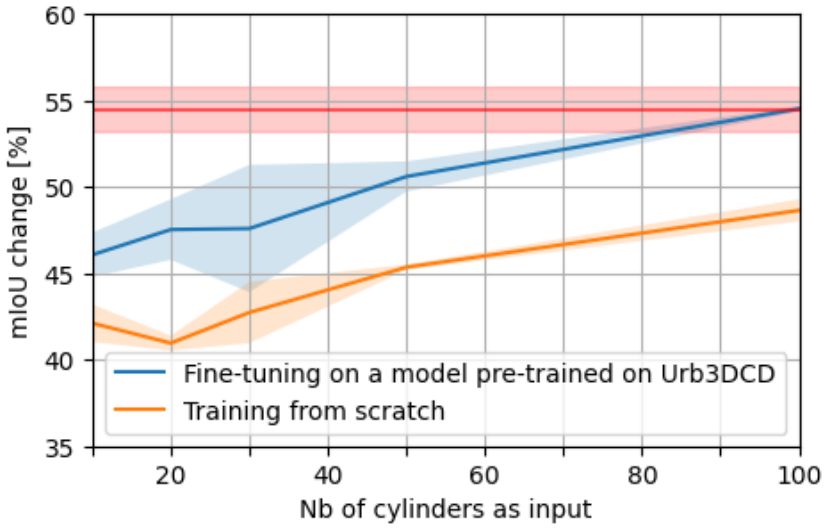


Figure 4.1: **Comparison between training from scratch and using pre-trained weights learned a simulated dataset of *Siamese KPConv*.** The mean of IoU over classes of change is given as a function of the number of cylinders of 50 m in diameter given as input. In red, the best results obtained with *Siamese KPConv* trained from scratch over 6,000 cylinders with random drawing.

4.3 Unsupervised binary change detection with deep change vector analysis

In the following section, we propose unsupervised methods for binary change detection in raw 3D PCs. These methods are inspired by state-of-the-art in unsupervised 2D images change detection methods.

4.3.1 Methodology

Our proposed method is fully unsupervised and is composed of two major steps, as described in Figure 4.2. The first one consists in extracting deep features that will be compared in the second step to extract changes. In the first stage, a network is trained to segment each PC individually into two different strategies following recent works in unsupervised 2D change detection (Saha et al., 2019; Saha et al., 2021). Sub-sections 4.3.1.1 and 4.3.1.2 focus on the training of the deep feature extractor. In this study, to adapt such a framework to 3D PCs, we use the Kernel Point – Fully Convolutional Neural Network (KP-FCNN) (Thomas et al., 2019) as the back-bone for the deep feature extraction part. Detailed descriptions of KP-FCNN and Kernel Point Convolution (KPConv) composing this architecture are available in Chapter 3 (Section 3.1.1). The third sub-section introduces how we adapt DCVA to 3D PC change detection.

We will denote $\mathcal{P}$ a PC and $\mathcal{F}^l$ its associate features at the layer $l \in \{0 \dots L\}$ of the network symbolized by $f_{\text{KP-FCNN}}$. The index 1 (resp. 2) corresponds to the older PC noted $\mathcal{P}_1$ (resp. newer PC noted $\mathcal{P}_2$) and N denotes the number of points p in the PC $\mathcal{P}$.

4.3.1.1 Training deep feature extraction: annex task

A first option to train the feature extraction network is to rely on annex tasks such as semantic segmentation using labels available in a public dataset.

Indeed, while datasets annotated according to the change are not common when dealing with 3D PCs, public datasets with a mono-date semantic annotation in urban environment are widely spread (Hackel et al., 2017; Roynard et al., 2018; Varney et al., 2020; Kölle et al., 2021). By choosing a public dataset as close as possible to the unlabeled change detection dataset to perform supervised training of the network, one could expect that extracted features will be consistent in unchanged areas and different in the case of changes. In our study, we use the H3D ALS to train the network for semantic segmen-

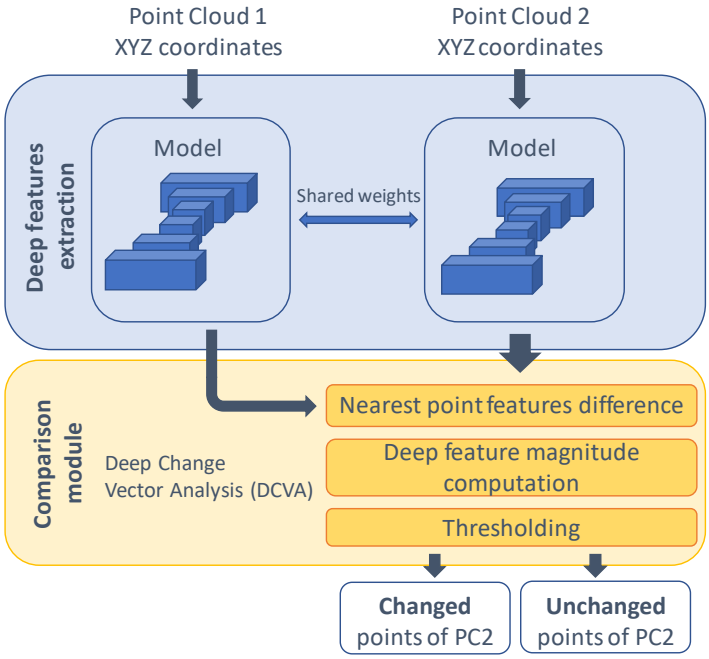


Figure 4.2: **Overview of the proposed method for unsupervised binary change detection.**

tation. Further described in Chapter 1, we recall that the H3D dataset consists of four different PCs at various dates and comes with labels related to 11 semantic classes that have been manually annotated (Kölle et al., 2021). In practice, the training is performed on H3D PCs acquired in March 2016 on behalf of the national mapping agency of Baden-Württemberg, Germany. In the following study, we refer to this method by Supervised Semantic Segmentation Training (SSST).

4.3.1.2 Training deep feature extraction: self-supervision

Inspired by Saha et al. (2021), we propose a self-supervised approach that does not require complementary data to train the feature extraction network. While in Saha et al. (2021), self-supervised learning idea is based on learning to extract similar features from very different SAR and optical acquisitions from a same scene, we thought the variation in 3D points distribution may also be an advantage. Let us note that even in unchanged parts, 3D PCs may have different distributions due to the various acquisition plans, sensors, weather conditions, etc. Although differences in distributions make the direct comparison of PCs impossible, this property can be an asset for training a network to predict similar attributes over an unchanged area regardless of distribution.

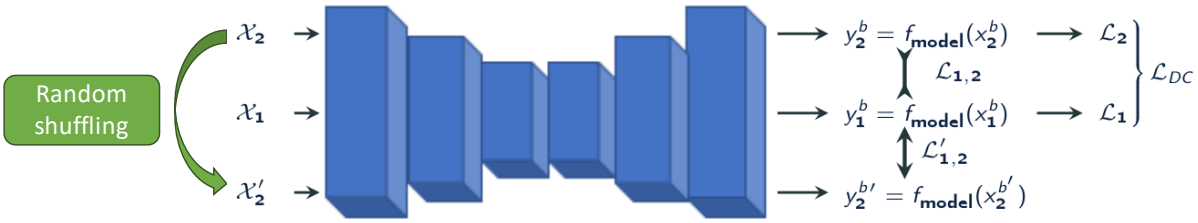


Figure 4.3: **Schema of the self-supervised training of the back-bone.** The three different losses are alternatively used to modulate the model weights: the deep clustering loss $\mathcal{L}_{DC}$, the temporal consistency loss $\mathcal{L}_{1,2}$ (with attractive arrows) and the contrastive loss $\mathcal{L}'_{1,2}$ (with repulsive arrows).

This is the idea of the self-supervised part. The network is trained using three different losses on an unlabeled training set from the same two campaigns of acquisition as the testing set. At each iteration, the back-propagation of the gradient is made using alternatively one of the three losses. Thereby, in each iteration, a batch of $\mathcal{B}$ tiles of the older PC, denoted as $\mathcal{X}_1 = \left\{x_1^1, \ldots, x_1^{\mathcal{B}}\right\}$, and the corresponding $\mathcal{B}$ tiles (i.e., cylinders) of the newer PC, $\mathcal{X}_2 = \left\{x_2^1, \ldots, x_2^{\mathcal{B}}\right\}$, are independently given to the network, resulting in features y:

$$y_1^b = f_{\text{KP-FCNN}}(x_1^b) \tag{4.1}$$

$$y_2^b = f_{\text{KP-FCNN}}(x_2^b) \tag{4.2}$$

where y_1^b and y_2^b have the dimension $N_1^b \times K$ and $N_2^b \times K$ respectively. We recall that N_1^b and N_2^b are the number of points in the corresponding tiles. K refers to the dimension of the output which is in practice the number of desired clusters (see the deep clustering loss below).

As for losses, we alternatively use three different terms, as illustrated in Figure 4.3. The first one is based on the deep clustering principle to force the network to learn discriminative features. Deep clustering relies on a pseudo-label assignment, which will be used to train the network (Caron et al., 2018). More details on its principles will be given in Section 4.4.1. In this study, pseudo-labels are obtained for each point by taking the argument of the maxima as the output of the network. For example, for each point i of the tile $x_1^b \in \mathcal{X}_1$, the corresponding pseudo-label $c_{1,i}^b$ is defined as:

$$c_{1,i}^b = \arg\max_{k \leq K} y_{1,i}^b(k) \tag{4.3}$$

where K is the number of clusters, which is a hyper-parameter to fix. It can be linked with the number of semantic classes to segment in a single PC (note that this does not concern the number of classes of change between two PCs). However, intuitively, if K is small, learned features will not be discriminative enough as large sets of points will be classified in the same class. On the contrary, with excessively high value, features will be too precise, and no generalization will be possible. In this study, it has been set empirically.

Based on these pseudo-labels, two deep clustering losses $\mathcal{L}_1$ and $\mathcal{L}_2$ are defined as the cross-entropy between (c_1, y_1) and between (c_2, y_2) respectively. The average of these two terms is taken to modulate weights:

$$\mathcal{L}_{DC} = \frac{\mathcal{L}_1 + \mathcal{L}_2}{2} \tag{4.4}$$

However, with such losses, we observed that the network was collapsing and predicting all the points in a single cluster. To prevent this obvious solution, a weighting of the cross-entropy losses was done by applying the following weights:

$$W_k = \frac{1}{\sqrt{\alpha N_{C_k}}} \tag{4.5}$$

for each cluster $k \in \{1, \ldots, K\}$, N_{C_k} being the number of points in the cluster k. α is fixed to $K \sum_{h=1}^{K} N_{C_h}$ as done in the public implementation of KP-FCNN in the Torch-Points3D framework (Chaton et al., 2020). Weights are recomputed at each epoch. Intuitively, the deep clustering loss enables the network to learn discriminative features to be able to segment each point into clusters.

In addition to these clustering losses, we add a temporal consistency loss whose rule is to push the network to make similar predictions for tiles from different times but at similar places. As a matter of fact, even in unchanged areas, the point distribution in 3D point clouds differ from each other. Thus, by assuming that permanent changes between two dates are rare in proportion to the unchanged parts in urban areas, the temporal consistency loss enforces the network to make similar predictions for each point of the newer PC compared to the corresponding nearest point in the older PC. Therefore, both predictions (y_1^b and y_2^b) are ordered before computing the loss:

$$y^b_{1_{\text{ordered}}} = y^b_{1_{j|j=\arg\min(\|p_{2i}-p_{1j}\|),\forall p_{2i}\in x^b_2}} \tag{4.6}$$

$$y^b_{2_{\text{ordered}}} = y^b_{2_{i,\forall p_{2i}\in x^b_2}} \tag{4.7}$$

$$l^b_{12,i} = \|y^b_{1_{\text{ordered}},i} - y^b_{2_{\text{ordered}},i}\|_1 \tag{4.8}$$

The temporal consistency loss, $\mathcal{L}_{1,2}$, is then given by taking the mean of $l^b_{12,i}$ over all considered points ($p^b_{2i} \in \mathcal{X}_2$) of all tiles of the batch. Notice that this strategy of nearest point correspondence is similar to the nearest point difference implemented for features fusion in supervised networks developed in Chapter 3.

The third loss is a contrastive loss to encourage the network to produce dissimilar features for different tiles. As proposed in Saha et al. (2019), the contrastive loss is computed in a similar way to $\mathcal{L}_{1,2}$ by having previously randomly shuffled the batch $\mathcal{X}_2$ into $\mathcal{X}'_2$ to obtain different tiles between $\mathcal{X}_1$ and $\mathcal{X}'_2$. The loss $l^{b'}_{12,i}$ is defined as follows:

$$y^b_{1_{\text{ordered}}} = y^b_{1_{j|j=\arg\min(\|p'_{2i}-p_{1j}\|),\forall p'_{2i}\in x^{b'}_2}} \tag{4.9}$$

$$y^{b'}_{2_{\text{ordered}}} = y^{b'}_{2_{i,\forall p'_{2i}\in x^{b'}_2}} \tag{4.10}$$

$$l^{b'}_{12,i} = -\|y^b_{1_{\text{ordered}},i} - y^{b'}_{2_{\text{ordered}},i}\|_1 \tag{4.11}$$

Similarly to Saha et al. (2019), $\mathcal{L}'_{1,2}$ is given by taking the mean of the exponential of the term $l^{b'}_{12,i}$ over all considered points of all tiles in the batch $\mathcal{X}'_2$. Here the exponential is added to avoid overpenalizing the network when $l^{b'}_{12,i}$ is too far from 0. Indeed, even by shuffling $\mathcal{X}_2$, some areas can keep the same semantic, for example there might always be some ground points.

To summarize, the deep clustering loss makes it possible to learn discriminative features, the temporal consistency loss forces the network to predict similar features for similar areas regardless of the point distribution, and the contrastive loss avoids a trivial solution where all predictions are similar for both times. The overall process is given in Algorithm 1 and illustrated in Figure 4.3. This method is referred to as Self-Supervised Learning (SSL) in the following part.

Algorithm 1 Self-supervised training of the back-bone using three different losses.

Initialize KP-FCNN weights
for $e \leftarrow 1$ to $\mathcal{E}$ **do**
  Sample $\mathcal{B}$ tiles from $\mathcal{P}_1$, denoted as $\mathcal{X}_1$
  Obtain corresponding $\mathcal{B}$ tiles from $\mathcal{P}_2$, denoted as $\mathcal{X}_2$
  Obtain $\mathcal{X}'_2$ as random shuffling of $\mathcal{X}_2$
  for $i \leftarrow 0$ to $\mathcal{I} - 1$ **do**
    for $b \in \mathcal{B}$ **do**
      $y_1^b = f_{\text{KP-FCNN}}(x_1^b)$
      $y_2^b = f_{\text{KP-FCNN}}(x_2^b)$
      $y_2^{b'} = f_{\text{KP-FCNN}}(x_2^{b'})$
    end for
    Compute the weights W_k considering y_1^b and y_2^b
    Calculate weighted deep clustering loss $\mathcal{L}_1$
    Calculate weighted deep clustering loss $\mathcal{L}_2$
    Calculate temporal consistency loss $\mathcal{L}_{1,2}$
    Calculate contrastive loss $\mathcal{L}'_{1,2}$
    if $i \bmod 3 = 0$ **then**
      Use $\mathcal{L}_{DC} = \frac{\mathcal{L}_1 + \mathcal{L}_2}{2}$ to modulate KP-FCNN weights
    else if $i \bmod 3 = 1$ **then**
      Use $\mathcal{L}_{1,2}$ to modulate KP-FCNN weights
    else
      Use $\mathcal{L}'_{1,2}$ to modulate KP-FCNN weights
    end if
  end for
end for

4.3.1.3 Deep feature comparison

Once a model is trained to perform a segmentation task, it can be used on both input PCs to extract features at different levels of abstraction and complexity depending on the layer. These extracted features can be used in order to highlight changes applying the Deep Change Vector Analysis (DCVA) principle, initially developed for 2D pixel change retrieving through deep features comparison (Saha et al., 2019). As shown in the comparison module of Figure 4.2, the point-wise change identification is realized by taking the magnitude of the difference (δ^l) between feature vectors computed for each point of the newer PC, $p_{2i} \in \mathcal{P}_2$, with the nearest point of the older PC, $p_{1j} \in \mathcal{P}_1$. In other words, the feature difference δ^l is computed between features $\mathcal{F}_1^l$ and $\mathcal{F}_2^l$ for each PC, $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively, according to the following equation:

$$\delta_i^l = f_{2i}^l - f_{1j|j=\arg\min(\|p_{2i}-p_{1j}\|)}^l \tag{4.12}$$

with $f_{1j}^l \in \mathcal{F}_1^l$, $f_{2i}^l \in \mathcal{F}_2^l$, $p_{1j} \in \mathcal{P}_1$ and $p_{2i} \in \mathcal{P}_2$. Notice that δ^l corresponds to $\ominus$ operation in supervised architectures presented in Chapter 3 (Equation 3.4). The magnitude of the difference, also called deep feature magnitude coefficient, is obtained by taking the L2-norm of δ^l. A threshold is applied on the deep feature magnitude coefficient to distinguish between changed and unchanged points. As in Saha et al. (2019), the threshold is selected using the Otsu algorithm (Otsu, 1979). The trained network extracts similar features for two similar areas, thus the deep magnitude coefficient is close to zero in the unchanged part. The choice of the layer from which features are taken is a hyper-parameter to be set. Combined with SSL, the method is called Self-Supervised Learning - Deep Change Vector Analysis (SSL-DCVA), whereas combined with SSST, the method is referred as Supervised Semantic Segmentation Training - Deep Change Vector Analysis (SSST-DCVA).

4.3.2 Experimental results and discussion

4.3.2.1 Experimental protocol and settings

Experiments with SSST-DCVA and SSL-DCVA methods are conducted on the real AHN-CD dataset introduced in Chapter 1 (Section 1.2) as well as on the simulated dataset with the LiDAR low density configuration (Urb3DCD-V2-1). Concerning the AHN-CD experiments, quantitative results are provided using the sub-part that has been manually annotated to consider a ground truth without errors (see Chapter 1 Section 1.2), further

Dataset	Method	Input cylinders radius (m)	dl_0 (m)	Nb. classes/ pseudo-clusters	Nb pre-training epochs	DCVA layer
Urb3DCD-V2-1	SSST-DCVA	50	1	7	45	8
	SSL-DCVA	50	1	6	15	7
AHN-CD	SSST-DCVA	10	0.2	7	90	8
	SSL-DCVA	20	0.5	6	15	7

Table 4.2: **Summary of hyper-parameters for SSL-DCVA and SSST-DCVA as function of the datasets.**

qualitative analysis is performed on a larger part of the testing set.

As done in the supervised deep framework for 3D PC change detection from Chapter 3, vertical 3D cylinders are chosen to make sure that whatever changes occur between the two dates, at least the ground is visible.

For the SSST part, two different trainings of KP-FCNN are performed to better fit with the final target dataset, i.e., Urb3DCD-V2-1 or AHN-CD, as they do not have the same point density. KP-FCNN is trained using cylinders of 10 m in radius with a first sub-sampling rate of 0.2 m (corresponding to pre-training further used for AHN-CD), and 50 m in radius with a first sub-sampling rate of 1 m (for the pre-training used for Urb3DCD-V2-1). A total of 6,000 cylinders are used for the training at each epoch. The batch size is 10. The 11 H3D classes are fused into 7 classes better suited for the target datasets. It requires around 90 epochs to converge for AHN-CD and 45 for Urb3DCD-V2-1.

Concerning SSL, the training is realized using cylinders of 20 m in radius and a first sub-sampling rate of 0.5 m for AHN-CD. The best results are obtained using 6 clusters for the deep clustering loss, and after 15 epochs of training. One hundred cylinders are used for each epoch, with a batch size of 10. The same configuration is used of Urb3DCD-V2-1 dataset except that with regard to the low point density of this dataset, the input cylinder radius is set at 50 m with a first sub-sampling rate of 1 m.

Networks are optimized using a stochastic gradient descent with a momentum of 0.98. The learning rate is set at 0.01 and decreases exponentially. Regardless of the first sub-sampling rate, the final results are given at the original resolution.

For the choice of the layer to take features for the DCVA, several configurations have been tested. Knowing that KP-FCNN has 9 layers, the best results, reported here, are obtained using the 7th layer for SSST and the 8th for the SSL strategy.

Following the thresholding step, a cleaning of isolated predictions is realized to spatially smooth the results.

Even if no specific unsupervised deep method exists so far in the literature for 3D change detection, for AHN-CD experiments, we decided to compare with the supervised *Siamese KPConv* network (presented in Chapter 3 Section 3.2) trained on the simulated Urb3DCD-V2-1 dataset and directly applied to our AHN-CD testing set without any re-training. The supervised baseline with *Siamese KPConv* whether trained on the AHN-CD training part (with the semi-automatic annotation of change) or Urb3DCD-V2-1 training set is also given. To further benchmark our methods against the state-of-the-art, we provide a comparison with C2C (Girardeau-Montaut et al., 2005) and M3C2 (Lague et al., 2013) distance-based methods. These methods constitute unsupervised baselines for 3D point-based binary change detection (Shirowzhan et al., 2019). In particular, to obtain final binary change information, a thresholding based on Otsu algorithm (Otsu, 1979) is applied on the Hausdorff point-to-point distance computed in C2C. The traditional M3C2 method uses local surface normal and orientation to compute the 3D distances between two PCs (Lague et al., 2013). This method relies on statistical tests on distances between local surface normal and orientation features of the two PCs to automatically extract significant changes. Conversely to Chapter 2 where thresholds are applied on M3C2 distance to distinguish between positive and negative changes, we only need here binary change information, directly available in the significant change variable computed by M3C2 plugin in CloudCompare software. Finally, and to ensure a fair comparison, let us note that we apply as post-processing the same cleaning of isolated predictions on C2C and M3C2 results as done with our methods.

4.3.2.2 Results on real AHN-CD dataset

Quantitative results are given in Table 4.3. Corresponding qualitative results on the manually annotated testing set are presented in Figure 4.4. To complete the qualitative analysis of the results, a larger scene has been visually inspected to understand the behavior of our methods in multiple conditions (see Figure 4.5).

As can be seen in Table 4.3, SSL-DCVA outperforms other methods including SSST-DCVA. It is worth noting that despite its simplicity (no training is required), C2C provides relevant results. However, the point-to-point distance seems limited in places where for example trees have been replaced by a building of approximately the same height (see regions of interest in Figure 4.4h) or where a new building replaced an old one as in the top of zoom 1 where buildings in AHN3 and AHN4 are very different (Figure 4.5(a,d,h)). Conversely to C2C, M3C2 provides inconsistent results here (see Figure 4.4g): the ground

	mAcc (%)	IoU (%)		Computation time	
		Unchanged	Changed	Training	Testing
SSL-DCVA (ours)	**85.20**	**78.91**	**69.38**	9 min	40 sec
SSST-DCVA (ours)	81.88	70.02	63.85	17 hours	40 sec
Siamese KPConv transfer	81.83	75.80	63.73	28 hours	25 sec
M3C2 (Lague et al., 2013)	51.77	3.66	39.90	-	5 sec
C2C (Girardeau-Montaut et al., 2005)	76.67	76.98	53.34	-	5 sec
Siamese KPConv (supervised)	94.23	92.27	87.65	15 hours	25 sec

Table 4.3: **Quantitative results of SSL-DCVA and SSST-DCVA on AHN-CD dataset.** Approximate computation times are provided for the training and testing (on the manually annotated part) step.

elevation has changed slightly between the two acquisitions, so almost all areas are marked as changed. Notice that even when removing ground points for metric computation, the M3C2 method is lagging behind other methods (still about 10% of mAcc behind the SSST-DCVA algorithm evaluated under the same conditions). Moreover, in this study, we aim at detecting changes in object semantics (new buildings, demolition, new vegetation, etc.), so a change in the ground height is not of interest to us and has not been marked as changed in the ground truth. To further explain the relative poor results of M3C2, we recall that this method was originally developed in a geoscience context to detect changes at different scales including centimetric (Lague et al., 2013). Thereby, it might be inadequate for urban environments, as already noticed in Chapter 2. A distance-based method may not distinguish between topographic and semantic changes as long as the geometry of objects has changed. Learning-based methods (see Figure 4.4(d-f)) seem to also retrieve small changed objects, which is not possible with distance-based methods without including too many changes.

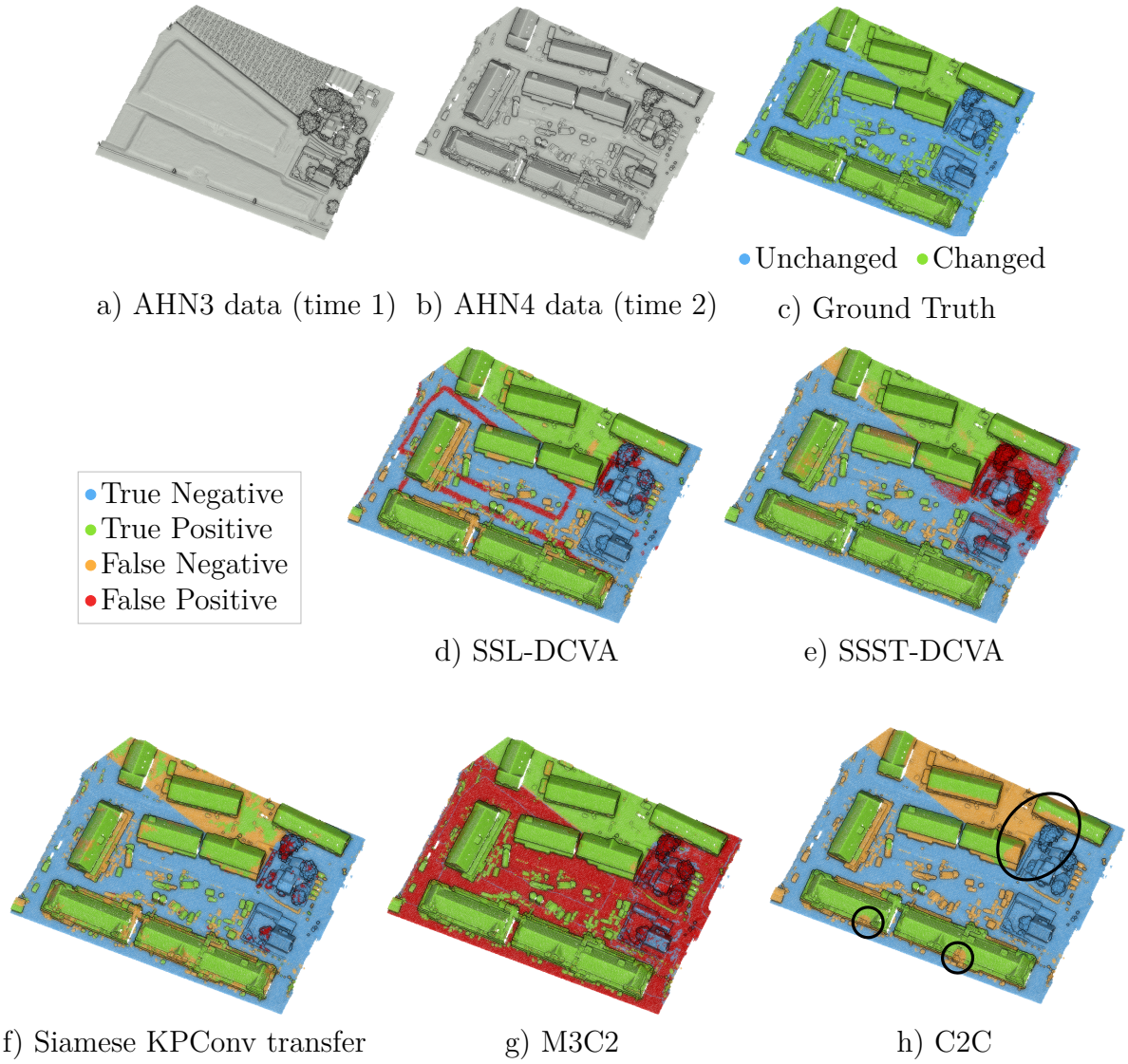
By looking at the results of the *Siamese KPConv* change detection network with transfer onto AHN dataset from Urb3DCD-V2 simulated dataset, we can see that performances are quite similar to the SSST-DCVA but are overtaken by SSL-DCVA method. In particular, some differences with the ground truth are visible in the demolished area and at object boundaries. This is probably due to the difference of building types present in the selected area of AHN dataset. Indeed, Urb3DCD dataset contains buildings from a French city center different from training and testing areas, as for example, AHN3 data (time 1) contains a glasshouse. The same problem occurs with SSST-DCVA methods, since H3D PCs have different resolution and quality than AHN PCs. This shows the

advantage of training directly on a dataset with similar properties to the testing set and using recent developments in self-supervised learning. However, when compared to the supervised *Siamese KPConv* network, unsupervised methods can still largely be improved. The main differences of SSL-DCVA with the ground truth are visible on small objects such as vehicles, road signs or vegetation (see Figure 4.4(c,d)). Furthermore, as can be seen in the buildings on the left side of Figure 4.4 and right side of Figure 4.5b, some omissions remain on new buildings with a flat roof. When looking at the mono-date segmentation of the PC realized before the DCVA step, one can see that flat roofs are classified in the same class as ground so, when comparing features, no changes are highlighted. This raises the difficulty of late-fusion change identification. Indeed, errors in the feature extraction step are propagated in the comparison step. Finally, some false detections are visible on the ground, forming a large trapezium (see Figure 4.4d). This is due to changes in the orientation of the ground surface.

Both of our methods encounter difficulties in unchanged vegetated areas (see the top of zoom 2 in Figure 4.5(i-k)) certainly because of the complexity of LiDAR data in such areas with a high variation of point distribution even without changes in the semantics of objects. This results in a mixture of points predicted as changed and unchanged. Furthermore, these vegetated areas may have grown, and the acquisition not realized in the same season implies some differences on the 3D representation of trees. Looking at Figure 4.5l, we can observe that C2C method is not better in this zoom where the vegetation has been removed. Indeed, in AHN3 some points are acquired from the ground to the top of the tree canopy thanks to the LiDAR sensor, thereby the point-to-point distance is not an efficient indicator for changes.

The one-to-one nearest neighbor correspondence lacks precision in the presence of occlusion in the 3D PCs. Indeed, due to the geometry of acquisition, some occlusions may appear in PCs, these hidden parts may not be similar in the two compared PCs leading to difficulties when comparing points in the DCVA part.

Finally, once again, SSL-DCVA seems more interesting than SSST-DCVA when looking at training time. SSST-DCVA takes about 17 hours to train on H3D dataset, while SSL-DCVA only requires about 9 minutes to train. The DCVA part on the manually annotated testing set takes about 40 seconds. All experiments were realized on a single GPU (Nvidia Tesla V100 SXM2 16 GB).

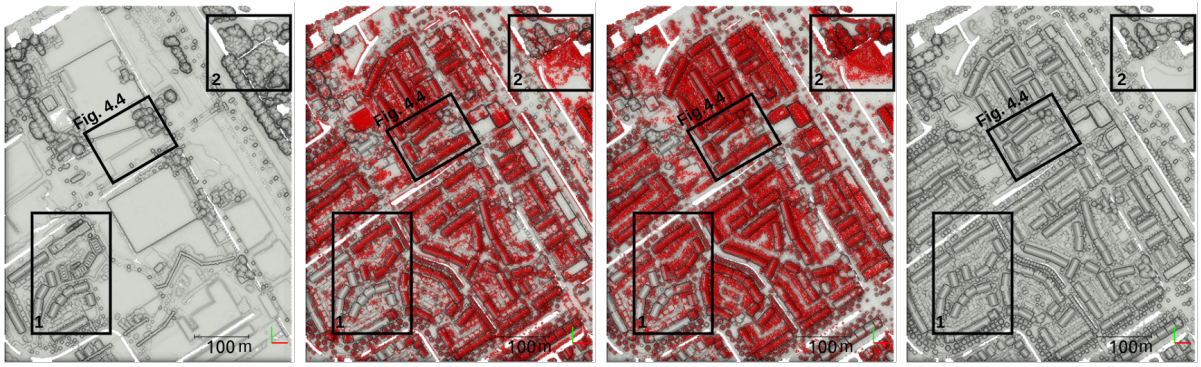


a) AHN3 data (time 1) b) AHN4 data (time 2) c) Ground Truth

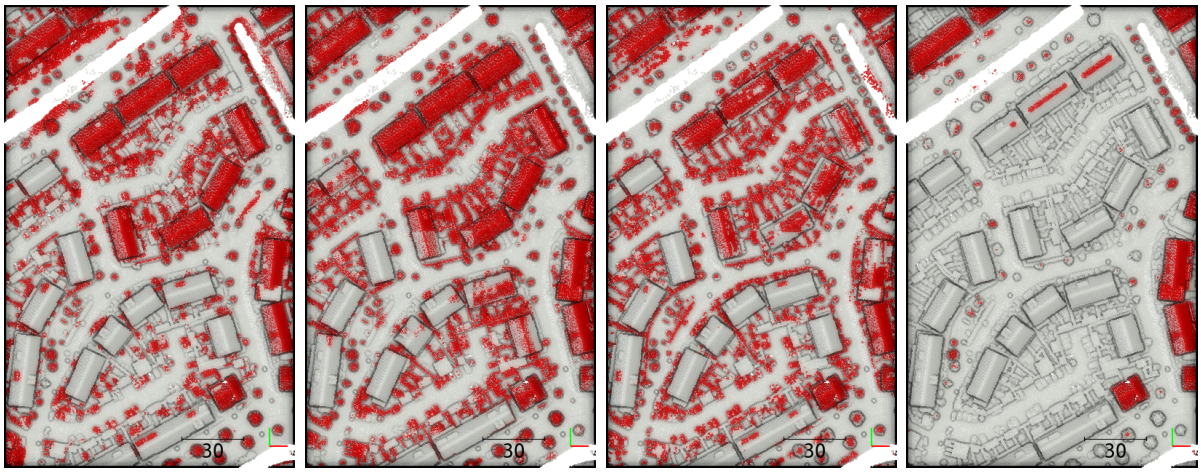
d) SSL-DCVA e) SSST-DCVA

f) Siamese KPConv transfer g) M3C2 h) C2C

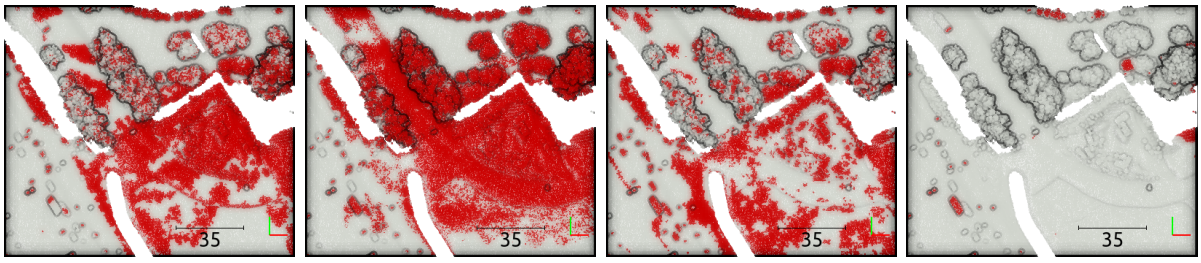
Figure 4.4: **Qualitative results on the manually annotated testing set.** Changes are indicated in red. Even if not perfect, deep learning based methods (SSL-DCVA (**d**), SSST-DCVA (**e**) and *Siamese KPConv* transfer (**f**)) seem to better retrieve changes in small objects than distance based methods (M3C2 (**g**) and C2C (**h**)). Regions of interest specifically discussed in the text are highlighted with ellipses.



(**a**) AHN3 data (time 1) (**b**) SSL-DCVA results (**c**) SSST-DCVA results (**d**) AHN4 data (time 2)



(**e**) Zoom 1: SSL-DCVA (**f**) Zoom 1: SSST-DCVA (**g**) Zoom 1: SKPConv transfer (**h**) Zoom 1: C2C



(**i**) Zoom 2: SSL-DCVA (**j**) Zoom 2: SSST-DCVA (**k**) Zoom 2: SKPConv transfer (**l**) Zoom 2: C2C

Figure 4.5: **Qualitative results on the testing set (not annotated).** Changes are indicated in red. For more precise results, zoom 1 and 2 are visible in (**e**) to (**l**), while zoom Fig. 4.4 corresponds to the manually annotated area presented in Figure 4.4.

	mAcc (%)	IoU (%) Unchanged	IoU (%) Changed
SSL-DCVA	74.02	84.10	40.38
SSST-DCVA	74.24	78.64	36.30
M3C2 (Lague et al., 2013)	71.40	65.41	29.89
C2C (Girardeau-Montaut et al., 2005)	**74.37**	**89.53**	**46.96**
Siamese KPConv (supervised)	94.52	96.03	82.63

Table 4.4: **Quantitative results of SSL-DCVA and SSST-DCVA on Urb3DCD-V2-1 dataset.**

4.3.2.3 Results on simulated Urb3DCD-V2-1 dataset

Quantitative results on the simulated Urb3DCD-V2-1 dataset are presented in Table 4.4. On this table, and conversely to previous results on AHN-CD dataset, we observed that both of the proposed methods do not seem to provide relevant results. Indeed, C2C distance-based method ends with a better change detection performance according to the IoU of changed and unchanged areas.

To explain these results, let us remark that SSL-DCVA method relies on two assumptions, one of which is that changes are rare. Even if this dataset is largely imbalanced (with a large proportion of points labeled as unchanged), it still contains a lot of changed objects (see illustrations of the simulated dataset for example in Figure 1.11). Thereby, the temporal consistency loss (Equation 4.8) may be disturbed. To assess this parameter, we propose to filter out training cylinders according to the percentage of unchanged points (using the ground truth). In Figure 4.6, we show the mIoU as a function of a purity coefficient corresponding to the threshold set on the percentage of unchanged points in training cylinders. When purity is 0, the threshold is set to 0 and all cylinders can be used for the SSL training, whereas when it is set to 0.9, cylinders used for the training contains at least 90% of unchanged points. As can be seen on this figure, there is a slight augmentation of performances of the methods when cylinders are filtered (purity coefficient strictly greater than 0). Notice that the real increase in performance is only visible with a threshold at 98% of unchanged points in training cylinders. To explain this, let us note that cylinders from this dataset contain around 84%($\pm$10%) of unchanged points. Thus, a purity between 0 and 0.8 almost does not filter any training cylinders. On the opposite, setting the purity coefficient to 1 is not optimal. Indeed, in this dataset, cylinders containing only unchanged points are very rare ($\approx$ 0.3% of the sampled cylinders without any selection, e.g., purity set to 0). Therefore, forcing the purity at 1 surely involves several times the

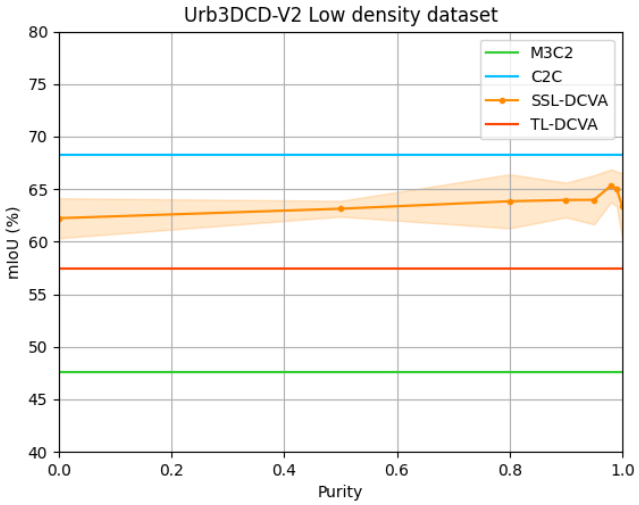


Figure 4.6: **Mean of IoU results for SSL-DCVA method as a function of the purity coefficient.** Purity coefficient corresponds to the threshold set on the percentage of unchanged points in training cylinders. M3C2, C2C and SSST-DCVA mIoU are indicated for comparison purpose, these methods do not depend on the assumption that changes are rare.

same cylinders in a training epoch and leads to overfitting.

On top of being simpler, C2C stays better than our methods in the simulated dataset, even with a large purity coefficient. A problem more inherent to the method seems however to persist. To our opinion, the problem comes from the DCVA part which relies on a point-to-point comparison based on the nearest point. This point comparison is not optimal in occluded parts as well as in dense urban areas (which is the case for Urb3DCD datasets that are acquired on models of Lyon city center). This last point has been already mentioned when describing C2C misclassifications in AHN-CD results in the last sub-section (see region of interest in Figure 4.4g). Even if DCVA comparison relies on multiple deep features, avoiding the problem when the two points being compared have different latent embeddings, the problem remains when the latent embedding of the two points under comparison is similar, meaning the same class is predicted by the back-bone network (whether trained by transfer learning or self-supervision). Indeed, in this case, the deep magnitude coefficient computed from deep features will be similar.

Furthermore, DCVA part relies on a binary thresholding (through Otsu method) of the histogram of the deep magnitude coefficient. This thresholding is performed of the testing set. To obtain a meaningful value, the area must contain enough changed objects

which is not always the situation in our experiments.

Finally, these methods provide only binary change segmentation. By looking at results on both datasets, a distance based method (C2C) seems to provide results not so far from our methods, or even better. Thereby, with regard to the complexity of training a deep method for only a binary change detection, we would recommend using a traditional method (e.g., C2C distance-based method) until more refined techniques are proposed. Considering the wide diversity of changed objects, binary change segmentation seems restrictive, thus appealing for unsupervised methods for multiple change segmentation.

4.4 Unsupervised multiple change detection with deep clustering

Depending on the use case, distinguishing between multiple types of changes can be a matter of interest. For example, in an urban context, some applications would require to identify new buildings, while others would rather focus on new balconies on remaining buildings. When dealing with supervised change detection, the annotation simply follows the use case. However, for unsupervised settings, the question is more complex. It would rather be interesting to have a method able to cluster a PC into several change categories and then let the user select which category is interesting for his use case. To do so, we propose to rely on the deep clustering principle, and in particular on DeepCluster (Caron et al., 2018). Deep clustering consists in jointly optimizing deep representation of the data and performing clustering with learned features (Ren et al., 2022; Zhou et al., 2022).

4.4.1 Methodology

Before describing our method for 3D PCs multiple change segmentation, an overview of DeepCluster (Caron et al., 2018) is given as it is an important component of our model.

DeepCluster principle

Among the variety of studies related to deep clustering (Ren et al., 2022; Zhou et al., 2022), DeepCluster appears to be one of the most fundamental. Proposed by Caron et al. (2018), this method resides in a rather simple idea of alternatively clustering deep latent representation of data to obtain pseudo-labels further used to train a Convolutional Neural Network (CNN), as illustrated in Figure 4.7. In particular, the convolutional network is

trained in a supervised manner using pseudo-labels as objective for prediction. In a traditional supervised approach, giving a set of N images x_n ($n \in [1, N]$), a parametrized classifier g_W predicts the correct labels (y_n) using the features extracted by $f_\theta(x_n)$ (W and θ being the parameters from the classifier and the back-bone convolutional model, respectively). They are optimized according to the following problem:

$$\min_{\theta,W} \frac{1}{N} \sum_{n=1}^{N} \ell(g_W(f_\theta(x_n)), y_n) \tag{4.13}$$

where ℓ is the loss function, a classical negative log-likelihood (Equation 3.6) in their method. This cost function is minimized using standard mini-batch stochastic gradient descent and backpropagation to compute the gradient. The difficulty in an unsupervised setting is therefore to define y_n.

In DeepCluster, Caron et al. (2018) proposed to rely on a classical clustering algorithm such as k-means (MacQueen, 1967) or power iteration clustering (PIC) (Lin and Cohen, 2010) to obtain a pseudo-label (y_{PL_n}) that is used instead of y_n. Caron et al. (2018) showed that the choice of the clustering algorithm is not crucial. Thereby, for illustration purposes, we continue with the example of k-means algorithm. This clustering method matches data to k groups (pseudo-clusters) by minimizing distance between each data and its corresponding cluster center, called centroid (and contained in the centroid matrix C in practice).

Finally, the unsupervised training process alternates between i) clustering the output features of the back-bone convolutional model ($f_\theta(x_n)$) (clustering step), and ii) update parameters θ and W using the obtained pseudo-labels (y_{PL_n}) thanks to Equation 4.13 (training step). This relies on the fact that a MLP classifier on top of a standard CNN with randomly initialized weights (θ) provides results far above from the chance (i.e., random) level (Noroozi and Favaro, 2016).

In practice, a few tricks are required to avoid trivial solutions, e.g., assigning all the inputs to the same cluster. First, the authors get rid of empty clusters by randomly dividing in two groups the largest cluster when an empty cluster appears. Second, if the pseudo-cluster representation of data is largely imbalanced, the deep model will tend to assign all data to the most represented pseudo-cluster. To counter this, they propose to sample input images during the training based on a uniform distribution over the pseudo-labels.

They showed the robustness of their method by training different architectures (Alexnet

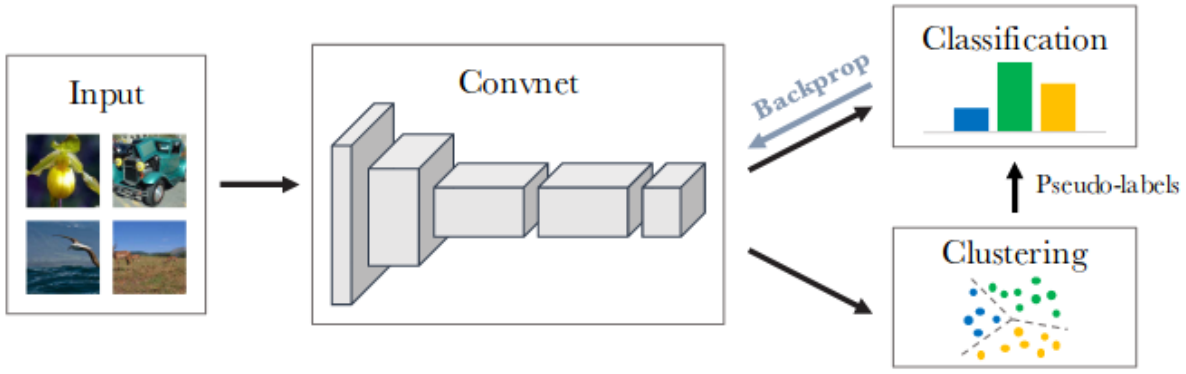


Figure 4.7: **Illustration of DeepCluster method.** Source: Illustration from Caron et al. (2018).

(Krizhevsky et al., 2017) and VGG-16 (Simonyan and Zisserman, 2014)) on ImageNet (Deng et al., 2009) or YFCC100M (Thomee et al., 2016) images datasets.

In the following, we adapt this principle to 3D PCs change detection.

DC3DCD: unsupervised learning for 3D multiple change extraction

Whereas the task and the data (2D image classification) of DeepCluster is far from 3D PCs multiple change segmentation, we nevertheless decided to adapt this method to our task and particular data. By replacing the CNN by a 3D PCs change detection back-bone, some change-related features can be extracted. Thereby, the clustering of these deep features results in change-related pseudo-clusters. We further rely on these pseudo-clusters to optimize the trainable parameters θ of the change detection back-bone. Figure 4.8 illustrates our method called DeepCluster 3D Change Detection (DC3DCD).

Algorithm 2 Fully unsupervised DeepCluster 3DCD training

```
Initialize the back-bone trainable parameters θ
for e ← 1 to ℰ do
    Run mini-batch k-means to obtain centroids C on the whole training set
    Assign to each point of the training set a pseudo-cluster
    Replace parameters of the prototype layer by C
    Compute the weights W_k considering pseudo-label distribution in the training set
    Training sample selection: random drawing considering W_k
    for i ← 1 to ℐ do
        Use ℒ_NLL (weighted by W_k) to modulate the back-bone trainable parameters (θ) considering pseudo-labels
    end for
end for
```

The overall training process of our method is given in the Algorithm 2. Even if the

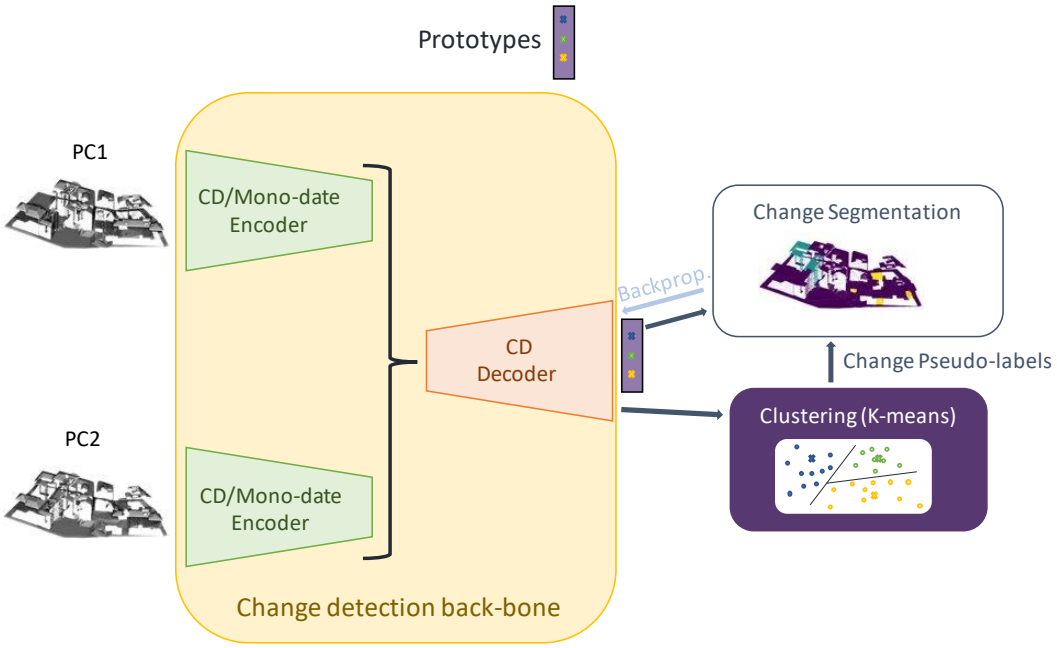


Figure 4.8: **Illustration of our proposed method: DC3DCD.** It is trained by alternatively clustering deep features to match a pseudo-label to each point of PC 2. These pseudo-labels are used to optimize the back-bone trainable parameters.

general idea of DeepCluster remains, some specific features of DC3DCD distinguishing it from the original DeepCluster proposal should be noted, namely:

- **Back-bone model**: Traditional CNN model cannot be used in raw 3D PCs and furthermore, it does not render change-related deep features. In the Chapter 3 of this thesis, we defined different architectures for supervised change detection in 3D PCs. These architectures can be used as back-bone to our unsupervised method. In practice, both *Siamese KPConv* (Figure 3.5) and *Encoder Fusion SiamKPConv* (Figure 3.17) will be experimented. Thus, parameters θ to be optimized are parameters of these back-bone architectures.

- **Use of a prototype layer**: In the original version of DeepCluster, the final classification layer of g_W is re-initialized before each parameter optimization session (i.e., training steps) because there is no correspondence between two consecutive cluster assignments. The authors further proposed an improved version of DeepCluster (DeepCluster-V2) by replacing the classifier g_W by the prototypes, i.e., cluster centers. This ends up with an explicit comparison of the features and the centroid

matrix C, and tends to improve stability and performance of DeepCluster (Caron et al., 2020). According to preliminary experiments, we also decided to use the centroid matrix C defining pseudo-cluster centers. Therefore, the last fully connected layer parameters of the back-bone are set using the centroid matrix C. This so-called prototype layer is updated after each clustering step and fixed during the training step.

- **Input data**: We aim at detecting changes into raw 3D PCs. In addition of using a specific back-bone able to compute features directly in 3D PCs, some pairs of bi-temporal vertical cylinders need to be used as input to the network (as for previous methods of this thesis). Furthermore, in their experimentation, Caron et al. (2018) provide Sobel-filtered images as input to the CNN instead of RGB images. Sobel filtering acts as an edge detector thanks to the computation of gradients on the image. This seems an important step in their method (Caron, 2021; Mustapha et al., 2022) and acts as a pre-computation of relevant features. However, when dealing with 3D PCs, there is no direct equivalent to Sobel filtering. As presented in Chapter 2, some traditional hand-crafted features are designed to extract geometric attributes in 3D PCs (e.g., linearity, planarity, omnivariance, etc.). Similarly, these hand-crafted features may support the network to focus on interesting PCs attributes as shown in Chapter 3 (Section 3.4.1). Thus, we will experimentally assess the contribution of hand-created features used as input to the network together with point coordinates. The hand-crafted features are related to point distribution (N_x, N_y, N_z,L_λ, P_λ, O_λ), height (Z_{range}, Z_{rank}, nH), and change (*Stability*). Although very different from the gradient computation idea, *Stability* could be interesting to guide the network in the change detection task (Chapter 3 Section 3.4.1). These ten hand-crafted features are detailed in the Chapter 2 (Section 2.2.1).

- **Size of the training set**: Change segmentation task implies assigning a pseudo-label to each point of the second PC (of pairs of the training set). To fit in memory, a mini-batch k-means (Sculley, 2010) clustering is used. The principle of splitting the largest cluster when an empty cluster appears is used as in DeepCluster.

- **Imbalanced dataset**: As already mentioned, change detection datasets are highly imbalanced. To avoid falling in a trivial solution where the back-bone predicts all points with the same label, after each clustering step weights W_k (considering pseudo-labels distribution) are computed using the same formula as in Equation 4.5.

These weights are further used to both select training cylinders and weight the loss ($\mathcal{L}_{NLL}$). We remind that the cylinder selection process was also applied in the supervised context. It aims at giving more training samples of underrepresented pseudo-clusters. It also acts as a kind of data augmentation because from one epoch to another, selected cylinders may differ according to the random drawing of the cylinder's central point. There are surely overlapping between cylinders, but the center of the cylinder may differ conversely to when the whole set is predicted (as in the clustering of the training set, where cylinders are selected according to a grid to cover the whole area). Without this trick, the method is likely to collapse to a single class prediction.

- **3D PCs data augmentation**: During the training step, data augmentation appears to be crucial for stability and performance of the method. The same data augmentations as in the supervised frameworks are used: random cylinders rotation around the vertical axis (same angle for both cylinders of a pair), and addition of a Gaussian noise at point level.

From predicted pseudo-labels to real labels

The above training using DC3DCD method is fully unsupervised, thereby no use of a ground truth is required. At the end of the overall training process, the back-bone predicts labels for all points of the second PC according to the change. Predicted labels do not directly correspond to the real labels. There is an oversegmentation of PCs inducted by the choice of K, the number of pseudo-clusters, which is often large compared to the number of real classes. By opting for such an oversegmentation setting, we expect to be able to address various use cases with different size and precision of classes. One real class is then composed of several predicted clusters, while we assume a predicted cluster to contain only one real class. To map a real label onto each predicted label, a mapping step is necessary. For this mapping, we consider that the user should be involved in order to select the kind of changes that is of interest given the use case. DC3DCD enables to train a back-bone to segment the PC into small areas containing the same types of change or unchanged objects. Thus, the user just has to select for each predicted cluster a corresponding real class. It can be viewed as a kind of active learning process. This is illustrated in Figure 4.9. This strategy finally involves K annotations to obtain a final change segmentation over the whole testing set (no matter its size). K corresponds to

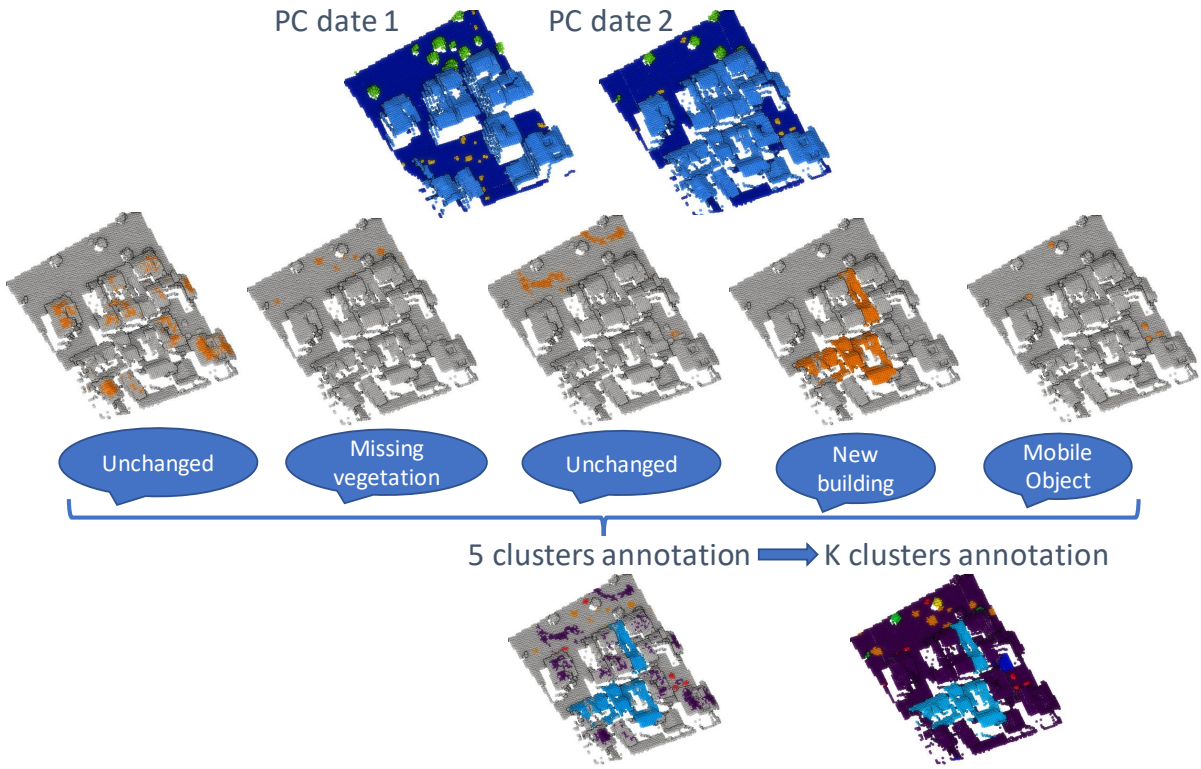


Figure 4.9: **Weakly supervised mapping of predicted clusters to real classes.** For the K predicted clusters, a mapping with the corresponding real class is performed by a user to obtain the final change segmentation of the PC. 5 mappings are provided for the sake of illustration. Segmenting the whole dataset requires K annotations only. This is far less than the millions of points that need to be annotated in order to build training and validation sets in a supervised setting.

the number of pseudo-clusters used during the training. This hyper-parameter to the method has to be fixed beforehand. For this reason, we describe our method as weakly supervised, as the required K annotations are much smaller than the millions of points contained in the dataset. In practice, for the experimental assessment of our method, we map a predicted cluster onto a real class taking into account the real majority class it contains.

4.4.2 Experimental results

4.4.2.1 Experimental settings and protocol

Below, we detail how we fix the main hyper-parameters and experimental set-up.

- **Number of pseudo-clusters** K: As shown in different studies related to DeepCluster (Caron et al., 2018; Mustapha et al., 2022), the choice of the number of pseudo-clusters is important. We experimented several values for K on the simulated dataset and found that 1,000 was an adequate compromise to have a stable training and not too many clusters. A too small value may not reflect all different types of changes (and thus not allow the user to select the changes of interest), while a too large value ends up with along training and annotation time. The same value will be used also with the real dataset AHN-CD.

- **Training step and parameters optimization**: For the training step, a SGD with momentum of 0.98 is applied to minimize a point-wise negative log-likelihood (NLL) loss (Equation 3.6) using the pseudo-labels defined in the clustering step. A batch size of 10 is used. The initial learning rate is set to 10^{-3} and scheduled to decrease exponentially. As in Caron et al. (2018), we experimentally verified that reassigning the clustering after each epoch is better than an update after each n epochs. Indeed, if several training epochs are conducted, the model seems to converge in the first local minimum associated with a non-optimal pseudo-clustering. In each epoch, 3,000 cylinder pairs are seen by the model. A total of 55 epochs, i.e., 55 clustering and training steps, is performed.

- **Datasets**: Both simulated Urb3DCD-V2 in low density LiDAR configuration and real AHN-CD dataset will be experimented. During the training, we make no use of the ground truth unless for the method assessment purpose (see Section 4.4.2.2). The first sub-sampling rate dl_0 is set to 1 m and the cylinder radius to 50 m for Urb3DCD-V2-1 while for AHN-CD dl_0 is set to 0.5 m and the cylinder radius to 20 m because of the difference of density of both datasets.

- **Comparisons with supervised methods**: A comparison with supervised methods proposed in Chapter 3 is provided, namely *Siamese KPConv*, *Encoder Fusion SiamKPConv*, DSM-based deep learning methods (adaptation of Daudt et al. (2018) networks to DSM inspired by Zhang et al. (2019)) and the RF algorithm trained on hand-crafted features (Tran et al., 2018).

- **Adaptation of a supervised method to a weakly supervised setting:** To evaluate the benefit of our method, we propose a comparison with deep learning-based supervised methods tuned to a weakly supervised setting. In practice, we use

Encoder Fusion SiamKPConv, the best of our supervised techniques presented in 3. To this end, we trained it with the same amount of annotated data as our DC3DCD setting. However, this is not straightforward since this network cannot be trained with only 1,000 points. Indeed, as during the supervised training of this network, labels should be provided for each point of the second PC of the pair, and as a cylinder contains more than 1,000 points (about 3,500), we cannot directly compare the supervised training with the same amount of labels (i.e., $K = 1,000$). Thus, both training and validation sets, we chose 7 cylinders, each one centered on one of the 7 classes contained in Urb3DCD-V2 to be sure that each class is represented. Note that with this minimal training configuration, the number of annotated points in the 14 cylinders is around 50,000. As 7 cylinders are less than the batch size of 10 used for all other deep learning-based method, we also provide results with a batch size of 2.

- **Comparisons with unsupervised methods**: To the best of our knowledge, there is no other weakly supervised or unsupervised deep learning method tackling 3D PCs multiple change segmentation (DCVA deals with the binary case only). Thereby, we provide a comparison with a k-means algorithm applied to the ten hand-crafted features of Tran et al. (2018) dedicated to change detection in 3D PCs. Note that LiDAR specific features (e.g., intensity or number of echoes) used in Tran et al. (2018) are ignored here since the simulated dataset does not contain such information. For fair comparison, the k-means is set to predict also $K = 1,000$ pseudo-clusters. The same user-guided mapping is done as proposed in the previous section (Figure 4.9) to assign the final classes.

Before presenting the quantitative results, we analyze in the next section the behavior of the network during the training.

4.4.2.2 Analysis of the learning process

Before presenting the results on the different datasets, we propose to study the behavior of DC3DCD during the training phase. To do so, we rely on the *Encoder Fusion SiamKPConv* back-bone and the configuration without the use of the ten hand-crafted features as input, so considering only 3D points coordinates. Note that the same tendencies are obtained with hand-crafted features or with *Siamese KPConv* back-bone, but we prefer to show results with a network that takes into account the minimum information regarding

changes. In practice, we compute criteria associated with the clustering quality and the pseudo-cluster distribution along the training process.

The evolution of the **clustering quality** during training epochs is computed by comparing the pseudo-clusters obtained thanks to the k-means on deep features, and the real classes. More precisely, we compute the normalized mutual information (NMI) given by the following formula:

$$NMI(Y, Y_C) = \frac{I(Y, Y_C)}{\sqrt{H(Y)H(Y_C)}} \tag{4.14}$$

where Y and Y_C contain the probabilities p_i, p_{C_i}, of each label $i = \{1...N\}$ associated with the true and pseudo-labels. H is the entropy defined as:

$$H(Y) = -\sum_{i=1}^{N} p_i \log_2 p_i \tag{4.15}$$

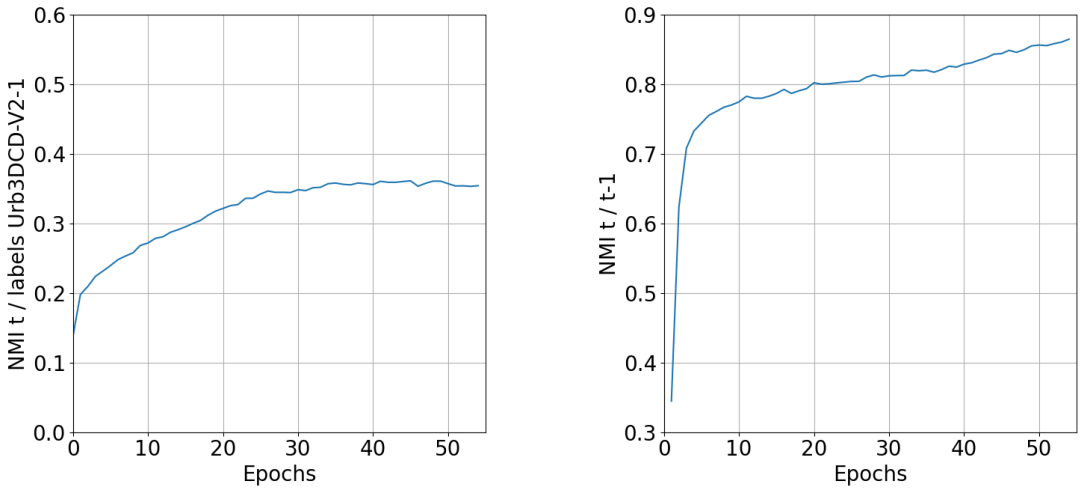
and I is the mutual information, defined as:

$$I(Y, Y_C) = H(Y) - H(Y|Y_C) \tag{4.16}$$

Intuitively, the NMI is a measure of the information shared between two clusterings, i.e., in our case the clustering of deep features and real classes. If the NMI is equal to 0, the two clusterings are totally independent. On the opposite, if the NMI is equal to 1, there is a perfect correlation between the two clusterings, i.e., one of them is deterministically predictable from the other.

We present the evolution of the clustering quality along the epochs in Figure 4.10a by giving the NMI between the clustering and the real labels of Urb3DCD-V2 dataset. As can be seen, the clustering tends to get closer to real classes along with the training process. It seems to stabilize after 30-40 epochs. Let us remark that at the end of the training, the NMI is around 0.35. It is still far to 1, but the same trend was observed in DeepCluster training quality assessment by Caron et al. (2018). In Figure 4.10b, we evaluate the number of reassignments of cluster from one epoch to the following using the NMI between the clustering of the two epochs. It seems that during the first epochs, there is an important evolution of the clustering, but the training rapidly converges to a rather stable clustering ($NMI > 0.8$). Again, the same tendency was obtained by Caron et al. (2018) for DeepCluster on ImageNet dataset.

As for the pseudo-cluster **distribution**, we remind that, ideally, a pseudo-cluster

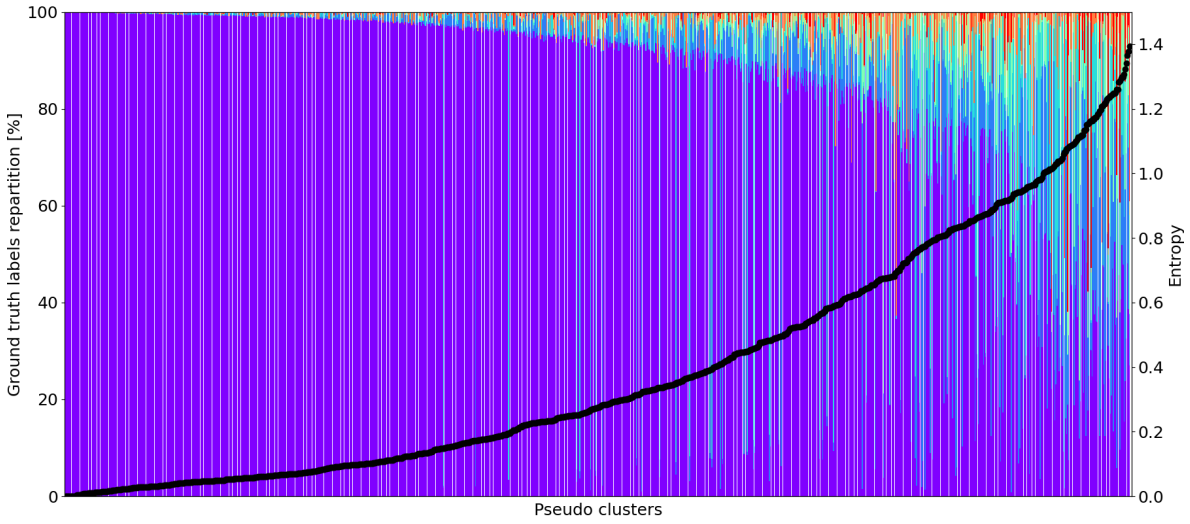


(**a**) Clustering quality (**b**) Cluster reassignment

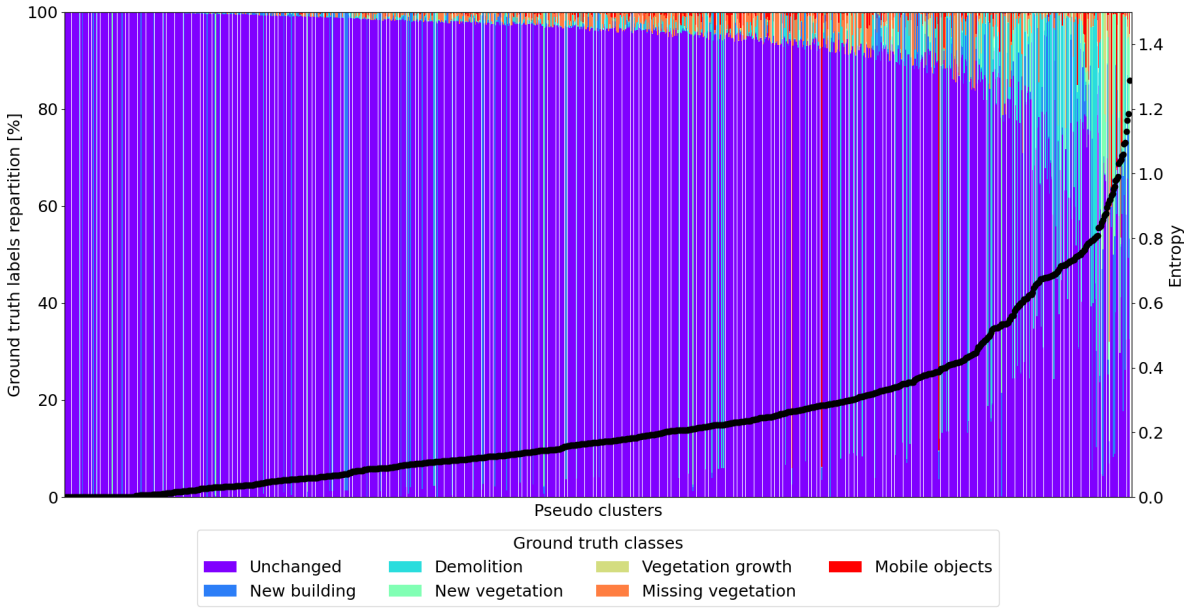
Figure 4.10: **Analysis of the behavior of DC3DCD during the training.** The evolution of clustering quality (**a**) is given thanks to the NMI between the clustering and the real labels of Urb3DCD-V2 dataset. The NMI between the clustering at epoch t and the clustering at epoch $t-1$ gives the cluster reassignment (**b**).

contains only one real class, and a real class can be distributed into several pseudo-clusters. To measure the purity of a pseudo-cluster, we propose to investigate the entropy H (Equation 4.15) of each pseudo-cluster. If it is near 0, then the pseudo-cluster contains almost only one real class. However, if a pseudo-cluster is divided into several classes, the entropy is higher. In Figure 4.11 is given the pseudo-cluster distribution at epoch 10 and 50. Pseudo-clusters are sorted in increasing entropy values. First, as can be seen, there is an improvement between epoch 10 and 50. The area under the entropy curves (dotted points in Figure 4.11) is indeed smaller at epoch 50 (0.24 of mean entropy) than at epoch 10 (0.39 of mean entropy), meaning that entropy values are globally smaller. Then, after 50 epochs of training, 80% of the pseudo-clusters have a "purity level" greater or equal to 93%. These results confirmed the relevance of the proposed weakly-supervised strategy to automatically map a pseudo-cluster onto the majority real class for the evaluation, as stated in the method description section (see Section 4.4.1).

After having studied the training behavior of the DC3DCD method, we will now compare it to the state-of-the-art on the testing set of both simulated and real datasets.



(**a**) Epoch 10



(**b**) Epoch 50

Figure 4.11: **Ground truth class distribution in pseudo-clusters** sorted by increasing entropy value at epoch 10 (**a**) and 50 (**b**). Each pseudo-cluster entropy is given by the dotted black curve.

4.4.2.3 Results on simulated Urb3DCD dataset

Quantitative evaluation of DC3DCD on the simulated Urb3DCD-V2-1 dataset is given in Tables 4.5 and 4.6. In these tables, we also recall supervised results of Chapter 3 for comparison purpose with a supervised technique. Let us first analyze DC3DCD results without hand-crafted features (two first lines of the bottom part of Tables 4.5 and 4.6). As can be seen, results for both *Siamese KPConv* and *Encoder Fusion SiamKPConv* back-bones are rather low. Indeed, while requiring the same annotation effort, k-means algorithm trained on the same hand-crafted as the RF method proposed in Tran et al. (2018) provides better results (cf. first line of middle part in Table 4.5). However, these results are interesting because the two experimented back-bones provide significantly different results. In particular, *Encoder Fusion SiamKPConv* ends up with a mIoU_{ch} 1.5 times higher than *Siamese KPConv*. While in a supervised setting, the improvement of *Encoder Fusion SiamKPConv* was of 5 points of mIoU_{ch}, in the weakly supervised context the choice of the architecture seems even more crucial.

Then, when hand-crafted features are added to the input of the network, results are largely improved (cf the two last lines of Table 4.5). While DC3DCD with *Siamese KPConv* architecture and hand-crafted features provides results comparable to the k-means algorithm, DC3DCD becomes interesting with both hand-crafted features and the *Encoder Fusion SiamKPConv* architecture. Indeed, in this configuration there is more than 15 points of mIoU_{ch} of improvement compared to the k-means. Furthermore, DC3DCD with this configuration is better than a fully supervised RF, and provides results comparable with fully supervised deep architectures trained on 2.5D rasterization of 3D PCs. Thereby, providing hand-crafted features is an important step in weakly supervised settings. One possible interpretation is that the unsupervised version is very tricky to train in reason of the large number of possible local minima. Adding hand-crafted features probably helps the initialization to be closer to the global minimum.

Notice that the *Encoder Fusion SiamKPConv* in a weakly supervised setting provides rather low results given the higher annotation effort required (about 50,000 annotated points). Results with a batch size of 10 are not stable. This is explained by the fact that only one batch is seen per epoch, and the same learning rate scheduler as with a batch size of 2 is used. Thus, this training is more prone to fall in a local minimum. Even with a reduced batch size, leading to more stable results, we can see the benefit of using DC3DCD for the training of *Encoder Fusion SiamKPConv* network because the effort of annotation is lower.

	Method	mAcc (%)	mIoU$_{ch}$ (%)
Supervised	Siamese KPConv	**91.21** ± 0.68	**80.12** ± 0.02
	Encoder Fusion SiamKPConv	94.23 ± 0.88	85.19 ± 0.24
	DSM-Siamese	80.91 ± 5.29	57.41 ± 3.77
	DSM-FC-EF	81.47 ± 0.55	56.98 ± 0.79
	RF	65.82 ± 0.05	52.37 ± 0.10
Weakly sup.	k-means	56.15 ± 0.62	41.46 ± 0.53
	Encoder Fusion SiamKPConv (batch size 10)	29.03 ± 22.46	12.84 ± 18.49
	Encoder Fusion SiamKPConv (batch size 2)	53.09 ± 3.73	36.60 ± 3.18
	DC3DCD *Siamese KPConv*	28.28 ± 3.73	14.43 ± 3.70
	DC3DCD *Encoder Fusion SiamKPConv*	52.30 ± 2.41	37.75 ± 2.11
	DC3DCD *Siamese KPConv* (with input features)	54.91 ± 5.45	42.27 ± 6.64
	DC3DCD *Encoder Fusion SiamKPConv* (with input features)	**68.45** ± 1.10	**57.06** ± 0.41

Table 4.5: **Quantitative evaluation of DC3DCD on Urb3DCD-V2 low density LiDAR dataset.** *Top*: supervised methods. KPConv based methods refer to the proposed ones in Chapter 3. DSM-based methods are adaptation of Daudt et al. (2018) networks to DSM inspired by Zhang et al. (2019) and RF refers to Random Forests. *Middle*: Weakly supervised methods with k-means and *Encoder Fusion SiamKPConv* results using 7 training cylinders in the training and validation set (equivalent to about 50,000 annotated points). *Bottom*: Weakly supervised methods with our proposed DC3DCD evaluated in 4 different settings: with *Siamese KPConv* or *Encoder Fusion SiamKPConv* architectures and with or without the addition of 10 hand-crafted features as input to the network.

Two different examples are given for a qualitative assessment of the method in Figures 4.12 and 4.13. As visible in Figure 4.13, main changes (e.g., new buildings or demolitions) seem quite well retrieved by the k-means and both DC3DCD configurations. However, when going more into details, some misclassifications can be seen on new building facades (Figure 4.12) or vegetation. For new building facades, a slight improvement over k-means is reached by DC3DCD, but it is still not perfect. The k-means technique has the same tendency as the RF method (see Chapter 3 Figure 3.8) and confuses small new buildings with new vegetation, surely because they have the same height as visible in Figure 4.12. As depicted in Table 4.6, main difficulties of the DC3DCD method concern vegetation growth and missing vegetation. Note that this was already the most difficult classes in the supervised context. The missing vegetation is almost always confused with demolition in DC3DCD with hand-crafted input features and the *Encoder Fusion SiamKPConv* architecture. This is even worse with the k-means and missing vegetation is never predicted with DC3DCD without hand-crafted input features. However, this make sense, since the 'missing vegetation' and 'demolition' classes are both negative changes. Surprisingly, mobile objects are quite well retrieved, especially for DC3DCD with hand-crafted input features and the *Encoder Fusion SiamKPConv* architecture (Table 4.6).

	Method	Per class IoU (%)						
		Unchanged	New building	Demolition	New veg.	Veg. growth	Missing veg.	Mobile Object
Supervised	SKPConv	95.82 ± 0.48	86.68 ± 0.47	78.66 ± 0.47	93.16 ± 0.27	65.17 ± 1.37	65.46 ± 0.93	91.55 ± 0.60
	EFSKPConv	**97.47** ± 0.04	**96.68** ± 0.30	**82.29** ± 0.16	**96.52** ± 0.03	**67.76** ± 1.51	**73.50** ± 0.81	**94.37** ± 0.54
	DSM-Siamese	93.21 ± 0.11	86.14 ± 0.65	69.85 ± 1.46	70.69 ± 1.35	8.92 ± 15.46	60.71 ± 0.74	8.14 ± 5.42
	DSM-FC-EF	94.39 ± 0.12	91.23 ± 0.31	71.15 ± 0.99	68.56 ± 3.92	1.89 ± 2.82	62.34 ± 1.23	46.70 ± 3.49
	RF	92.72 ± 0.01	73.16 ± 0.02	64.60 ± 0.06	75.17 ± 0.06	19.78 ± 0.30	7.78 ± 0.02	73.71 ± 0.63
Weakly sup.	k-means	91.82 ± 0.05	70.46 ± 0.25	59.83 ± 0.11	59.20 ± 0.48	6.00 ± 0.19	0.00 ± 0.00	53.26 ± 3.19
	EFSKPConv (b. s. 10)	58.38 ± 46.55	13.22 ± 16.83	26.15 ± 23.54	11.04 ± 19.12	0.41 ± 0.71	7.29 ± 11.99	19.48 ± 33.46
	EFSKPConv (b. s. 2)	89.88 ± 0.53	29.26 ± 16.83	48.66 ± 2.57	52.91 ± 9.19	7.59 ± 6.40	14.35 ± 15.16	66.84 ± 5.42
	DC3DCD SKPConv	84.51 ± 0.70	13.33 ± 3.05	29.50 ± 13.19	40.31 ± 9.15	3.03 ± 1.80	0.08 ± 0.01	0.33 ± 0.29
	DC3DCD EFSKPConv	90.90 ± 0.79	64.06 ± 5.13	54.35 ± 3.84	58.14 ± 20.03	1.45 ± 2.05	0.94 ± 0.78	47.57 ± 2.58
	DC3DCD SKPConv (i. f.)	92.90 ± 0.21	76.61 ± 2.09	67.22 ± 2.63	61.33 ± 10.07	8.66 ± 6.54	16.39 ± 14.95	23.44 ± 40.54
	DC3DCD EFSKPConv (i. f.)	**93.96** ± 0.11	**79.26** ± 0.68	**67.88** ± 0.49	**75.34** ± 2.81	**19.48** ± 4.00	**20.29** ± 2.90	**80.10** ± 3.16

Table 4.6: **Per-class IoU scores of DC3DCD on Urb3DCD-V2 low density LiDAR dataset.** *Top*: supervised methods. KPConv based methods refer to the proposed ones in Chapter 3. DSM-based methods are adaptation of Daudt et al. (2018) networks to DSM inspired by Zhang et al. (2019) and RF refers to Random Forests. *Middle*: Weakly supervised methods with k-means and *Encoder Fusion SiamKPConv* results using 7 training cylinders in the training and validation set (equivalent to about 50,000 annotated points). *Bottom*: Weakly supervised methods with our proposed DC3DCD evaluated in 4 different settings: with *Siamese KPConv* (SKPConv) or *Encoder Fusion SiamKPConv* (EFSKPConv) architectures and with or without the addition of 10 hand-crafted features as input to the network. Veg. stands for vegetation; b. s. for batch size; i.f. for input features.

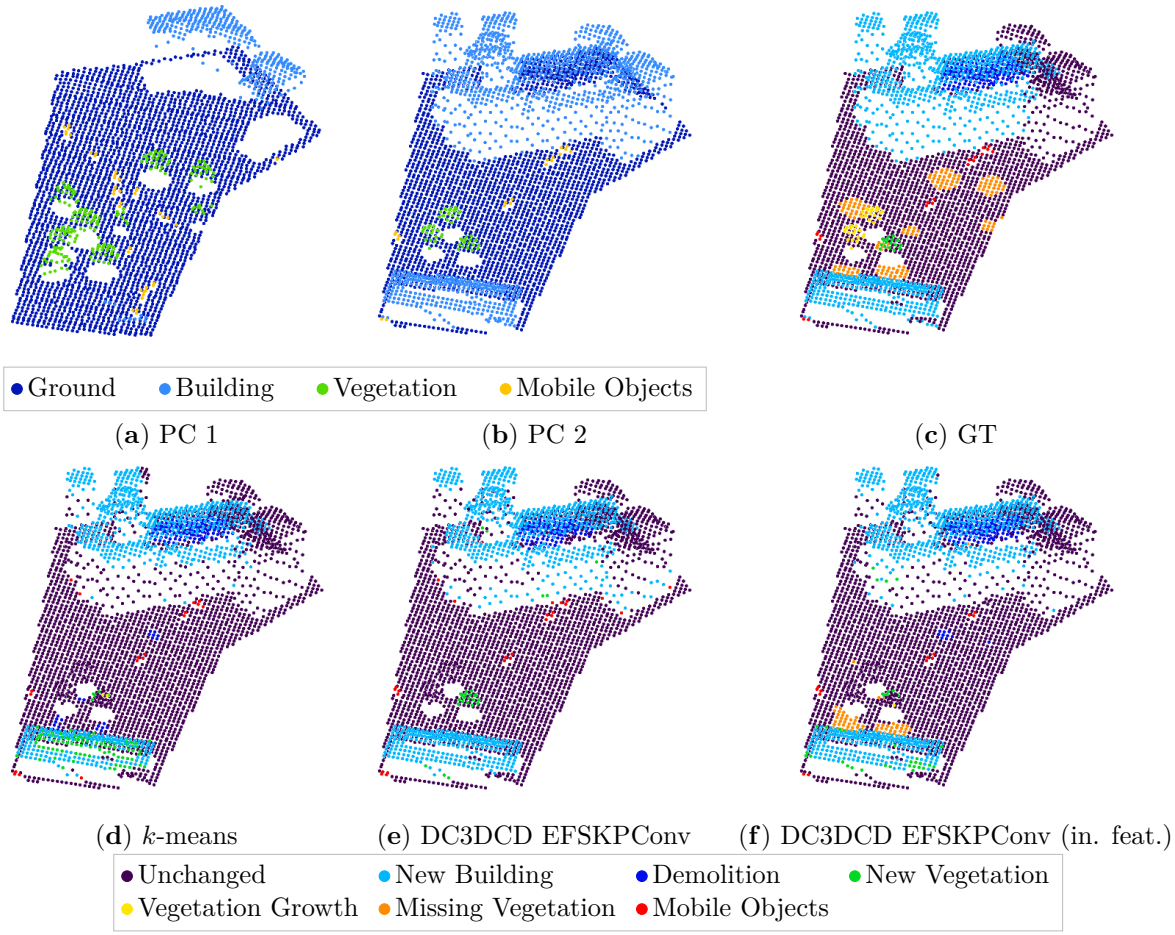


Figure 4.12: **Visual change detection results on Urb3DCD-V2 low density LiDAR sub-dataset (area 1):** (**a**-**b**) the two input point clouds; (**c**) ground truth (GT): simulated changes; (**d**) *k*-means results; (**e**) DC3DCD with the *Encoder Fusion SiamKPConv* architecture results; (**f**) DC3DCD with the *Encoder Fusion SiamKPConv* architecture and the addition of 10 hand-crafted features as input results.

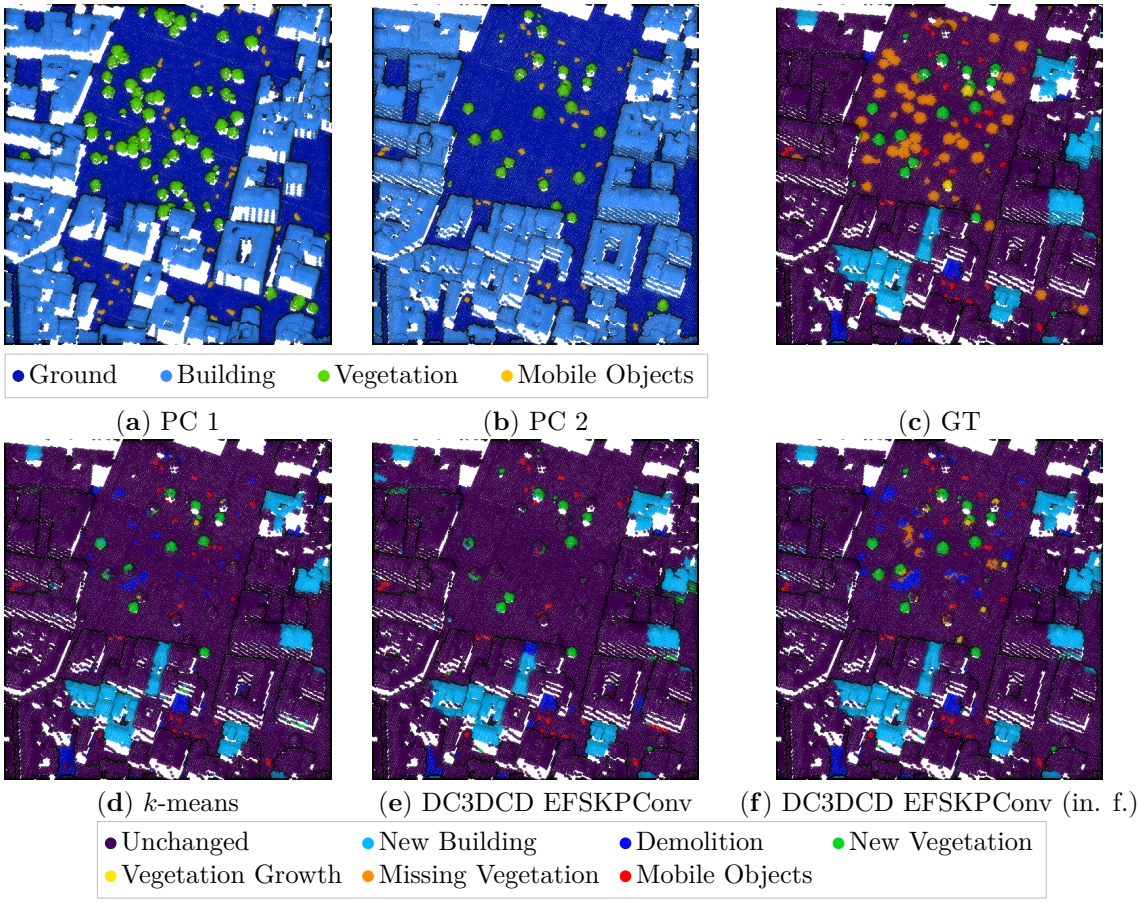


Figure 4.13: **Visual change detection results on Urb3DCD-V2 low density LiDAR sub-dataset (area 2):** (**a**-**b**) the two input point clouds; (**c**) ground truth (GT): simulated changes; (**d**) k-means results; (**e**) DC3DCD with the *Encoder Fusion SiamKPConv* architecture results; (**f**) DC3DCD with the *Encoder Fusion SiamKPConv* architecture and 10 hand-crafted input features.

4.4.2.4 Results on real AHN-CD dataset

Concerning the real AHN-CD dataset, quantitative results on the manually annotated testing set are given in Table 4.7, and Table 4.8 for per class results. For comparison purpose, we also provide results of supervised methods. However, we recall that they have been trained on the semi-automatically annotated AHN-CD dataset containing several ground truth errors (see Chapters 1 and 3). This explains lower results of the RF compared to the k-means which have been mapped onto real classes using the manually annotated set (as for DC3DCD method). As for the simulated dataset, we can see that DC3DCD provides better results than the k-means algorithm. Figure 4.14 shows that main changes are well retrieved for both weakly supervised methods. However, in the k-means results, larger objects of the clutter class such as trucks are mixed up with buildings. There are also lots of misclassifications in unchanged vegetation and unchanged building facades (see region of interest in Figure 4.14f). As far as DC3DCD is concerned, unchanged vegetation is well classified. A few mistakes are visible in some 'new clutter' objects. We recall that this class is a mix of a lot of objects, from vegetation to cars or garden sheds, surely explaining why its classification score is lower. Complementary results on a larger AHN-CD testing tile are shown in Figure 4.15. The ground truth is given by the semi-automatic process detailed in Chapter 1 (Section 1.2). The mapping onto the real classes is performed using this ground truth for both k-means method and DC3DCD. As visible in this example, most of 'new clutter' class objects are omitted or mixed up with the new building class, also the demolition class is totally omitted by the k-means algorithm (Figure 4.15d). In the DC3DCD results in Figure 4.15e, clutter class seems better retrieved, even though it is not perfect implying the main differences with the ground truth (Figure 4.15g). In the areas of interest depicted by the black rectangles in Figure 4.15, we observe that DC3DCD seems to better adapt to the user context (i.e., by the ground truth defined by the user) than the k-means, even though the same ground truth-guided mapping step has been performed. Indeed, here buildings are not set as new in the ground truth and in DC3DCD, conversely to k-means. Finally, on this tile, if we compare to the ground truth, DC3DCD obtains 55.91% of mIoU_{ch}, while the k-means only 24.63%.

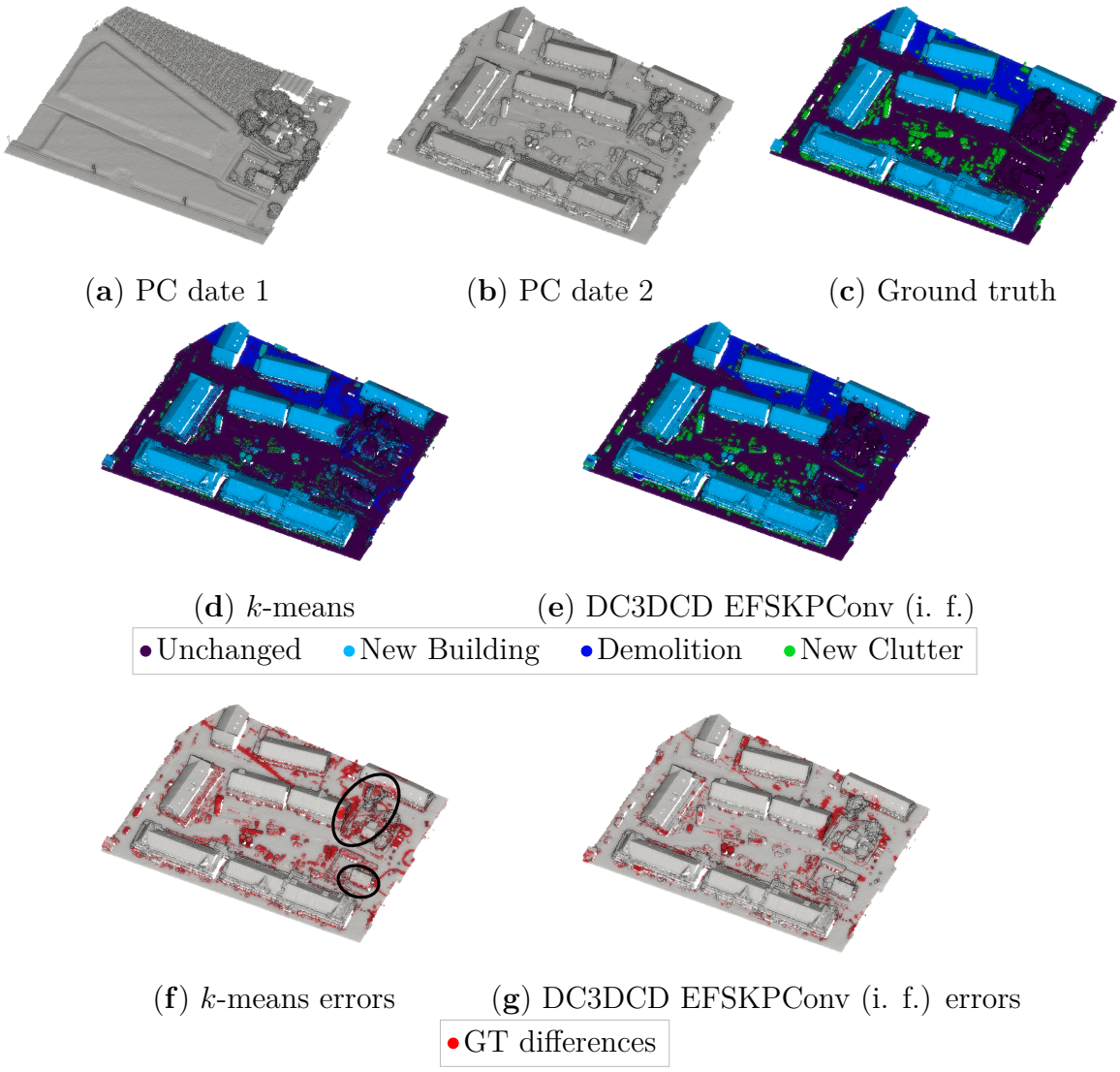
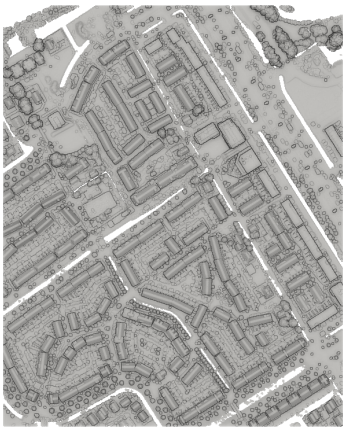
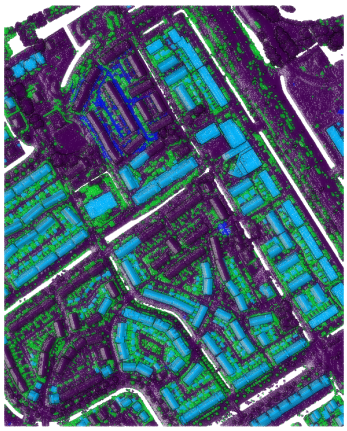


Figure 4.14: **Qualitative results on the manually annotated sub-part of AHN-CD dataset:** (**a**-**b**) PCs at date 1 and 2; (**c**) ground truth; k-means results (**d**) and errors (**f**); DC3DCD results (**e**) and errors (**g**) using the *Encoder Fusion SiamKPConv* architecture and the 10 hand-crafted features in input. Regions of interest specifically discussed in the text are highlighted with ellipses.

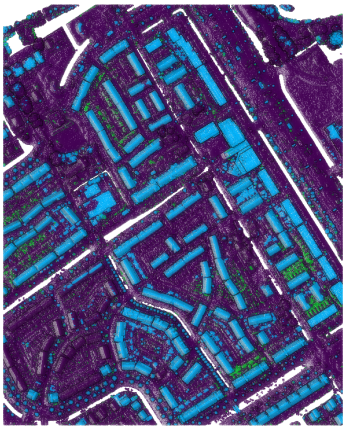
(**a**) PC date 1



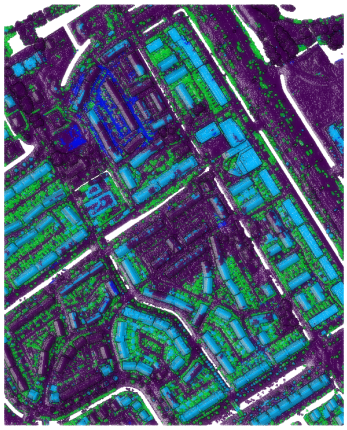
(**b**) PC date 2



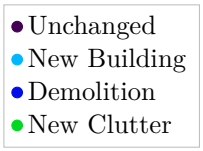
(**c**) Ground truth

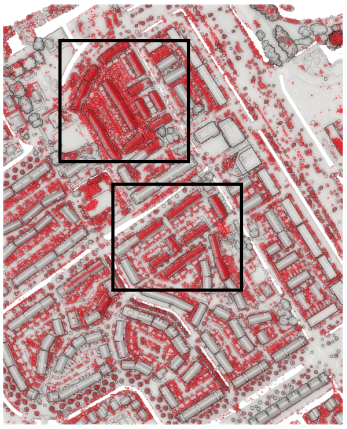


(**d**) k-means

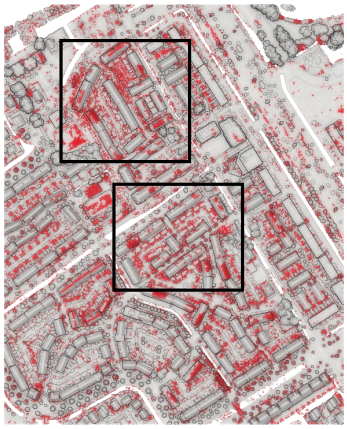


(**e**) DC3DCD EFSKPConv (i. f.)





(**f**) k-means errors



(**g**) DC3DCD EFSKPConv (i. f.) errors

Figure 4.15: **Qualitative results on the semi-automatically annotated AHN-CD dataset:** (**a-b**) PCs at date 1 and 2; (**c**) ground truth; k-means results (**d**) and corresponding errors (**f**); DC3DCD results (**e**) and corresponding errors (**g**) using the *Encoder Fusion SiamKPConv* architecture and the 10 hand-crafted features in input. Regions of interest specifically discussed in the text are highlighted with rectangles.

	Method	mAcc (%)	mIoU$_{ch}$ (%)
Supervised	Siamese KPConv	85.65 ± 1.55	72.95 ± 2.05
	Encoder Fusion SiamKPConv	**90.26** ± 0.22	**75.00** ± 0.74
	DSM-Siamese	50.87 ± 1.15	30.96 ± 2.48
	DSM-Pseudo-Siamese	70.71 ± 5.09	48.85 ± 7.03
	DSM-FC-EF	71.47 ± 1.43	45.57 ± 0.98
	RF	47.94 ± 0.02	29.45 ± 0.02
WS	k-means	70.07 ± 0.56	53.12 ± 0.79
	DC3DCD *Encoder Fusion SiamKPConv* (10 in. feat.)	**83.18** ± 1.10	**66.69** ± 2.19

Table 4.7: **Qualitative assessment of DC3DCD on the manually annotated subpart of AHN-CD dataset.** *Top*: supervised methods. KPConv-based methods refer to the proposed one in Chapter 3. DSM-based methods are adaptation of Daudt et al. (2018) networks to DSM inspired by Zhang et al. (2019) and RF refers to Random Forests. In supervised settings, the training is performed on the semi-automatically annotated AHN-CD dataset containing some errors (see Chapters 1 and 3). *Bottom*: Weakly supervised methods with k-means and our proposed DC3DCD with *Encoder Fusion SiamKPConv* architecture and with the addition of 10 hand-crafted features as input to the network.

	Method	Per class IoU (%)			
		Unchanged	New building	Demolition	New clutter
Supervised	Siamese KPConv	89.75 ± 2.18	82.77 ± 5.38	86.44 ± 0.88	**46.65** ± 0.16
	Encoder Fusion SiamKPConv	**94.79** ± 0.34	**95.31** ± 1.95	**88.87** ± 1.59	41.16 ± 1.30
	DSM-Siamese	77.10 ± 1.51	76.77 ± 0.79	4.91 ± 8.33	11.20 ± 1.71
	DSM-Pseudo-Siamese	78.00 ± 5.09	75.32 ± 8.59	47.46 ± 11.92	23.76 ± 0.56
	DSM-FC-EF	70.77 ± 1.13	90.32 ± 0.61	30.58 ± 1.76	15.81 ± 0.81
	RF	78.24 ± 0.00	74.64 ± 0.03	0.00 ± 0.00	13.72 ± 0.06
WS	*k*-means	84.13 ± 0.49	83.13 ± 0.89	55.40 ± 0.50	20.84 ± 1.00
	DC3DCD *Encoder Fusion SiamKPConv* (10 input features)	**91.34** ± 1.21	**89.91** ± 0.72	**69.52** ± 4.97	**40.63** ± 0.97

Table 4.8: **Per class IoU DC3DCD results on the manually annotated testing part of AHN-CD dataset** given in %. For both supervised and weakly supervised methods, the best results are shown in bold. *Top:* supervised methods. KPConv-based methods refer to the proposed one in Chapter 3. DSM-based methods are adaptation of Daudt et al. (2018) networks to DSM inspired by Zhang et al. (2019) and RF refers to Random Forests. In supervised settings, the training is performed on the semi-automatically annotated AHN-CD dataset containing some errors (see Chapters 1 and 3). *Bottom:* Weakly supervised methods with *k*-means and our proposed DC3DCD with *Encoder Fusion SiamKPConv* architecture and with the addition of 10 hand-crafted features as input to the network.

4.4.3 Discussion

In this section, we have proposed an unsupervised change detection method with a weakly user-guided mapping to real classes providing interesting results. This problem is still open and complex and in the following, we point out some observations and discussions about possible improvements.

Importance of network's architectures and input features

We saw in the result section that the choice of the back-bone architecture and the addition of hand-crafted features as input along with 3D point coordinates are crucial. This is in agreement with the original publication of DeepCluster, where authors provided gradient of images as input to obtain interesting results (Caron et al., 2018). These results in a weakly supervised context also emphasizes conclusions of Chapter 3 on the necessity of applying convolution on features difference. To explain this, let us note that the unsupervised context is a largely unconstrained problem. While the annotation allows counterbalancing architectures weaknesses, this is indeed no longer possible for the unsupervised setting. Thereby the choice of an architecture that more specially extracts change-related features through convolutions of difference of features from both inputs at multiple scales, and the addition of well-designed hand-crafted features, allows guiding the training of the network toward a relevant minimum, leading to a reliable change segmentation.

Comparison with binary change detection methods

Previously in this chapter, we proposed to decrease the need for annotation by using transfer learning from simulated data to real data (see Section 4.2), or by presenting some unsupervised methods based on DCVA for binary change detection (see Section 4.3). In Table 4.9, we propose to compare all the developed methods on the manually annotated subpart of AHN-CD in a binary change segmentation setup. DC3DCD results presented here are obtained using the user-guided mapping directly to the changed and unchanged classes. As visible in these quantitative results, DC3DCD is much more accurate than other fully-unsupervised methods, however it requires 1,000 annotation for the mapping of the pseudo-clusters to the real classes. This weakly supervised setting might be a compromise between high results of supervised methods, but obtained with millions of annotated points, and lower results of unsupervised methods without any annotation at all.

	Method	mAcc (%)	IoU (%) Unchanged	IoU (%) Changed
Sup.	Siamese KPConv	**97.08**	**95.39**	**92.95**
	Encoder Fusion SiamKPConv	96.75	94.79	92.10
Unsup.	SSL-DCVA	85.20	78.91	69.38
	SSST-DCVA	81.88	70.02	63.85
	Siam. KPConv transfer	79.29	77.89	58.89
	M3C2 (Lague et al., 2013)	51.77	3.66	39.90
	C2C (Girardeau-Montaut et al., 2005)	76.67	76.98	53.34
W. Sup	DC3DCD *Encoder Fusion SiamKPConv* (10 in. feat.)	**94.43**	**91.24**	**86.96**

Table 4.9: **Comparison of all methods proposed in this chapter on the manually annotated sub-part of AHN-CD dataset.**

Improving DC3DCD with contrastive learning

As mentioned, the problem in an unsupervised setting is to train a network to extract appropriate features for a specific task. In a task involving comparison of similar and dissimilar data (like change detection task), the contrastive loss is often used to force the network to extract identical features for similar data. Therefore, an idea can be to force the network to predict similar features for unchanged areas. To test this principle, we propose to introduce the following contrastive term in the loss function:

$$\mathcal{L}_{contrastive} = 0.5 \times y_{sim} \times F_{CD}^2 \quad \text{with} \quad y_{sim} = \begin{cases} 1 & \text{if similar} \\ 0 & \text{else.} \end{cases} \tag{4.17}$$

where F_{CD} is the L_2-norm of output features and y_{sim} is the similarity term (set to 1 for unchanged points, and 0 elsewhere). Using this contrastive term in the loss aims at forcing to 0 change-related features in unchanged areas. To test this idea, we combine the contrastive loss in Equation 4.17 with the deep clustering loss (NLL using the pseudo-label, Equation 3.6) taking the mean, and train the *Encoder Fusion SiamKPConv* network since it gave the best results.

We first carried out experiments by taking the similarity y_{sim} from the ground truth (as y_{sim} is not available in practice, we first test the idea by taking real values of y_{sim}). Results were really interesting since, as visible in Tables 4.10 and 4.11, DC3DCD reached 73.51% of mIoU$_{ch}$ without the use of hand-crafted features and 82.63% of mIoU$_{ch}$ with the use of hand-crafted features on Urb3DCD-V2-1 dataset. We recall that on this dataset and in a fully supervised setting *Siamese KPConv* and *Encoder Fusion SiamKPConv*

networks obtained 80.12% and 85.19% of mIoU_{ch} respectively. Thus, the addition of the contrastive part allows meeting fully supervised results (in an ideal case where y_{sim} is known).

This first experience validated the idea of using the contrastive loss. However in practice, y_{sim} needs to be estimated. To obtain the similarity y_{sim}, we first used the significant changes given by M3C2 or a binary thresholding of C2C distance. Results are mitigated (see Tables 4.10 and 4.11) since the best results obtained using M3C2 for y_{sim} allow us to improve by only 2 points DC3DCD without hand-crafted input features. With hand-crafted input features, results are worsened when the contrastive term is added during the training (using y_{sim} based on M3C2, obtained mIoU_{ch} is 46.42%).

Another idea is to rely on multi-task learning (Vandenhende et al., 2021; Zhang and Yang, 2021): a multi-task framework based on DC3DCD that jointly extracts mono-date features that can be used for similarity computation has been designed. As illustrated in Figure 4.16, we added decoders for mono-date semantic segmentation to the backbone architectures to also obtain semantic segmentation of PCs. Both *Siamese KPConv* and *Encoder Fusion SiamKPConv* have encoders to extract mono-date features, therefore we just added a decoder taking as input these mono-date features instead of feature differences for *Siamese KPConv* for example. Thereby, we used the same idea as before to train the network but with two separate clusterings, performed on output features of the change decoder on one side, as in previous experiments, and on output features of mono-date decoders on the other side. This results in both change pseudo-labels and mono-date pseudo-labels which are used to modulate change encoder-decoder and mono-date encoders-decoders respectively. In practice, we shared trainable parameters between mono-date encoders and decoders. We trained first the semantic segmentation part and then apply a binary clustering (using k-means) on the nearest point mono-date features difference to obtain the similarity y_{sim} used in the contrastive term of the change detection loss. Concerning, the number of pseudo-clusters for mono-date $K_{mono-date}$, 4 and 500 have been tested (4 is the number of semantic segmentation classes in Urb3DCD-V2 dataset, 500 is considered as a sample large bound). While semantic segmentation scores, using the same user-guided mapping for mono-date semantic segmentation, are very promising (90.79% of mIoU on the 4 semantic segmentation classes of Urb3DCD-V2 with hand-crafted input features and $K_{mono-date} = 500$), using the associated y_{sim} still leads to unsatisfactory results (see Tables 4.10 and 4.11). Indeed, when $K_{mono-date}$ is set to 500, we obtain 50.14% of mIoU_{ch} (with hand-crafted features) which is better than with distance-

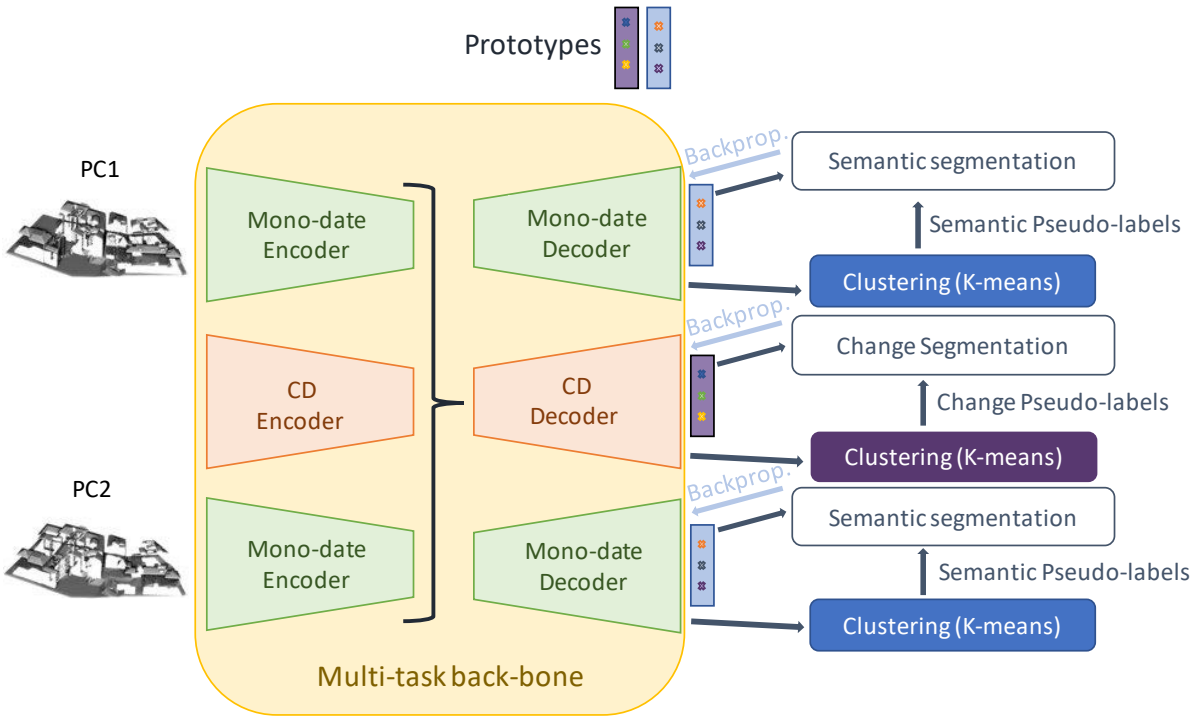


Figure 4.16: **DC3DCD-V2 using multi-task learning.**

based methods (M3C2 or C2C) but worse than without the contrastive part of the loss. Thereby, computing similarity from the nearest point mono-date features difference does not seem adapted. Two main reasons for this non-success can be advanced: i) the nearest point comparison faces finally the same problems as DCVA method (namely problems in occlusion and dense urban areas, see Section 4.3.2.3), and ii) the method predicts 500 different mono-date semantic classes (far more than existing real classes) and nothing forces that two clusters that are near in semantics (e.g., belonging to the same real class) are near in the feature space, leading to differences of features likely to be very high. To counterbalance this last point, we tested using only 4 mono-date pseudo-clusters, but results are even worse, probably due to the fact that the DeepCluster strategy to train a network required a number of pseudo-clusters greater than the number of real classes.

Qualitative assessment of these results is supported by Figures 4.17 and 4.18. When the similarity comes from the binary change ground truth, visual results really lookalike the multi-change ground truth (Figures 4.17f and 4.18c). Qualitative results of multi-task learning are quite encouraging. Surprisingly, borders of 'missing vegetation' seems well

retrieved, but the center is still confused with demolition.

Concerning the real dataset AHN-CD, the same types of results are obtained as detailed in Tables 4.12 and 4.13. When the similarity comes from the binary change ground truth, we obtained results approaching the supervised setting. Since on this dataset, we have obtained better results for binary change detection with our deep unsupervised methods than with existing distance-based methods, we decided to take the similarity provided by SSL-DCVA (Section 4.3.1.2). However, this leads to worse results than without the contrastive term in the loss, surely because SSL-DCVA results are still far from perfect.

All these experiments aim at evaluating the potential of contrastive losses to improve our unsupervised results. Results with the similarity y_{sim} issued from ground truth are very promising since they reach comparable results than the fully supervised networks. However, the method is highly dependent on the quality of this binary change annotation, and in case of mitigated binary annotation, it worsens DC3DCD results. These first perspective experimentations are, to our opinion, encouraging to limit the annotation effort while preserving interesting results. Our future work will concentrate on the estimation of precise y_{sim}.

	Method	i. f.	y_{sim}	mAcc (%)	mIoU$_{ch}$ (%)
Sup.	SKPConv			91.21 ± 0.68	80.12 ± 0.02
	EFSKPConv			**94.23** ± 0.88	**85.19** ± 0.24
Weakly supervised	DC3DCD EFSKPConv			52.30 ± 2.41	37.75 ± 2.11
	DC3DCD-V2 EFSKPConv		GT	**83.45** ± 2.22	**73.51** ± 3.74
	DC3DCD-V2 EFSKPConv		M3C2	54.01 ± 3.54	39.59 ± 4.18
	DC3DCD-V2 EFSKPConv		C2C	39.74 ± 1.84	25.57 ± 1.97
	DC3DCD-V2 EFSKPConv		Multi-task ($K_{seg.sem.} = 4$)	34.31 ± 5.85	19.21 ± 5.81
	DC3DCD-V2 EFSKPConv		Multi-task ($K_{seg.sem.} = 500$)	47.62 ± 6.76	32.07 ± 6.15
	DC3DCD EFSKPConv	✓		68.45 ± 1.10	57.06 ± 0.41
	DC3DCD-V2 EFSKPConv	✓	GT	**89.04** ± 0.70	**82.63** ± 0.73
	DC3DCD-V2 EFSKPConv	✓	M3C2	58.80 ± 2.14	46.42 ± 2.45
	DC3DCD-V2 EFSKPConv	✓	C2C	42.01 ± 0.67	28.04 ± 0.60
	DC3DCD-V2 EFSKPConv	✓	Multi-task ($K_{seg.sem.} = 4$)	53.04 ± 8.22	38.90 ± 8.51
	DC3DCD-V2 EFSKPConv	✓	Multi-task ($K_{seg.sem.} = 500$)	62.95 ± 1.81	50.14 ± 3.85

Table 4.10: **Quantitative evaluation of DC3DCD-V2 on Urb3DCD-V2 low density LiDAR dataset.** *Top*: supervised methods. KPConv based methods refer to the proposed ones in Chapter 3. *Middle*: Weakly supervised methods with our proposed DC3DCD and DC3DCD-V2 with Encoder Fusion SiamKPConv architecture without the addition of 10 hand-crafted features as input to the network. *Bottom*: Weakly supervised methods with our proposed DC3DCD and DC3DCD-V2 with Encoder Fusion SiamKPConv architecture and with the addition of 10 hand-crafted features (i. f.) as input to the network.

	Method	i. f.	y_{sim}	Per class IoU (%)						
				Unchanged	New building	Demolition	New veg.	Veg. growth	Missing veg.	Mobile Object
Sup.	SKPConv			95.82 ± 0.48	86.68 ± 0.47	78.66 ± 0.47	93.16 ± 0.27	65.17 ± 1.37	65.46 ± 0.93	91.55 ± 0.60
	EFSKPConv			**97.47** ± 0.04	**96.68** ± 0.30	**82.29** ± 0.16	**96.52** ± 0.03	**67.76** ± 1.51	**73.50** ± 0.81	**94.37** ± 0.54
Weakly supervised	DC3DCD			90.90 ± 0.79	64.06 ± 5.13	54.35 ± 3.84	58.14 ± 20.03	1.45 ± 2.05	0.94 ± 0.78	47.57 ± 2.58
	DC3DCD-V2		GT	**97.28** ± 0.10	**93.66** ± 1.30	**75.24** ± 4.34	**83.78** ± 6.00	**49.52** ± 7.00	**53.60** ± 11.98	**85.28** ± 3.54
	DC3DCD-V2		M3C2	93.02 ± 0.03	75.05 ± 5.28	57.43 ± 2.03	69.02 ± 5.77	10.45 ± 8.60	4.47 ± 4.22	21.11 ± 10.22
	DC3DCD-V2		C2C	90.73 ± 0.45	64.93 ± 3.79	56.52 ± 3.74	10.76 ± 5.80	0.41 ± 0.56	0.44 ± 0.51	20.37 ± 6.96
	DC3DCD-V2		Multi-task ($K_{seg.sem.} = 4$)	87.15 ± 2.38	29.55 ± 16.82	39.72 ± 7.84	28.11 ± 2.95	0.00 ± 0.00	0.08 ± 0.09	17.78 ± 11.09
	DC3DCD-V2		Multi-task ($K_{seg.sem.} = 500$)	88.56 ± 2.11	40.44 ± 12.79	45.16 ± 12.73	50.12 ± 6.53	4.03 ± 5.09	0.43 ± 1.19	51.93 ± 6.90
	DC3DCD	✓		93.96 ± 0.11	79.26 ± 0.68	67.88 ± 0.49	75.34 ± 2.81	19.48 ± 4.00	20.29 ± 2.90	80.10 ± 3.16
	DC3DCD-V2	✓	GT	**97.73** ± 0.05	**94.50** ± 0.20	**81.10** ± 0.38	**92.22** ± 0.92	**61.32** ± 2.39	**73.39** ± 1.18	**90.12** ± 1.30
	DC3DCD-V2	✓	M3C2	93.17 ± 0.13	77.48 ± 1.85	65.34 ± 0.55	76.96 ± 5.26	25.65 ± 4.94	0.31 ± 0.54	32.76 ± 4.15
	DC3DCD-V2	✓	C2C	92.25 ± 0.15	70.01 ± 2.77	66.56 ± 1.12	27.11 ± 5.78	0.00 ± 0.00	4.59 ± 0.64	0.00 ± 0.00
	DC3DCD-V2	✓	Multi-task ($K_{seg.sem.} = 4$)	91.96 ± 1.10	69.01 ± 7.24	63.07 ± 1.49	42.76 ± 13.31	4.11 ± 6.71	17.23 ± 7.28	31.20 ± 25.63
	DC3DCD-V2	✓	Multi-task ($K_{seg.sem.} = 500$)	93.68 ± 0.48	78.23 ± 2.94	66.02 ± 1.46	71.49 ± 2.49	0.00 ± 0.00	16.07 ± 4.77	69.06 ± 16.59

Table 4.11: **Per class quantitative evaluation of DC3DCD-V2 on Urb3DCD-V2 low density LiDAR dataset.** *Top:* supervised methods. KPConv based methods refer to the proposed ones in Chapter 3. *Middle:* Weakly supervised methods with our proposed DC3DCD and DC3DCD-V2 with Encoder Fusion SiamKPConv architecture without the addition of 10 hand-crafted features as input to the network. *Bottom:* Weakly supervised methods with our proposed DC3DCD and DC3DCD-V2 with Encoder Fusion SiamKPConv architecture and with the addition of 10 hand-crafted features (i. f.) as input to the network.

	Method	i.f	y_{sim}	mAcc (%)	mIoU$_{ch}$ (%)
Sup.	SKPConv			85.65 ± 1.55	72.95 ± 2.05
	EFSKPConv			**90.26** ± 0.22	**75.00** ± 0.74
W. sup.	DC3DCD EFSKPConv	✓		83.18 ± 1.10	66.69 ± 2.19
	DC3DCD-V2 EFSKPConv	✓	GT	**84.43** ± 1.13	**70.78** ± 2.22
	DC3DCD-V2 EFSKPConv	✓	SSL-DCVA	78.53 ± 0.83	64.70 ± 1.27

Table 4.12: **Quantitative evaluation of DC3DCD-V2 on AHN-CD dataset (manually annotated part).** *Top*: Siamese KPConv (SKPConv) and Encoder Fusion SiamKPConv (EFSKPConv) supervised methods proposed in Chapter 3. *Bottom*: Weakly supervised methods with our proposed DC3DCD and DC3DCD-V2 with Encoder Fusion SiamKPConv (EFSKPConv) architecture and with the addition of 10 hand-crafted features (i. f.) as input to the network.

				Per class IoU (%)			
	Method	i.f	y_{sim}	Unchanged	New building	Demolition	New clutter
Sup.	SKPConv			89.75 ± 2.18	82.77 ± 5.38	86.44 ± 0.88	**46.65** ± 0.16
	EFSKPConv			**94.79** ± 0.34	**95.31** ± 1.95	**88.87** ± 1.59	41.16 ± 1.30
W. sup.	DC3DCD	✓		91.34 ± 1.21	89.91 ± 0.72	69.52 ± 4.97	40.63 ± 0.97
	DC3DCD-V2	✓	GT	**95.03** ± 0.98	**92.31** ± 0.62	78.86 ± 6.62	**41.18** ± 0.18
	DC3DCD-V2	✓	SSL-DCVA	89.90 ± 0.81	83.71 ± 1.55	**80.91** ± 4.24	29.41 ± 0.72

Table 4.13: **Per class quantitative evaluation of DC3DCD-V2 on AHN-CD dataset (manually annotated part).** *Top*: Siamese KPConv (SKPConv) and Encoder Fusion SiamKPConv (EFSKPConv) supervised methods proposed in Chapter 3. *Bottom*: Weakly supervised methods with our proposed DC3DCD and DC3DCD-V2 with Encoder Fusion SiamKPConv architecture and with the addition of 10 hand-crafted features (i. f.) as input to the network.

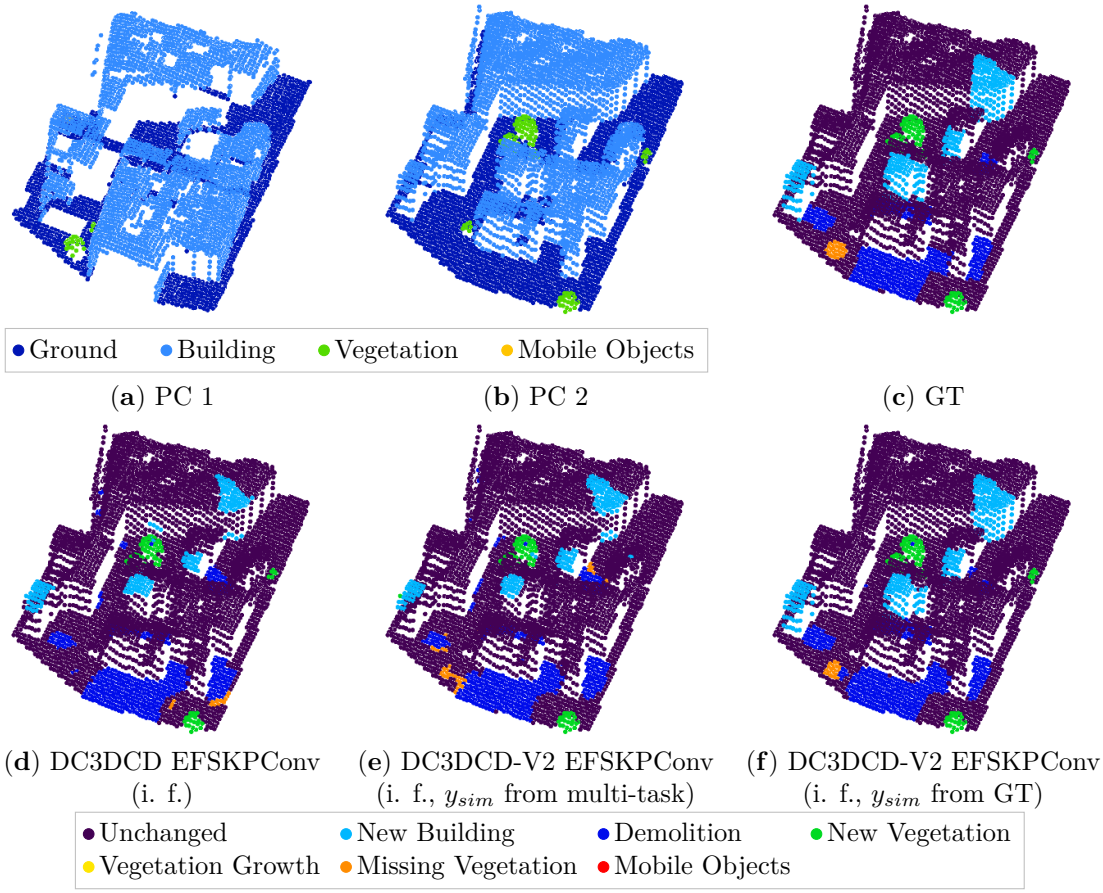
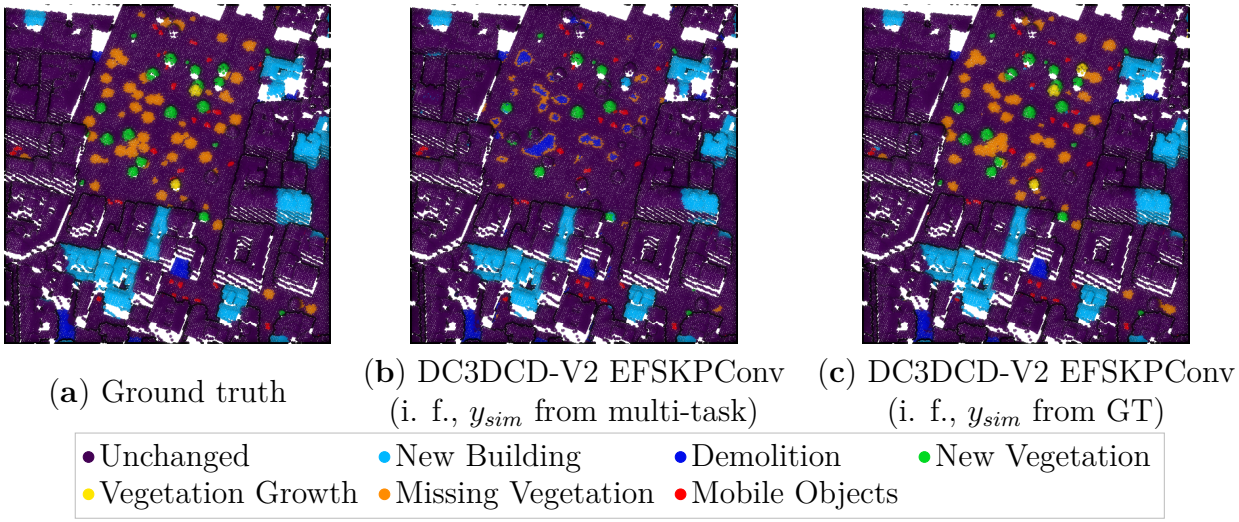


Figure 4.17: **Visual change detection results on Urb3DCD-V2 low density LiDAR sub-dataset:** (**a**-**b**) the two input point clouds; (**c**) ground truth (GT): simulated changes; (**d**) DC3DCD with the *Encoder Fusion SiamKPConv* architecture and 10 hand-crafted input features (i. f.) results; (**e**) DC3DCD-V2 with the *Encoder Fusion SiamKPConv* architecture, 10 hand-crafted input features results and the similarity y_{sim} computed from the multi-task configuration($K_{mono-date} = 500$); (**f**) DC3DCD-V2 with the *Encoder Fusion SiamKPConv* architecture, 10 hand-crafted input features results and the similarity y_{sim} computed from the ground truth (GT).



(**a**) Ground truth (**b**) DC3DCD-V2 EFSKPConv (i. f., y_{sim} from multi-task) (**c**) DC3DCD-V2 EFSKPConv (i. f., y_{sim} from GT)

Figure 4.18: **Visual change detection results on Urb3DCD-V2 low density LiDAR sub-dataset (area 2):** (**a**) ground truth (GT): simulated changes; (**b**) DC3DCD-V2 with the *Encoder Fusion SiamKPConv* architecture, 10 hand-crafted input features (i. f.) and the similarity y_{sim} computed from the multi-task configuration($K_{mono-date} = 500$); (**c**) DC3DCD-V2 with the *Encoder Fusion SiamKPConv* architecture, 10 hand-crafted input features and the similarity y_{sim} computed from the ground truth (GT). For comparison, one can refer to Figure 4.13 providing k-means and DC3DCD results over the same area.

4.5 Conclusion

In this chapter, we focused on how to **reduce the number of annotated data** for 3D PCs change detection task. To do so, we first proposed to evaluate the **transfer learning performances of *Siamese KPConv*** deep network compared to other deep networks on DSM or to the RF algorithm. Then, we also evaluated the **benefit of pre-training the network on simulated dataset to decrease the size of training set needed on the real data**. Thanks to pre-training, only less than 1/3000 of cylinders from the target domain are needed to reach the maximal score. It significantly reduces the burden of manual annotation.

In a second time, we proposed two **fully unsupervised methods to tackle 3D binary change detection based on self-supervised learning and deep transfer learning** to effectively characterize the target area in the deep latent space. Then the **Deep Change Vector Analysis (DCVA)** has been used to compare points in the feature space and segment the area in change and unchanged parts. While in the real dataset AHN-CD, the self-supervised learning-based method allowed getting more accurate change prediction than traditional distance-based methods, these unsupervised methods still lack of precision in dense urban areas and further only provide binary change information.

Therefore, in a last section, we proposed a **weakly supervised method based on the DeepCluster principle to tackle multi-class change segmentation in raw 3D PCs**. We saw the **importance of the choice of an appropriate architecture** to extract valuable change-related features. Also, **guiding the network using hand-crafted input features** along with 3D points coordinates is advocated. Using these recommended configuration, our proposed method, **DeepCluster 3D Change Detection (DC3DCD)**, allows obtaining **better results than a fully supervised traditional machine learning** algorithm relying on hand-crafted features and to **reach scores of fully supervised deep networks trained on 2.5D rasterization** of PCs (i.e., the only existing models in the literature before this thesis). We further proposed to **improve DC3DCD** by introducing a **contrastive loss** leading to comparative results as fully supervised deep network in an ideal case where the similarity boolean used in the contrastive loss is faultless. However, there are still improvements to be made to find a right manner to obtain the similarity.

While the unsupervised methods designed in this chapter and the supervised ones

introduced in Chapter 3 were assessed on urban contexts, we will consider another application context to illustrate the ability of deep networks to deal with various use cases.

Chapter 5

Applications to geosciences

Contents

This chapter is focused on experimentation in geosciences. Until now, our experiments have been focused on urban environments because of the prime importance of monitoring such environments and the lack of public annotated datasets in other fields of study. We experiment here our supervised networks (*Siamese KPConv* and *Encoder Fusion SiamKPConv*) in two different use cases in geosciences where we collected some annotated data.

First, in collaboration with researchers from the Littoral, Environnement, Géomatique, Télédétection (LETG) lab (UMR 6554) in Brest (France), we built up a dataset with a manual ground truth annotation for erosion detection in steep cliffs. As presented in Section 5.1, the dataset contains several multi-sensor 3D acquisitions over Petit Ailly cliffs in Varengeville-sur-Mer, France[19].

In a second part, we focused on landslide sources and deposits caused by a magnitude M_w 7.8 earthquake in a wide mountainous area in Kaikōura region, New-Zealand. This work, presented in Section 5.2[20], was realized in collaboration with researchers from Observatoire des Sciences de l'Univers de Rennes (OSUR) lab (UMR 6118) in Rennes (France).

5.1 Cliff monitoring

The dynamics of cliff erosion is a complex phenomenon triggered by various factors whose relative contribution is still difficult to estimate. Since cliff erosion is likely to increase with sea level rise due to climate change (Slott et al., 2006; Ashton et al., 2011; Masson-Delmotte et al., 2021), understanding changes in coastal cliffs in order to better manage them would ensure the safety of communities and infrastructure threatened by erosion. In cliff erosion, one can distinguish mass movements and debris falls. According to Varnes (1978), debris falls refer to tiny rocks falling from the cliff face, whereas mass movements refer to rock falls corresponding to larger scale movements of parts or all of the cliff. To quantify cliff retreat rates over decades, the simplest way is to measure the distance from different cliff top locations using aerial photographs or historical maps (Lee and Clark, 2002; Brooks and Spencer, 2010). However, since changes at the cliff top may not be representative of changes over the entire cliff face (Young et al., 2009), comparisons between cliff face 3D PCs are also used. For this second approach, methods such as TLS,

19. This study was presented in the ISPRS Congress 2022 (de Gélis et al., 2022).
20. This study will be presented in IGARSS 2023 (de Gélis et al., 2023a).

ALS, MLS, unmanned aerial vehicle (UAV) photogrammetry, or terrestrial photogrammetry (TP) are used to monitor the cliff face (Young et al., 2010; James and Robson, 2012; Michoud et al., 2014; Letortu et al., 2018). Surveys are taken over time and PCs are compared using tools such as C2C (Girardeau-Montaut et al., 2005) in CloudCompare software (Girardeau-Montaut, 2016). Manual analysis of the differences is then performed to determine areas of erosion on the cliff face and accumulation at the cliff foot. PCs differences provide cliff retreat rates and allow for estimation of eroded volume (Letortu et al., 2015). Different methods trying to automate cliff faces changes extraction and cliff top and toe delineation exist, however most of them do not process 3D data directly but 2.5D rasterization of PCs and finally use DEM differences (Young and Ashford, 2006; Swirad and Young, 2021). For example, in Young and Ashford (2006), the elevation variability is calculated through DEM difference where negative cells represent erosion and positive cells represent accumulation, enabling to highlight significant erosion areas on cliffs with slight slope. In Swirad and Young (2021), inventories of erosion and deposition objects are retrieved with a method consisting in four steps: i) PC processing, ii) cliff face identification, iii) change object inventories, and iv) object classification. Erosion rates are calculated using 2.5D DEM analysis combined with vertical and planimetric detection thresholds, Normalized Difference Vegetation Index (NDVI), machine learning processes and manual quality control. These methods, although relevant, are not applicable on vertical or very steep cliffs. A direct 3D approach is more adapted to study vertical cliffs and to limit the loss of information generated by the data rasterization. Thereby, we propose to try our supervised networks to retrieve erosion and accumulation in cliffs.

5.1.1 Study area

For this study, we focus on a section of cliffs at Varengeville-sur-Mer, located in Normandy, Seine-Maritime (Northwestern France), along the English Channel. Cliffs of Seine-Maritime cover 120 km of coast (from Le Havre to Le Tréport), and are around 60 m to 70 m high. Geologically, they are part of the sedimentary Paris Basin and made of Upper Cretaceous chalk interbedded with flint bands (Pomerol et al., 1987; Costa, 1997; Laignel, 2003; Mortimore and Duperret, 2004). Along the Cap d'Ailly, where Varengeville-sur-Mer is located, cliffs are specific: the residual flint formation over Santonian chalk strata has been replaced by a bed of clay and sandy sediments from the Paleogene age (Bignot, 1962). The Cap d'Ailly is an erosion hotspot where retreat rates can locally exceed 0.80 m/y (Letortu et al., 2014). In Varengeville-sur-Mer, the Petit Ailly

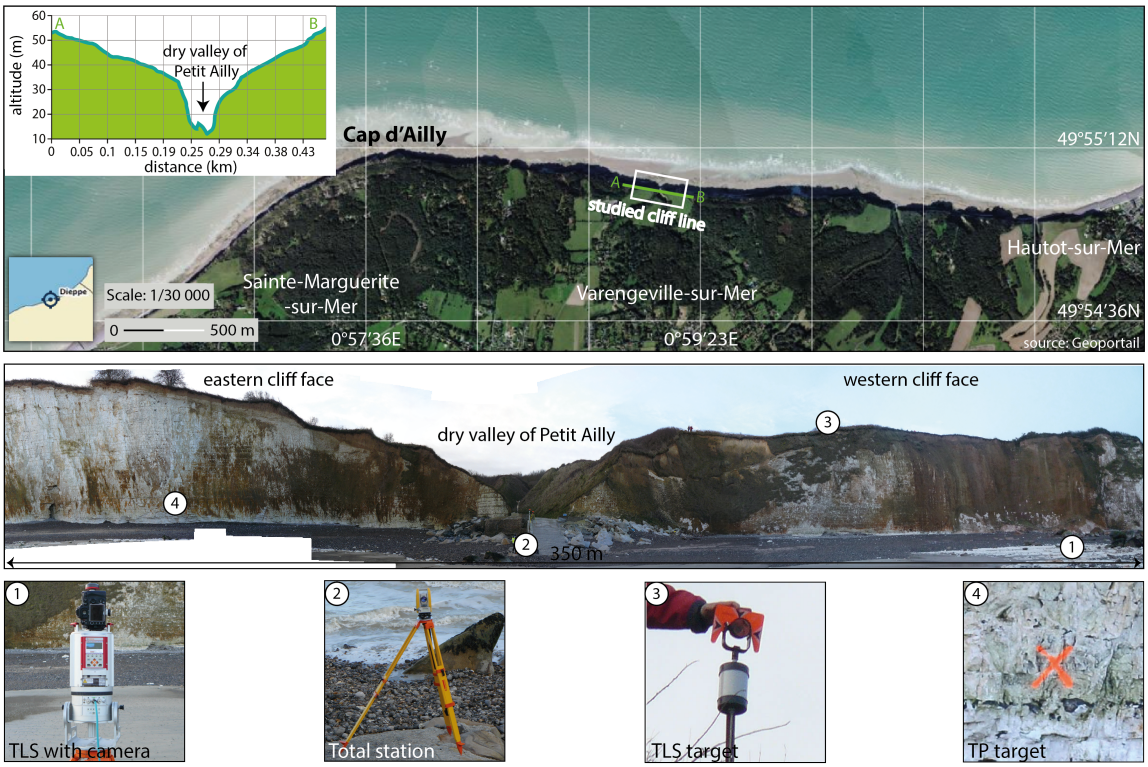


Figure 5.1: **Panorama and aerial photography of Petit Ailly cliff (Varengeville-sur-Mer) and instrumentation used for the TLS survey** (2016-01-28). Source: modified from Letortu et al. (2018).

cliff face (250 m to 350 m long, 40 m high, slope from 70° to overhang) (Letortu et al., 2018; Jaud et al., 2019) (see Figure 1) is monitored from 2010 every 4-5 months (Letortu et al., 2015; Letortu et al., 2019) in the framework of DYNALIT French Observatory Service by terrestrial laser scanning and photogrammetry (Letortu et al., 2018) to improve knowledge on retreat rates and main triggering mechanisms for rock fall activity. This large dataset highlights that rock fall activity is intense over 250 m to 350 m, with a retreat rate of 0.36 m/y and 4 rock falls over 1,000 m^3 (Letortu et al., 2019).

5.1.2 Varengeville-sur-Mer 3D change detection dataset

PCs used in this study are taken from several sources: TP and TLS, with various densities (see Table 5.1). The PC from TP survey contains a significant larger number of points than PCs from TLS surveys, hence TP PC has a larger point density than TLS PCs (9,440 points/m^2 for 2016 TP PC and a maximum of 381 points/m^2 for 2017 TLS PC).

Date	Type	Nb of points (pts)	Mean surface density (points/m^2)
2013-09-25	TLS	2,148,462	202
2016-01-28	TP	123,702,868	9,440
2017-11-02	TLS	4,154,905	381
2018-01-16	TLS	4,038,505	334
2020-04-14	TLS	3,293,548	276

Table 5.1: **Petit Ailly PCs dataset.**

Thereby, the following study is based on multi-sensor acquisitions.

Terrestrial Laser Scanning survey

The TLS used for this study is a Riegl® VZ-400 emitting a laser pulse in the near-infrared (1,550 nm), using the time-of-flight of laser pulse to measure the position of a point. Scan data provided by this instrument have a theoretical accuracy of 0.005 m and a precision of 0.003 m at a range of 100 m. The Riegl® VZ-400 is equipped with a Nikon D800 camera with a fisheye lens, providing photographs that can be used to texturize the 3D point cloud. The georeferencing of the PCs is performed with the Trimble M3 total station to measure several reflective targets used as ground control points (GCPs). The advantage of the use of a total station is that it allows to measure targets closer to the base of the cliff without being affected by mask effects as a Differential Global Positioning System (DGPS) would be. PCs are projected in RGF 93 – Lambert 93 and IGN69, i.e., official planimetric coordinate system and vertical datum in France, respectively. Numerous TLS targets (see Figure 5.1) are used (redundancy of measurements) and set at different distances from the scanner to limit the alignment error (Letortu et al., 2018).

Terrestrial Photogrammetry survey

The terrestrial photogrammetric device used in this study is a Nikon D800 reflex camera configured with a 35 mm focal length. The camera positions should be a short distance apart and at least 20 m from the cliff face. Photographs are acquired from multiple positions at 10° to 20° angular intervals over a wide range of angles to cover the area (James and Robson, 2012). To ensure a quality result, an overlap of at least 60% (80% is recommended for Structure-from-Motion (SfM) photogrammetry) is required between each photograph, and each scene must be taken from various points of view (Letortu et al.,

PC pairs	Type	Class distribution (%)			
		Unchanged	Erosion	Accumulation	'No data to compare'
2013-09-25 – 2016-01-28	TLS – TP	51.77	40.57	1.32	6.33
2016-01-28 – 2017-11-02	TP – TLS	66.07	22.43	0.47	11.02
2017-11-02 – 2018-01-16	TLS – TLS	79.65	16.20	3.19	0.96
2018-01-16 – 2020-04-14	TLS – TLS	69.67	29.92	0.00	0.41

Table 5.2: **Petit Ailly PC pairs.**

2018). In order to georeference the models, GCPs are also needed for TP survey. We use TP targets and measure their absolute coordinates with a total station (see Figure 5.1). To limit doming effect due to radial distortion and erroneous camera model, GCPs should be numerous (James and Robson, 2014; Jaud et al., 2017). Photographs of the cliff and target coordinates are used to derive a georeferenced 3D PC using the Structure-from-Motion Multiview Stereo Photogrammetry (SfM-MVS) algorithm in Agisoft Metashape. Accuracy is not measured directly for TP survey because it requires geodetic references (surveyor's nails) that are not visible in the 2016 PC. Comparison between TLS and TP 2016 PCs reveals a mean error value from 0.013 m to 0.03 m. Further details on the TP survey method used are described in Letortu et al. (2018).

Point clouds annotation

Manual annotation is performed between the PC pairs (see Table 5.2) using C2C tool in order to assign a label to each point of the second PC of the pair. Four classes are defined: 'unchanged', 'erosion', 'accumulation' and 'no data to compare'. The 'unchanged' class designates points whose position is unchanged or almost unchanged (with a tolerance margin of 20 cm) between the two surveys of each PC pair. Points of the most recent PC of the pair are considered as 'erosion' when they appear behind the oldest PC and as 'accumulation' when they are located in front of the oldest PC and rather at the cliff foot or on slight slopes (below 15°) (see Figure 5.2). A 'no data to compare' class is also defined to address the problem of an occlusion (due to different methods of data acquisition) only visible in the oldest PC of the pair of PCs. Indeed, in this case, the C2C distance computed between the two PCs is inconsistent, thus preventing us to evaluate whether it is accumulation, erosion or no change. We recall that the C2C distance has just been used to help the manual annotation task. To annotate the datasets with respect

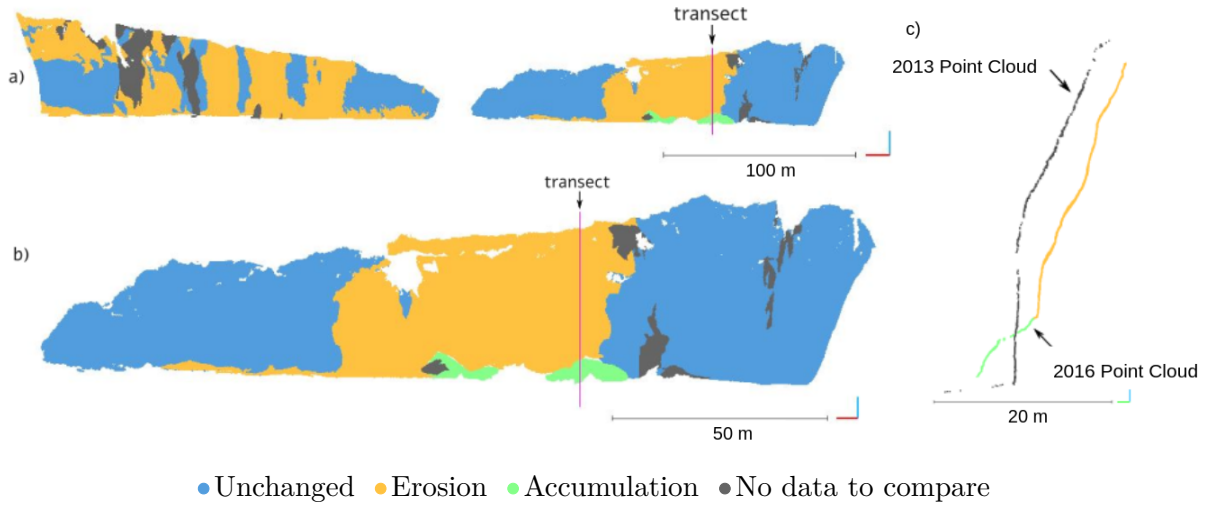


Figure 5.2: **Petit Ailly cliff annotation:** (**a**) annotation of the PC from 2016-01-28 compared to the PC from 2013-09-25; (**b**) western part of Ailly cliff and transect position; (**c**) transect showing erosion of the cliff with accumulation at the cliff base.

to the distance separating the two PCs of each pair, a threshold of 50 cm is used, and refined to 20 cm for some parts. Thereby, in this study, we mainly focus on extracting movements greater than 20 cm to 50 cm, thus smaller movement like debris falls may be missed.

Dataset configuration

In order to conduct our experimentation, we divide our dataset into three parts dedicated to training, validation, and testing. We recall that data are annotated according to the previous acquisition, forming 4 pairs of PCs. As can be seen in Table 5.2, each class is not equally represented in each PCs. As a matter of fact, accumulation class is very rare compared to eroded and even unchanged areas. It is further not present in each PC acquisition. Thus, the division of the dataset is made in such a way that each split is as representative as possible of each class of change. Thereby, the split of pairs of PCs in each training, validation and testing set is made as indicated in Table 5.3. Accumulation class is less represented in point clouds, thus we divided the 2016-2017 and 2017-2018 pairs into eastern and western parts at the dry valley of Petit Ailly (see Figure 5.1) to have examples of accumulation in all of the training, validation, and testing set.

Training set	Validation set	Testing set
2013-2016	2016-2017 East	2017-2018 East
2016-2017 West	2017-2018 West	2018-2020

Table 5.3: **Split of the dataset into training, validation and testing sets.** For each pair of PCs, years relates to the year of acquisition given in Table 5.1. The annotation is always given for the second PC of the pair with regard to the first PC.

5.1.3 Experimental results

5.1.3.1 Experimental settings

In the following, both *Siamese KPConv* (Chapter 3, Section 3.2.1) and *Encoder Fusion SiamKPConv* (Chapter 3, Section 3.4.2) are assessed. To setup experiments, we have to choose the initial sub-sampling rate (dl_0) of PCs. Thus, no matter the type of input PC (TLS or TP), the input PC's resolution at the first layer is always the same. Notice that final results are interpolated back to initial PC resolution given in Table 5.1. To our opinion, dl_0 should be chosen as small as possible to stick with PC's initial density, while in the same time fitting with available memory in the GPU. Finally, this first sub-sampling rate is directly linked with the size of input given to the network. Indeed, as for satellite images divided into patches, PCs are also subdivided to feed the network. Conversely to urban change detection experiments of Chapter 3, we choose here to divide PCs into spheres and not vertical cylinders. Indeed, in urban areas, the motivation of considering cylinders instead of spheres is to be sure to always include ground in vertical cylinders (see Chapter 3). The study case here is quite different, and spheres appear more suited than cylinders because they contain fewer points, allowing to choose a larger radius (we recall that considering too many points as input for the network leads to memory capacity issues). Spheres are centered on a point of the second date PC, thus the radius should not be chosen too small to ensure to give also points of the first PC in case of large changes as well as providing enough context. In Thomas et al. (2019), the authors chose the radius of input sphere of 50 times dl_0. However, according to our own experiments, the best results were obtained with spheres of 10 m in radius and dl_0 set to 0.15 m.

As far as other network hyper-parameters are concerned, we use the same configuration as in Chapter 3: a SGD with momentum to minimize a point-wise negative log-likelihood loss, with a batch size of 10, a momentum of 0.98 and an initial learning rate of 10^{-2}. The learning rate is scheduled to decrease exponentially. A probability dropout of 0.5 in

the last classification layers is set. Also, in order to prevent from overfitting, we set a L_2 loss regularization with a factor of 10^{-6}. As the dataset is largely imbalanced, the loss is weighted according to training set class distribution. For the training, as for the urban context, 3,000 samples are used at each training epoch. Data augmentation (addition of a random Gaussian noise, and input spheres rotation the around vertical axis) are also used. Conversely to urban application, the random rotation around the vertical axis is fixed to randomly vary between $-15°$ and $15°$, in order to prevent from situations where a PC initially in front of the other to be found behind, thus disturbing the erosion detection.

In the following experiment, we decided to normalize input spheres along X and Y axis by retrieving the minimum value into the sphere of X and Y axis respectively. As for the vertical Z axis, we do not perform any normalization regarding the minimum value in the whole cliff, in order to keep information related to elevation. This may help for change classification, in particular for accumulation class, since it is mainly located at the cliff foot.

For presented experiments, the training time was about 20 hours on a single GPU (Titan RTX), while the prediction for the whole testing set (see Table 5.3) takes about 8 minutes.

We provide a comparison with a distance-based method, M3C2, giving a mean surface change along a normal direction (Lague et al., 2013). Based on this distance, we apply an empirical thresholding at $-0.2\,\mathrm{m}$ and $0.2\,\mathrm{m}$ to extract accumulation, unchanged and erosion areas.

5.1.3.2 Qualitative and quantitative results

Quantitative results are shown in Table 5.4. As can be seen, *Encoder Fusion SiamKPConv* architecture brings better results than *Siamese KPConv* on this dataset. In particular, per class IoU indicates that unchanged areas and erosion are well classified by *Encoder Fusion SiamKPConv*. This is confirmed by qualitative results presented in Figures 5.3, 5.4 and 5.5 corresponding to each part of the testing dataset. Main differences with the ground truth appear at boundaries of erosion parts and in some more intricate areas, as for instance in the top middle left side of the Figure 5.4 where erosion, unchanged and 'no data to compare' classes are almost mixed up and cliff structure is more complex. Concerning the 'no data to compare' class, main parts classified as such in the ground truth are well classified by *Encoder Fusion SiamKPConv*. However, more areas identified as 'no data to compare' by the network are present. To explain this, let us remark that

	Per class IoU(%)			
	Unchanged	Erosion	Acc.	No data to compare
Siamese KPConv[21]	83.39	61.06	21.88	26.21
Encoder Fusion SiamKPConv	93.84	**87.78**	26.68	**55.22**
M3C2 + threshold	**95.06**	87.23	**48.20**	41.58

Table 5.4: **Quantitative results for *Siamese KPConv*, *Encoder Fusion SiamKPConv* and M3C2 methods for cliff erosion detection.** Acc. stands for accumulation.

in some parts, only a few points allow the annotator to label with high precision whether there is or not a change, whereas in some other parts, the identification is trickier, or even impossible, leading to an annotation as 'no data to compare'. Thereby, ground truth and prediction should not be strictly compared.

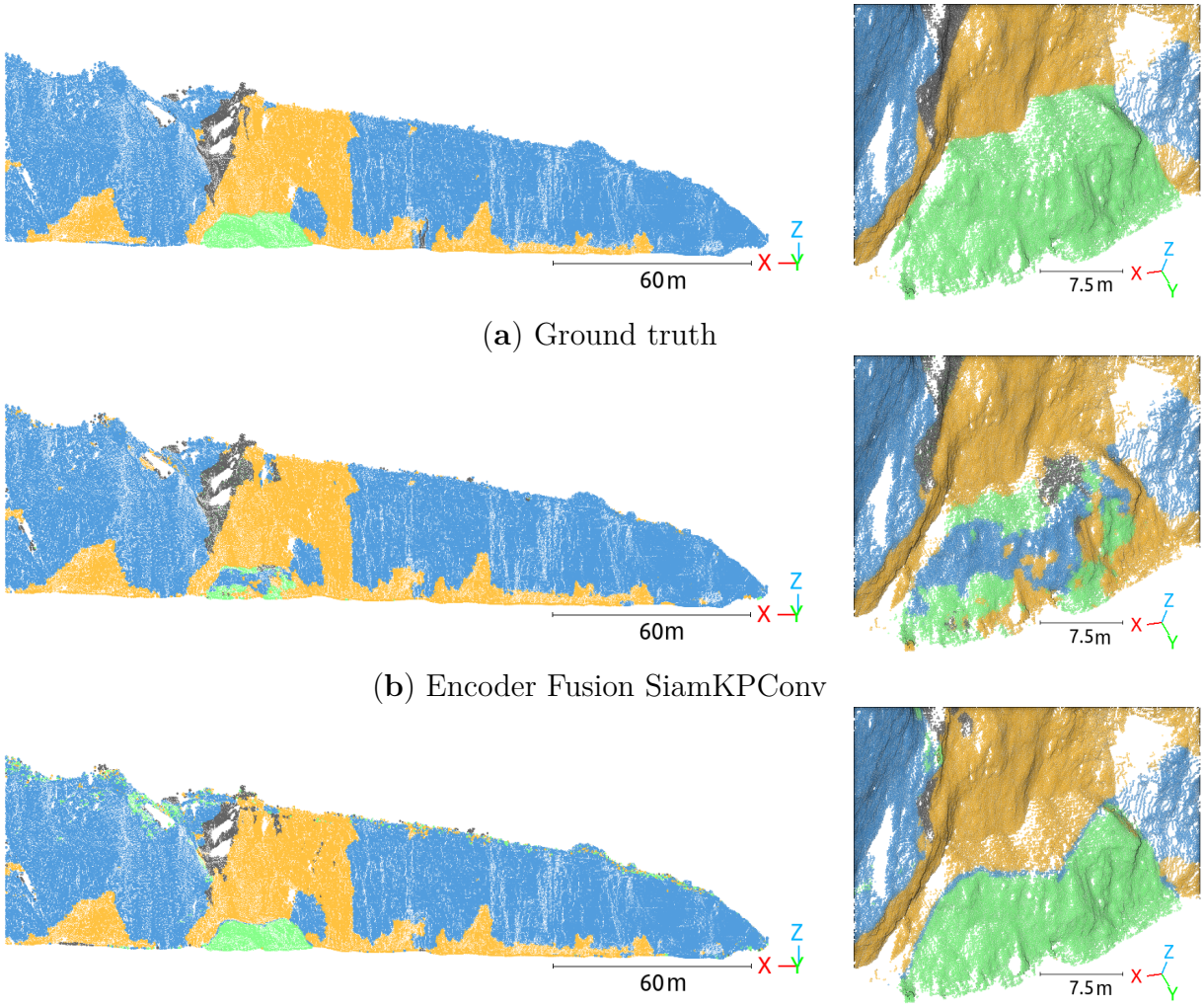
A huge difference with the ground truth is visible in Figure 5.3b for the accumulation class where our methods seem unclear. As related by quantitative results, accumulation class obtains a considerably lower score in comparison to erosion. In this fully supervised context, these results are surely explained by the only few accumulation examples available in the whole dataset. Indeed, the training set contains only 1.5% of points for accumulation, whereas the erosion represents 31.1% of points. It is worth noting that testing set follows the same trend (see Figure 5.3(a), 5.4(a) and 5.5(a)).

5.1.4 Discussion

Comparison between *Siamese KPConv* and *Encoder Fusion SiamKPConv* architectures

As seen in the previous section, *Siamese KPConv* provides lower results than *Encoder Fusion SiamKPConv*. The same trend was obtained in urban datasets (see Chapter 3 Section 3.4), however the gap between the two methods is higher in Petit Ailly cliff dataset than for the urban context. To explain this, let us remark that in cliff erosion detection, the semantic of sensed objects (i.e., cliff) remains the same through time. Changes to be identified in this task are indeed rather related to the studied object in itself: the cliff semantic is the same, but its shape is modified through time. This might explain the great gap between results of the two architectures. Indeed, *Siamese KPConv* detects changes only through mono-date encoder feature difference while *Encoder Fusion SiamKPConv*

21. Results presented in this chapter are different from those included in the paper presented at ISPRS Congress 2022 (de Gélis et al., 2022) due to some implementation changes.

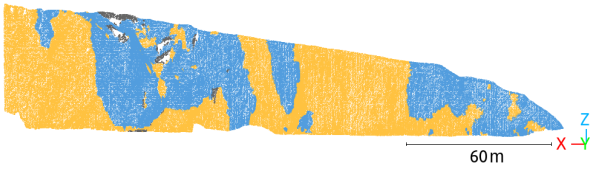


(**a**) Ground truth

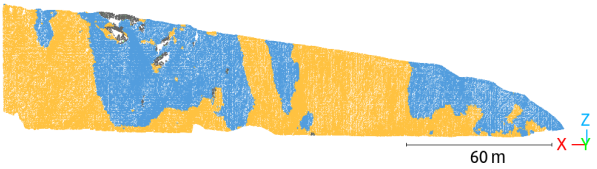
(**b**) Encoder Fusion SiamKPConv

(**c**) M3C2

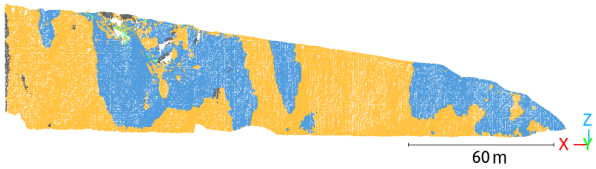
Figure 5.3: **Results on the eastern part of the cliffs between 2017 and 2018 acquisitions:** (**a**) ground truth; (**b**) *Encoder Fusion SiamKPConv* results; (**c**) M3C2 results. A zoom over the accumulation is also given.



(**a**) Ground truth



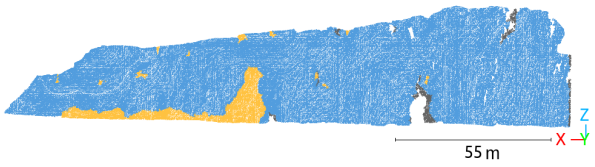
(**b**) Encoder Fusion SiamKPConv



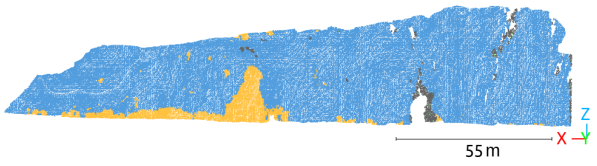
(**c**) M3C2

● Unchanged ● Erosion ● Accumulation ● No data to compare

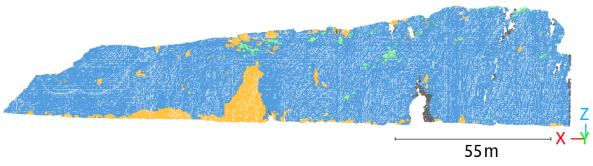
Figure 5.4: **Results on the eastern part of the cliffs between 2018 and 2020 acquisitions:** (**a**) ground truth; (**b**) *Encoder Fusion SiamKPConv* results; (**c**) M3C2 results.



(**a**) Ground truth



(**b**) Encoder Fusion SiamKPConv



(**c**) M3C2

● Unchanged ● Erosion ● Accumulation ● No data to compare

Figure 5.5: **Results on the western part of the cliffs between 2018 and 2020 acquisitions:** (**a**) ground truth; (**b**) *Encoder Fusion SiamKPConv* results; (**c**) M3C2 results

.

architecture has a specific part of the network to encode feature difference at multiple scales thanks to convolution. In the Petit Ailly cliff dataset, each point of the two PCs has the same mono-date semantic, surely leading to quite comparable mono-date features extracted by the encoder of *Siamese KPConv*, explaining the rather low results of this method.

Comparison with M3C2 results

In Table 5.4, a comparison with the distance-based method M3C2 is provided. To distinguish between different types of changes, a threshold is applied on this distance. According to Table 5.4, M3C2 obtains better results on the accumulation class. However, as seen in the results' description part, our methods obtained lower results mainly because of the lack of representativity of the accumulation class in this dataset. It is worth noting that M3C2 method is also far from being perfect on this class. Indeed, M3C2 results are only based on a measure of distance between two PCs. Thereby, when the accumulation appears 'behind' the first PC, i.e., when the stock of debris accumulated at the foot of the cliff decreases from one survey to the other, accumulation will not be highlighted by M3C2 method and will be even mixed with erosion (negative distance). Note that the testing set does not contain exactly such examples, but it is likely to happened in reality. Still, in Figure 5.3c, this is visible when comparing M3C2 results with the ground truth where only the half of the accumulation is well identified. Furthermore, as visible in Figures 5.3c and 5.5c, some little accumulations are identified on the cliff by M3C2, conversely to the ground truth probably corresponding to vegetation growth. Using only a distance-based method will never allow to automatically distinguish between different positive (or negative) changes. Even though our methods do not bring optimal results here, explained by the extremely low percentage of accumulation class in the dataset, they also give an interpretation of changes by giving directly a categorization. For example, for accumulation class, the morphology, and position with regard to the cliffs should be taken into account and our architectures automatically extract features based on points neighborhood at different scales (with Kernel Point convolution operations). Finally, our methods give a categorization of changes based on surroundings of points. For a generalization on various types of cliffs, other classes of change (e.g., on vegetation) could be added using *Encoder Fusion SiamKPConv* method, which would not be possible with a threshold applied on a distance-based method such as M3C2.

Class representativity

Encoder Fusion SiamKPConv can be applied on cliffs with various slope degrees with satisfactory results, but it should be noted that the results are dependent to the annotation quality. Notice that the same protocol for the cleaning of 3D PCs and annotation should be applied for each pair of PCs. Despite encouraging results obtained by this method, supervised deep learning still requires a dataset both large and representative of each class during the training phase. Thus, the accumulation class that is underrepresented might obtain better results if our dataset contains more examples. The same remark stands for the 'no data to compare' class, which is also (as mentioned in Section 5.1.3) more subjective. Conversely to urban context, cliff variability is important (especially in cavity) so information from missing data at the first PC of the pair cannot be interpolated as it is done for hidden building facade in urban environment in previous chapters. Hence, we need to create this class in areas where no conclusions on cliff dynamics can be drawn.

Artifact robustness

It is worth noting that input data should be registered together. However after the experimentation, we noticed a Z offset error on the eastern part of the 2013-2016 pair, probably due to a target movement during the survey or the depression of the TLS during acquisition on a wet sandy foreshore, leading to mislabeling of erosion parts. Although this pair is in the training set, we saw that erosion identification is satisfactory, showing robustness of the method to acquisition artifacts and mislabeled data during the training.

Because of their verticality, Normandy chalk cliffs and more precisely Varengeville-sur-Mer cliffs have a very scarce vegetation cover, limiting the training on vegetation growth and retreat. Hence, these classes are not highlighted in the ground truth (corresponding points are labeled as unchanged), and subsequently ignored by the deep networks. Nevertheless, these classes should be considered if applying the method on vegetalized cliffs.

Training configurations

In order to select the best training configuration, several settings of input data have been tested. As explained in Section 5.1.3.1, input geometry (spherical or cylindrical) as well as first sub-sampling rate dl_0 have an influence on results. Indeed, considering a too large sphere radius implies a higher sub-sampling rate, thus avoiding change extraction.

Similarly, choosing a too thin sub-sampling rate requires to diminish input sphere radius, implying that not enough context can be taken into account for feature extraction. Furthermore, we noticed that with spheres of a fixed 10 m radius, diminishing the sub-sampling rate finally does not improve results: this may be explained by the fact that setting dl_0 to 0.15 m is enough accurate compared to ground truth precision. Thereby, the sub-sampling rate and the size of input sphere should be chosen according to the level of details of the ground truth.

We have reported results with spherical sub-PCs, conversely to the vertical cylindrical inputs used with urban data of Chapter 3 that were motivated by the verticality of changes and the need for including ground in PCs. In this cliff context, changes are more likely to happen on horizontal axis, so we also conducted an experiment by taking some cylindrical inputs oriented according to north-south axis as the cliff is oriented along eastern-western axis (see Figure 5.1). Results were quite similar to those obtained with spherical inputs. Another idea would have been to select cylinders oriented according to the normal of the cliff face to be more precise in the orientation of cylinders.

Erosion detection scale

The method developed here presents an interest for the long-term monitoring of cliffs since it allows the detection of mass movements of rocks. In order to detect debris falls, more time should be spent to annotate data very accurately and adjustment of the first sub-sampling rate (dl_0) is required. This rate should be chosen equal or thinner than debris size, worth noting that the minimum threshold for detecting erosion and accumulation would be the ground sampling distance (GSD) of the sparser PC of the initial dataset. Thus, initial clouds resolution should be homogeneous and adequate regarding the size of the minimal debris falls to detect. Indeed, without homogeneity between the resolutions of the different datasets, detection artifacts could appear.

We have shown the relevance of our networks to deal not only with urban data but also with a first application in geosciences. In the next section, we will cope with landslide dynamics in a use case that require dealing with vegetation as well.

5.2 Post-earthquake landslide detection

Extreme events such as earthquakes and storms often cause thousands of landslides leading to hazard for local populations (Pollock and Wartman, 2020), and cause severe hillslope

and landscape erosion (Malamud et al., 2004). Landslide analysis is essential to better understand the mechanics of landslides, hillslope erosion (Marc et al., 2019), and to predict hydro-sedimentary hazards such as alteration of river dynamics due to landslide sediment deposit (Croissant et al., 2017). To this end, the creation of efficient and exhaustive landslide maps is essential. In mountainous areas, topographic changes are difficult to localize because of the access difficulty and the large size of studied areas. Thereby, landslide mapping is performed through 2D satellite or aerial images (Behling et al., 2014; Marc et al., 2019) or 3D point clouds acquired via aerial LiDAR survey or photogrammetry (Ventura et al., 2011; Bernard et al., 2021). In particular, elevation data ease landslide volume computation and change detection (Okyay et al., 2019) while 2D images may lead to omitted topographic changes in vegetated areas (Bernard et al., 2021). For this reason, 3D data appear again to be an interesting solution. Yet, the majority of studies rely on 2.5D rasterization of 3D data into DEMs of each epoch (Okyay et al., 2019), and then subsequently using a DSMd, also called Difference of DEM (DoD) in the geosciences community (Bernard et al., 2021; Okyay et al., 2019). However, on top of information loss due to the rasterization process (cell size and interpolation), DoD leads to strong errors in very steep surfaces (Bernard et al., 2021). To overcome these issues, we propose to apply our supervised architectures to process directly the raw 3D PCs to detect landslide sources and deposits.

5.2.1 Study area and dataset

We focus on the Kaikōura region in New-Zealand where severe topographic modifications have been sensed following the magnitude M_w 7.8 earthquake of 14 November 2016 (see Figure 5.6a). Following this extreme event, around 30,000 landslides over a 10,000 km^2 were detected (Massey et al., 2020). Two aerial LiDAR acquisitions were realized with 2 years and 8 months difference before and after the earthquake. Acquisitions have an average ground point density of 3.8 ± 2.1 and 11.5 ± 6.8 points/m^2 respectively.

Bernard et al. (2021) proposed a semi-automatic workflow called 3D point cloud differencing (3D-PcD) for the detection of landslide sources and deposits from multi-temporal airborne LiDAR data. This method is based on the M3C2 algorithm (Lague et al., 2013) and on a two-step approach to filter out false detections that can emerge due to vegetation classification errors and geometrical inaccuracies: i) using significant topographic changes (provided by M3C2 algorithm) and ii) exploiting a patch-based metric to detect remaining false detections. Thereby, the ground truth has been obtained by a combina-

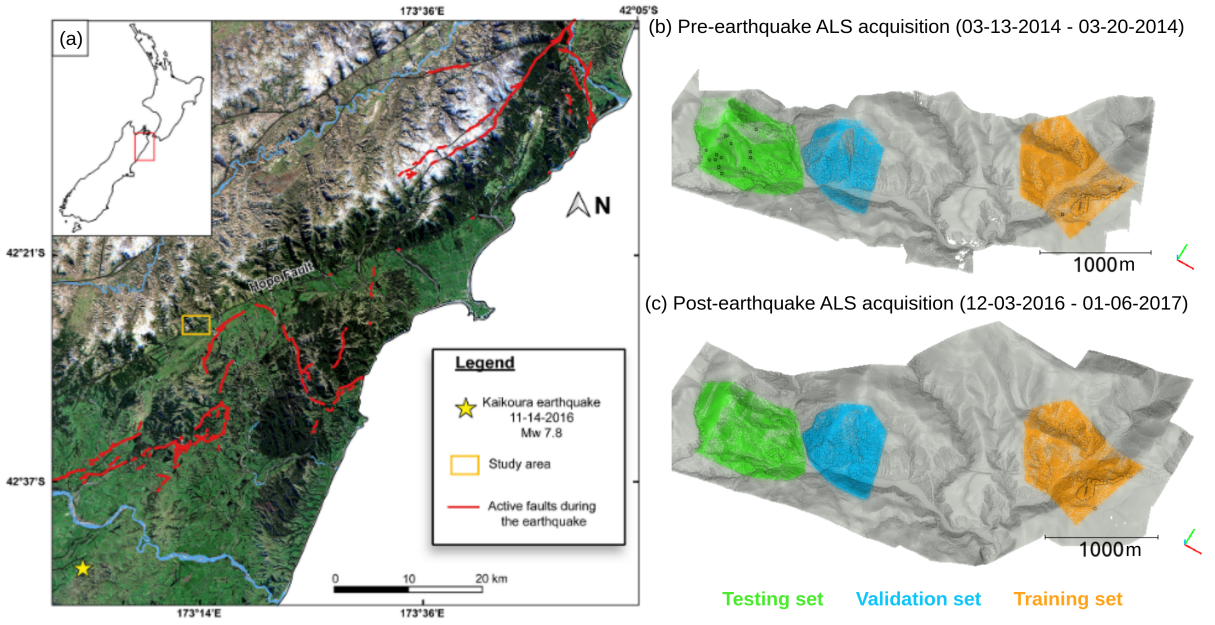


Figure 5.6: **Kaikōura study area:** (**a**) regional map of the regional context and location of the study area near Kaikōura (Source: Bernard et al. (2021)); (**b-c**) pre- and post-earthquake ALS acquisitions colorized as function of the split for our deep learning experiments.

tion of 3D-PcD (Bernard et al., 2021), vegetation filtering (to prevent from the detection of inaccurate changes), and manual visual analysis for label refinement.

Taking the $5\,\mathrm{km}^2$ area where landslide sources and deposits were retrieved in Bernard et al. (2021), we selected three non-overlapping areas to constitute our training, validation and testing sets. These three splits are depicted in Figure 5.6. Unlike Bernard et al. (2021), our network is fed with raw 3D point clouds without any filtering of vegetation. It is trained to highlight sources and deposit areas in a single prediction step.

5.2.2 Experimental results

5.2.2.1 Experimental settings

Siamese KPConv and *Encoder Fusion SiamKPConv* architectures, presented in Chapter 3, are again experimented.

In the following experiments, only the 3D coordinates of points are given as input to the network. In this context, interesting changes are more of vertical nature, thus as

for urban change detection, and unlike steep cliff application, we use vertical cylinders to segment the dataset. In particular, cylinders of a radius of 30 m are used with a first sampling rate dl_0 at 0.8 m. Concerning other hyper-parameters, we use similar values as those considered for urban settings. In particular, 3,000 samples are used at each training epoch. They are selected using the same random drawing according to the class representativity as in urban context. The same data augmentation process is used, i.e., cylinder rotation and addition of a Gaussian noise.

5.2.2.2 Results and discussion

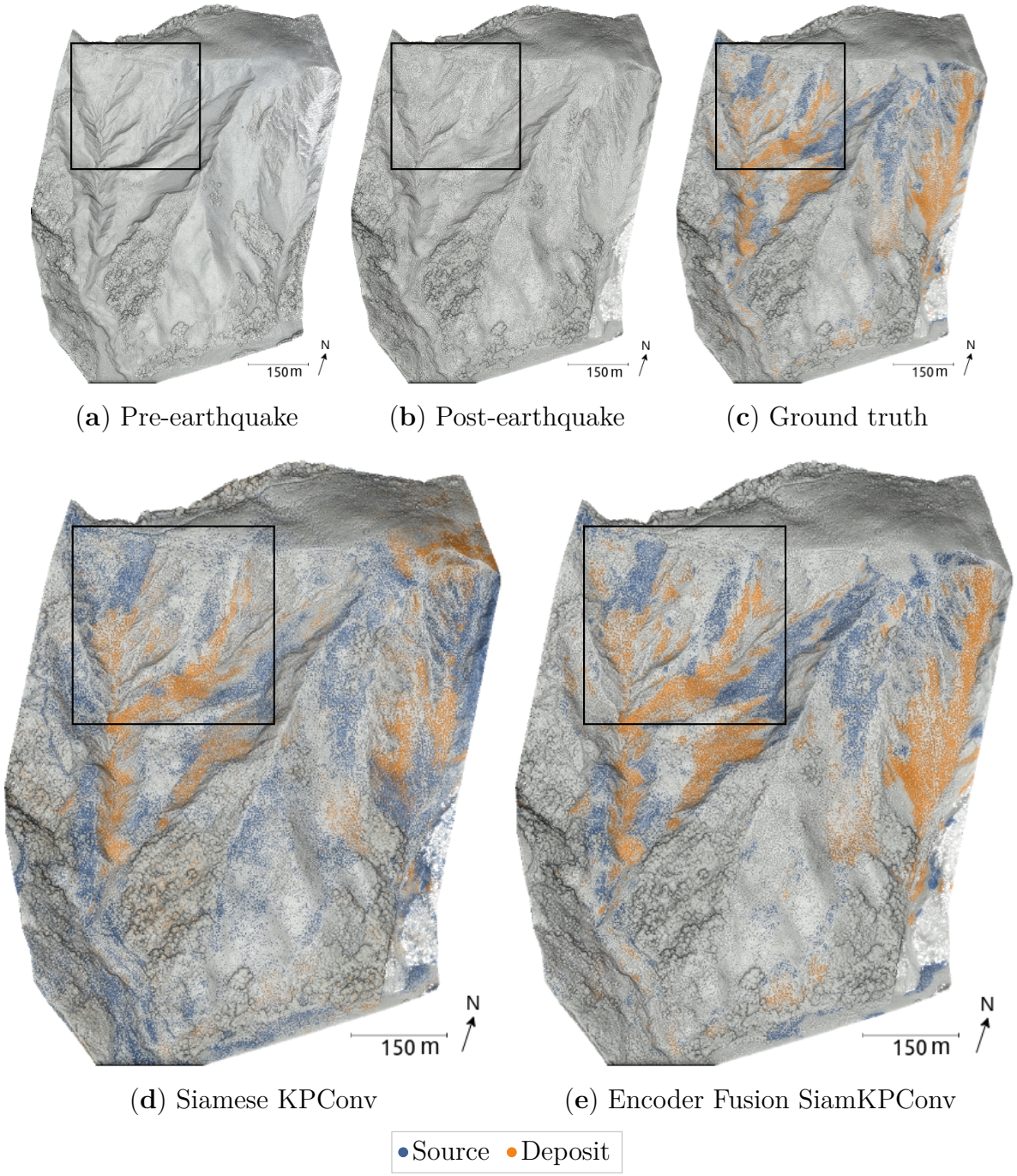
Quantitative and qualitative results are presented in Table 5.5, Figures 5.7 and 5.8. Similarly to the Petit Ailly cliffs, *Siamese KPConv* fails in recovering landslide sources and deposits. Though only the vegetation does not appear to be identified as changes, it seems that almost all ground surfaces are highlighted as sources or deposits (see for example the zoom in Figure 5.8b). In addition, *Siamese KPConv* widely over detects both sources and deposits. These unsatisfactory results are surely explained by the same reason as the one given for the Petit Ailly cliff dataset concerning the changes not in the semantic but rather in the shape of objects (see Section 5.1.4). This is even emphasized by the fact that *Siamese KPConv* manages to distinguish between ground and vegetation thanks to its mono-date encoders. Conversely, *Encoder Fusion SiamKPConv* seems to perform well on this dataset (with a significant improvement over *Siamese KPConv*), as indicated both by the quantitative results in Table 5.5 and the visual comparison with ground truth in Figures 5.7 and 5.8.

Conversely, *Encoder Fusion SiamKPConv* predictions are very similar to the ground truth. In particular, the main landslide sources and deposits have been correctly retrieved even under the vegetation cover. Principal differences between the ground truth and the prediction appear in smaller changed areas. Some overdetection of sources are visible, as for example in the river bed at the bottom of Figure 5.7e. This is not so surprising since there are indeed some modifications in the river bed, despite these modifications being not interesting for landslide source and deposit identification task. River bed in the training and validation set represents only a small proportion of the data, surely explaining why the network has not learned that changes in river bed are out-of-interest, as it does for changes related to vegetation. This highlights the added value of deep learning methods over distance-based methods that cannot be used as is to identify landslides and other ground changes in vegetated areas. Indeed, a first step of vegetation removal is required,

Method	mAcc (%)	mIoU (%)	Per class IoU (%)		
			Unchange	Source	Deposit
Siamese KPConv	60.59	36.88	59.83	18.03	32.78
Encoder Fusion SiamKPConv	**93.87**	**83.84**	**93.58**	**74.38**	**83.57**

Table 5.5: **Results on Kaikōura landslide detection dataset** given in %.

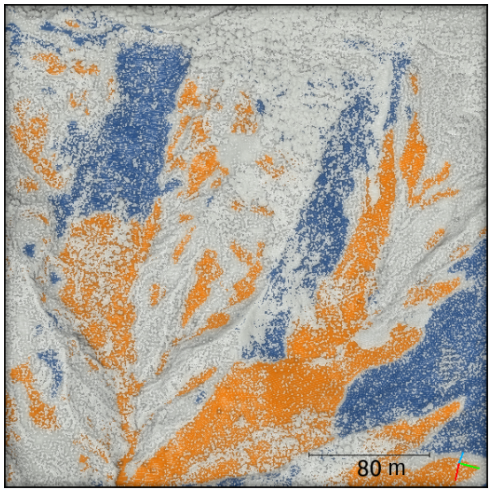
as done for the ground truth annotation of Kaikōura dataset. However, errors is the vegetation identification impact the change detection step as outlined in Bernard et al. (2021). Using *Encoder Fusion SiamKPConv* allows to directly use raw 3D PCs without any task-specific pre-processing step.



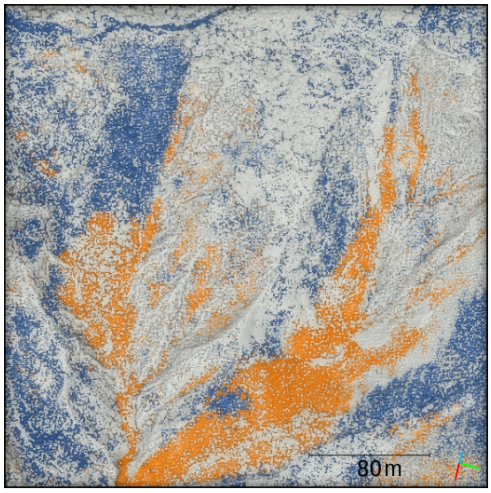
(**a**) Pre-earthquake (**b**) Post-earthquake (**c**) Ground truth

(**d**) Siamese KPConv (**e**) Encoder Fusion SiamKPConv

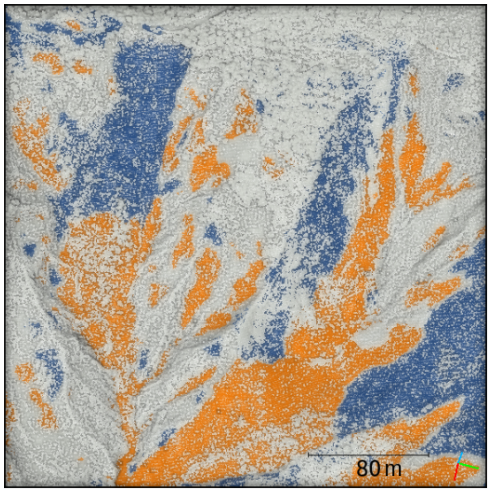
Figure 5.7: **Qualitative results of landslide source and deposit identification in Kaikōura testing set.** Pre- (**a**) and post-earthquake (**b**) input point clouds are shown along with the ground truth in (**c**). The landslide source and deposit predictions are visible for *Siamese KPConv* (**d**) and *Encoder Fusion SiamKPConv* (**e**) networks. Zooms into rectangles are available in Figure 5.8.



(**a**) Zoom in the ground truth



(**b**) Zoom in the prediction of Siamese KPConv



(**c**) Zoom in the prediction of Encoder Fusion SKPConv

Figure 5.8: **Zoom in qualitative results of landslide source and deposit identification in Kaikōura dataset.** The ground truth is provided in (**a**). The landslide source and deposit predictions are visible for *Siamese KPConv* (**b**) and *Encoder Fusion SiamKPConv* (**c**) networks.

5.3 Conclusion

In this chapter, we **experimented our supervised architectures** (*Siamese KPConv* and *Encoder Fusion SiamKPConv*) in two applications related to **geosciences**, namely **cliff erosion detection**, and **post-earthquake landslide source and deposit identification**. Through these two experiments over the multi-sensor 3D acquisitions over Petit Ailly cliffs (Varengeville-sur-Mer, France) and the aerial LiDAR surveys over the mountainous region of Kaikōura (New-Zealand), we highlighted that ***Encoder Fusion SiamKPConv* is not specific to urban environment** as long as the dataset is sufficiently representative of the classes of interest. Once again, these assessments outlined the primacy of *Encoder Fusion SiamKPConv* over *Siamese KPConv*, emphasizing previous conclusions over **the necessity of encoding changes via convolutions over the mono-date feature difference**.

CONCLUSIONS AND PERSPECTIVES

Conclusions of the thesis

In this thesis, we proposed several methods to **detect multiple classes of changes from raw 3D PCs**. The objective is to associate with each point of an actual PC a label related to the change with regard to a past PC. In particular, we investigated how to tackle this task using deep neural networks and how to handle particularly complex raw 3D PC data.

To our knowledge, this is the first study that considers deep learning-based methods for tackling 3D PC change detection. We showed the possibility of using such methods to obtain valuable change segmentation into diverse contexts, varying from urban applications to geosciences. While the main criticism of using deep learning-based methods relies on the large number of required annotated data to train associated networks, we further investigated how to lower the need for such annotated data.

The main contributions of this thesis are summarized below:

- **Public annotated datasets for 3D PCs change detection**: Given the **lack of public dataset** with bi-temporal PCs and change-related annotation, we first proposed in Chapter 1 a **semi-automatic process** to obtain **AHN-CD, a bi-temporal PC dataset containing change annotation**. Although the semi-automatic process allows to rapidly and easily obtain per-point change-related labels from a dataset with mono-date semantic annotation, AHN-CD dataset still contains a lot of ground truth errors due to the difficulty of automatically comparing two PCs. Therefore, we further developed an original **simulator of multi-temporal urban models** and propose to **reproduce an airborne LiDAR surveying campaign** over these simulated models to finally obtain **multi-temporal 3D PCs**. As changes in urban models are introduced by the simulator, acquired PCs are directly **annotated according to the change** and also the semantic of the scene objects at each timestamp. The simulator lets the possibility to configure the LiDAR acquisition as a function of density and noise level. Thereby, we deliver **Urb3DCD**[22], a

22. The dataset is publicly available at the following link: `https://ieee-dataport.org/open-acces`

simulated dataset containing different sub-datasets to mimic various sensor quality and type in order to experiment methods in diverse conditions.

- **Supervised deep networks for raw 3D PCs change detection**: After benchmarking state-of-the-art methods for 3D PCs change detection in Chapter 2, we came to the conclusion that existing methods are still limited, especially for understanding the scene at a global scale. Therefore, building upon the success of deep learning in Earth observation, we proposed in Chapter 3 ***Siamese KPConv***, a deep network relying on **3D kernel point convolutions** (Thomas et al., 2019) and well-known **Siamese architecture** to tackle change segmentation directly from raw PCs. After assessing this method on various datasets under different conditions, we showed that our proposed method was obtaining far more precise results than the state-of-the-art methods, including in more difficult areas containing occlusions. By exploring **variants of *Siamese KPConv***, we emphasized the **sake of extracting change-related features** through **convolutions** applied on the **mono-date feature difference**. In Chapter 5, we also showed the effectiveness of these methods in the field of geosciences for cliff erosion detection and post-earthquake landslide source and deposit identification.

- **Reducing the need of annotation**: Considering the difficulties of obtaining accurate ground truths and training sets, we explored in Chapter 4 different methods relying on **transfer learning**, **self-supervised learning**, **deep change vector analysis** and **deep clustering** to **lower the need of annotation**. We experimented these methods on both real and simulated datasets, and demonstrated some promising prospects for weakly supervised raw 3D PCs change segmentation outperforming traditional unsupervised and fully supervised machine learning methods. However, we also highlighted some difficulties inherent to PCs data (e.g., point-to-point comparison), and to the largely unconstrained problem of unsupervised learning.

In addition to these main contributions, we would like to point out few take-home messages. First, similarly to observations in other remote sensing tasks (e.g., 2D satellite image understanding), **deep learning** is able to provide very **accurate results for multi-class change segmentation into raw 3D PCs in a supervised setting**.

s/urb3dcd-urban-point-clouds-simulated-dataset-3d-change-detection.

Then, both supervised and unsupervised experiments showed the importance of applying **convolutions on multi-scale features differences** in order to obtain better change detection task-related features. Lastly, we demonstrated promising possibilities in reducing the amount of necessary annotated data. However, as **unsupervised setting** is a largely **unconstrained problem**, it is important to **steer the network** toward the change detection task by a careful **choice of the architecture** of the deep network, and by **guiding the network through the use of well-designed hand-crafted features**.

Perspectives

3D point clouds are becoming more and more widespread. For example, we can cite the example of France, where the entire territory will be covered with a high density (HD) aerial LiDAR ($\sim$10 points/m^2) by the end of 2025. In addition, satellite missions providing 3D data are multiplying (e.g., Pléiades, Pléiades Néo, CO3D), allowing access to more and more 3D data such as multi-temporal point clouds. This calls for methods able to process these particular data and especially under the angle of change detection. While in this thesis, we opened the field of deep learning for 3D PCs change detection and demonstrated the possibilities offered by these deep methods, it appeals for more development. By looking at the rapid evolution of deep learning methods, for example in the field of computer vision, we hope multi-temporal PCs understanding can benefit from it. In the following, we outline some future perspectives at both short and long terms.

Short-term perspectives

- **Supervised techniques**: Concerning supervised methods, even though we already obtained some probing results, some recent advances can be applied in our context. For example, one could imagine introducing **attention mechanism** for multi-scale fusion of both change-related and mono-date features, as was already successfully experimented in recent studies in 2D image change detection (Fang et al., 2021; Chen et al., 2022a; Yin et al., 2023). Investigating transformers (Guo et al., 2021) is for sure an interesting perspective for the improvement of our method. However, the scaling to large remote sensing PCs is often not immediate. Note that if it is successful in a supervised setting, these improvements can also be directly introduced in the DC3DCD weakly supervised method by updating the back-bone.

- **Unsupervised techniques**: In the Chapter 4, we proposed a method (DC3DCD) to tackle multiple change segmentation into 3D PCs in a weakly supervised setting. While the training of DC3DCD is fully unsupervised, the user is required to map pseudo-clusters onto real classes. One perspective would be to combine this user-guided mapping with **interactive deep learning**. Even if we saw that a majority of pseudo-clusters is almost pure (i.e., contains only one real class) (see Figure 4.11), some pseudo-clusters still contain several real classes. An idea would be to let the user indicate if a cluster needs to be divided or to be fixed as is. Thereby, i) a large amount of points will consequently be annotated according to real classes corresponding to all points contained in fixed clusters; ii) using interactive learning idea (Kontogianni et al., 2020; Lenczner et al., 2022), these annotated points can be further used as a sparse ground truth to **fine-tune the whole network** in a few iterations; iii) for all fixed clusters in the 'unchanged' class, the contrastive loss proposed at the end of Chapter 4 can be used for these points where a trustable ground truth has been established; iv) we can imagine forcing the clustering algorithm to divide clusters indicated as not pure (using for example ideas of hierarchical divisive methods). Also, once the whole model is trained using DC3DCD strategy to extract discriminative change-related features, it can be interesting to **train an MLP on top of the frozen network** to adapt more to user recommendation (namely, grouping or division of clusters) to iteratively and rapidly converge (just the MLP parameters need to be updated) to a satisfactory change solution.

- **Problems of PCs comparison**: Concerning DC3DCD method, we also presented some preliminary experiments showing that adding a **contrastive** term in the **loss** could improve results up to fully supervised methods if we have a faultless partition of points into binary changes. We already tried several methods (e.g., through multi-task learning) to obtain this binary partition of points, but we obtained mitigated results due to the complexity of directly comparing two PCs. Indeed, a direct point-to-point comparison is impossible unless a strategy is used to match points from the two dates. While in this thesis we basically used the nearest point strategy, some recent works (Courty et al., 2016; Fiorucci et al., 2023) proposed to use **optimal transport** strategies to match points of the two dates. Indeed, as a PC can be modelled in an empirical discrete probability distribution, optimal transport can be used to match these two distributions of the two PCs.

Long-term perspectives

- In an urban context, we proposed a simulator to generate annotated 3D PCs. In Chapter 4, we showed first experiments on how to take the advantage of such simulated data to pre-train a deep network, and thus, drastically diminish the use of annotated real data without any specific fine-tuning strategies. In a more general way, the process that consists of using simulations to pretrain networks is appealing, in particular to limit the number of annotated data. To go further, we can exploit **knowledge distillation** (Wang and Yoon, 2021) between a teacher network trained on the simulated data and a student network for predicting change in the real data. This amounts to domain adaptation, as recently presented for 3D PCs from different modalities, including synthetic-to-real data adaptation (Cardace et al., 2023; Zahs et al., 2023) or 2D images of multiple sources adaption (Liu et al., 2022a; Liu et al., 2022b).

- Depending on the use case, different types of changes can interest a user. We proposed in the weakly supervised setting to cluster all types of changes and to let the user selecting those of interest. One perspective consists of making the network predicting a **hierarchical change map**. Hierarchical change segmentation would allow attributing several labels concerning the scale of the change. For example, to distinguish between changed buildings with large modification (e.g., totally new building) to medium changes (e.g., modification in the roof shape, house extensions, supplementary floors, ...) or even only slight modification (e.g., new dormer or balcony), it would be interesting to tag this modified building with different labels to more easily select the scale of desired change extraction. Teruggi et al. (2020) showed the sake of multi-scale hierarchical segmentation in 3D PCs for cultural heritage buildings (such as cathedrals) using machine learning. However, to the best of our knowledge, no deep learning method for hierarchical segmentation of 3D PCs exist yet.

- While bi-temporal change detection was tackled in this thesis, with the multiplication of 3D PC acquisitions, it could be interesting to further use **time series of 3D PCs**. Note that the thesis was motivated by the support of CO3D with dedicated time series analysis solutions. In the change detection task, time series help to distinguish between permanent and non-permanent changes. For example, this can be an asset for vegetation monitoring affected by seasonal variation. Also,

in geosciences, 3D PC time series are often available to monitor a specific area (e.g., the Varengeville-sur-Mer cliff erosion dataset detailed in Chapter 5, beach-dune monitoring (Vos et al., 2022), glacier monitoring (Zahs et al., 2022)). Thus, having methods to tackle multi-temporal change detection could be interesting to monitor non-abrupt long-term changes, for example. Furthermore, the use of time series may help in the proposed perspective of detecting changes in a hierarchical context. Indeed, relying on several dates may help in retrieving more general evolution tendency of certain areas. For example, an isolated change may be distinguished from a global one thanks to long term time series. This could be used for urban planing, for example among others.

Point Cloud Change Detection in the Wild

We demonstrated the relevance of our supervised methods in different scopes from urban to geosciences with erosion or landslides detection, for example. We strongly believe that our deep supervised methods, in particular *Encoder Fusion SiamKPConv*, are already mature enough. Being agnostic to the type of 3D PCs and application, supported by publicly available source code, they can solve various change detection problems if well-used. Indeed, as shown in this thesis, if a large enough annotated dataset is available (with splits containing a fair distribution of all classes of change), the method is able to predict different kind of changes. This opens new possibilities to study the evolution of our landscapes for a more global understanding of environmental dynamics.

BIBLIOGRAPHY

Achlioptas, Panos, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas (2018), «Learning representations and generative models for 3D point clouds», *in*: *International conference on machine learning*, PMLR, pp. 40–49.

Afham, Mohamed, Isuru Dissanayake, Dinithi Dissanayake, Amaya Dharmasiri, Kanchana Thilakarathna, and Ranga Rodrigo (2022), «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*, pp. 9902–9912.

Aghababaei, Hossein, Giampaolo Ferraioli, Laurent Ferro-Famil, Yué Huang, Mauro Mariotti d'Alessandro, Vito Pascazio, Gilda Schirinzi, and Stefano Tebaldini (2020), «Forest SAR tomography: Principles and applications», *in*: *IEEE geoscience and remote sensing magazine* 8.2, pp. 30–45.

Allan, Richard P, Ed Hawkins, Nicolas Bellouin, and Bill Collins (2021), *IPCC, 2021: Summary for Policymakers*.

Alliegro, Antonio, Davide Boscaini, and Tatiana Tommasi (2021), «Joint supervised and self-supervised learning for 3D real world challenges», *in*: *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, pp. 6718–6725.

Amini Amirkolaee, H and H Arefi (2019), «3D change detection in urban areas based on DCNN using a single image», *in*: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.

Ashton, Andrew D., Mike J. A. Walkden, and Mark E. Dickson (June 2011), «Equilibrium responses of cliffed coasts to changes in the rate of sea level rise», *in*: *Marine Geology* 284.1-4, pp. 217–229, ISSN: 0025-3227, DOI: 10.1016/j.margeo.2011.01.007, (visited on 01/03/2022).

Atzmon, Matan, Haggai Maron, and Yaron Lipman (2018), «Point convolutional neural networks by extension operators», *in*: *arXiv preprint arXiv:1803.10091*.

Awrangjeb, M., C.S Fraser, and G. Lu (2015), «Building change detection from LiDAR point cloud data based on connected component analysis», *in*: *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences* 2, p. 393.

Bao, Tengfei, Chenqin Fu, Tao Fang, and Hong Huo (2020), «PPCNET: A combined patch-level and pixel-level end-to-end deep network for high-resolution remote sensing image change detection», *in*: *IEEE Geoscience and Remote Sensing Letters* 17.10, pp. 1797–1801.

Behling, Robert, Sigrid Roessner, Hermann Kaufmann, and Birgit Kleinschmit (2014), «Automated spatiotemporal landslide mapping over large areas using rapideye time series data», *in*: *Remote Sensing* 6.9, pp. 8026–8055.

Berg, Paul, Minh-Tan Pham, and Nicolas Courty (2022), «Self-Supervised Learning for Scene Classification in Remote Sensing: Current State of the Art and Perspectives», *in*: *Remote Sensing* 14.16, p. 3995.

Bergamasco, Luca, Sudipan Saha, Francesca Bovolo, and Lorenzo Bruzzone (2019), «Unsupervised change-detection based on convolutional-autoencoder feature extraction», *in*: *Image and Signal Processing for Remote Sensing XXV*, vol. 11155, SPIE, pp. 325–332.

Bernard, Thomas G, Dimitri Lague, and Philippe Steer (2021), «Beyond 2D landslide inventories and their rollover: synoptic 3D inventories and volume from repeat lidar data», *in*: *Earth Surface Dynamics* 9.4, pp. 1013–1044.

Besl, Paul J and Neil D McKay (1992), «Method for registration of 3-D shapes», *in*: *Sensor fusion IV: control paradigms and data structures*, vol. 1611, Spie, pp. 586–606.

Bignot, G. (1962), «Étude sédimentologique et micropaléontologique de l'Éocène du Cap d'Ailly (près de Dieppe-Seine-Maritime)», PhD thesis, Université de Paris, (visited on 12/13/2021).

Boulch, Alexandre (2020), «ConvPoint: Continuous convolutions for point cloud processing», *in*: *Computers & Graphics* 88, pp. 24–34.

Boulch, Alexandre, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert (2018), «SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks», *in*: *Computers & Graphics* 71, pp. 189–198.

Brooks, S. M. and T. Spencer (Dec. 2010), «Temporal and spatial variations in recession rates and sediment release from soft rock cliffs, Suffolk coast, UK», *in*: *Geomorphology* 124.1-2, pp. 26–41, ISSN: 0169-555X, DOI: 10.1016/j.geomorph.2010.08.005, (visited on 01/03/2022).

Cai, Zhinan, Zhiyu Jiang, and Yuan Yuan (2021), «Task-related self-supervised learning for remote sensing image change detection», *in*: *ICASSP 2021-2021 IEEE In-*

ternational Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 1535–1539.

Cardace, Adriano, Riccardo Spezialetti, Pierluigi Zama Ramirez, Samuele Salti, and Luigi Di Stefano (2023), «Self-Distillation for Unsupervised 3D Domain Adaptation», *in*: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 4166–4177.

Caron, Mathilde (2021), «Self-supervised learning of deep visual representations», PhD thesis, Université Grenoble Alpes.

Caron, Mathilde, Piotr Bojanowski, Armand Joulin, and Matthijs Douze (2018), «Deep clustering for unsupervised learning of visual features», *in*: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132–149.

Caron, Mathilde, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin (2020), «Unsupervised learning of visual features by contrasting cluster assignments», *in*: *Advances in neural information processing systems* 33, pp. 9912–9924.

Champion, Nicolas, Didier Boldo, Marc Pierrot-Deseilligny, and Georges Stamon (2010), «2D building change detection from high resolution satelliteimagery: A two-step hierarchical method based on 3D invariant primitives», *in*: *Pattern Recognition Letters* 31.10, pp. 1138–1147.

Chang, Angel X, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. (2015), «ShapeNet: An information-rich 3D model repository», *in*: *arXiv preprint arXiv:1512.03012*.

Chaton, Thomas, Nicolas Chaulet, Sofiane Horache, and Loic Landrieu (2020), «Torch-Points3D: A Modular Multi-Task Frameworkfor Reproducible Deep Learning on 3D Point Clouds», *in*: *arXiv preprint arXiv:2010.04642*.

Chen, Dave Zhenyu, Angel X Chang, and Matthias Nießner (2020), «Scanrefer: 3D object localization in RGB-D scans using natural language», *in*: *European Conference on Computer Vision*, Springer, pp. 202–221.

Chen, Haolan, Shitong Luo, Xiang Gao, and Wei Hu (2021a), «Unsupervised learning of geometric sampling invariant representations for 3D point clouds», *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 893–903.

Chen, Hongruixuan, Chen Wu, Bo Du, and Liangpei Zhang (2019a), «Deep Siamese multi-scale convolutional network for change detection in multi-temporal VHR images», *in*:

2019 10th International Workshop on the Analysis of Multitemporal Remote Sensing Images (MultiTemp), IEEE, pp. 1–4.

Chen, Jiahao, Junfu Fan, Mengzhen Zhang, Yuke Zhou, and Chen Shen (2022a), «MSF-Net: A Multiscale Supervised Fusion Network for Building Change Detection in High-Resolution Remote Sensing Images», *in*: *IEEE Access* 10, pp. 30925–30938.

Chen, Puhua, Lei Guo, Xiangrong Zhang, Kai Qin, Wentao Ma, and Licheng Jiao (2021b), «Attention-Guided Siamese Fusion Network for Change Detection of Remote Sensing Images», *in*: *Remote Sensing* 13.22, p. 4597.

Chen, Siheng, Chaojing Duan, Yaoqing Yang, Duanshun Li, Chen Feng, and Dong Tian (2019b), «Deep unsupervised learning of 3D point clouds via graph topology inference and filtering», *in*: *IEEE transactions on image processing* 29, pp. 3183–3198.

Chen, Ye, Jinxian Liu, Bingbing Ni, Hang Wang, Jiancheng Yang, Ning Liu, Teng Li, and Qi Tian (2021c), «Shape self-correction for unsupervised point cloud understanding», *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8382–8391.

Chen, Yujin, Matthias Nießner, and Angela Dai (2022b), «4DContrast: Contrastive learning with dynamic correspondences for 3D scene understanding», *in*: *European Conference on Computer Vision*, Springer, pp. 543–560.

Chen, Yuxing and Lorenzo Bruzzone (2022), «A Self-Supervised Approach to Pixel-Level Change Detection in Bi-Temporal RS Images», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–11, DOI: 10.1109/TGRS.2022.3203897.

Choi, K., I. Lee, and S. Kim (2009), «A feature based approach to automatic change detection from LiDAR data in urban areas», *in*: *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci* 18, pp. 259–264.

Chopra, Sumit, Raia Hadsell, and Yann LeCun (2005), «Learning a similarity metric discriminatively, with application to face verification», *in*: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, IEEE, pp. 539–546.

Coletta, V, V Marsocci, and R Ravanelli (2022), «3DCD: A new dataset for 2D and 3D change detection using deep learning techniques», *in*: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 43, pp. 1349–1354.

Costa, Stéphane (1997), «Dynamique littorale et risques naturels : L'impact des aménagements, des variations du niveau marin et des modifications climatiques entre la baie de

Seine et la baie de Somme (Haute-Normandie, Picardie; France)», PhD thesis, Paris 1, (visited on 05/24/2015).

Courty, Nicolas, Rémi Flamary, Devis Tuia, and Thomas Corpetti (2016), «Optimal transport for data fusion in remote sensing», *in*: *2016 IEEE international geoscience and remote sensing symposium (IGARSS)*, IEEE, pp. 3571–3574.

Croissant, Thomas, Dimitri Lague, Philippe Davy, Tim Davies, and Philippe Steer (2017), «A precipiton-based approach to model hydro-sedimentary hazards induced by large sediment supplies in alluvial fans», *in*: *Earth Surface Processes and Landforms* 42.13, pp. 2054–2067.

Crosby, Christopher J, Ramon Arrowsmith, Viswanath Nandigam, and Chaitanya Baru (2011), «Online access and processing of LiDAR topography data», *in*: *Geoinformatics: Cyberinfrastructure for the Solid Earth Sciences*, Cambridge University Press, pp. 251–265.

Czerniawski, Thomas, Jong Won Ma, and Fernanda Leite (2021), «Automated building change detection with amodal completion of point clouds», *in*: *Automation in Construction* 124, p. 103568.

Dai, Chenguang, Zhenchao Zhang, and Dong Lin (2020), «An Object-Based Bidirectional Method for Integrated Building Extraction and Change Detection between Multimodal Point Clouds», *in*: *Remote Sensing* 12.10, p. 1680.

Daudt, Rodrigo Caye, Bertrand Le Saux, and Alexandre Boulch (2018), «Fully convolutional siamese networks for change detection», *in*: *2018 25th IEEE International Conference on Image Processing (ICIP)*, IEEE, pp. 4063–4067.

de Gélis, Iris, Thomas Bernard, Dimitri Lague, Thomas Corpetti, and Sébastien Lefèvre (2023a), «Landslide detection in 3D point clouds with deep siamese convolutional network», *in*: *2023 IEEE International Geoscience and Remote Sensing Symposium IGARSS*.

de Gélis, Iris, Zoé Bessin, Pauline Letortu, Marion Jaud, Christophe Delacourt, Stéphane Costa, Olivier Maquaire, Robert Davidson, Thomas Corpetti, and Sébastien Lefèvre (2022), «Cliff change detection using Siamese KPConv deep network on 3D point clouds», *in*: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* V-3-2022, pp. 649–656, DOI: 10.5194/isprs-annals-V-3-2022-649-2022, URL: https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/V-3-2022/649/2022/.

de Gélis, Iris, Thomas Corpetti, and Sébastien Lefèvre (2023b), «Change detection needs change information: improving deep 3D point cloud change detection», *in*: *arXiv preprint arXiv:2304.12639*, DOI: 10.48550/arXiv.2304.12639.

de Gélis, Iris, Sébastien Lefèvre, and Thomas Corpetti (2021a), «3D urban change detection with point cloud siamese networks», *in*: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLIII-B3-2021, pp. 879–886, DOI: 10.5194/isprs-archives-XLIII-B3-2021-879-2021, URL: https://hal.science/hal-03379504.

— (2021b), «Change Detection in Urban Point Clouds: An Experimental Comparison with Simulated 3D Datasets», *in*: *Remote Sensing* 13.13, ISSN: 2072-4292, DOI: 10.3390/rs13132629, URL: https://www.mdpi.com/2072-4292/13/13/2629.

— (2023c), «DC3DCD: unsupervised learning for multiclass 3D point cloud change detection», *in*: *arXiv preprint arXiv:2305.05421*, DOI: 10.48550/arXiv.2305.05421.

— (2023d), «Siamese KPConv: 3D multiple change detection from raw point clouds using deep learning», *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 197, pp. 274–291, ISSN: 0924-2716, DOI: https://doi.org/10.1016/j.isprsjprs.2023.02.001, URL: https://www.sciencedirect.com/science/article/pii/S0924271623000394.

de Gélis, Iris, Sébastien Lefèvre, Thomas Corpetti, Thomas Ristorcelli, Chloé Thénoz, and Pierre Lassalle (2021c), «Benchmarking Change Detection in Urban 3D Point Clouds», *in*: *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pp. 3352–3355, DOI: 10.1109/IGARSS47720.2021.9553018.

de Gélis, Iris, Sudipan Saha, Muhammad Shahzad, Thomas Corpetti, Sébastien Lefèvre, and Xiao Xiang Zhu (2023e), «Deep Unsupervised Learning for 3D ALS Point Clouds Change Detection», *in*: *arXiv preprint arXiv:2305.03529*, DOI: 10.48550/arXiv.2305.03529.

Dechesne, Clément, Pierre Lassalle, and Sébastien Lefèvre (2021), «Bayesian U-Net: Estimating Uncertainty in Semantic Segmentation of Earth Observation Images», *in*: *Remote Sensing* 13.19, p. 3836.

Deng, Haowen, Tolga Birdal, and Slobodan Ilic (2018), «Ppf-foldnet: Unsupervised learning of rotation invariant 3D local descriptors», *in*: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 602–618.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009), «ImageNet: A large-scale hierarchical image database», *in*: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, DOI: 10.1109/CVPR.2009.5206848.

Diestel, W (2003), *Arbaro—tree generation for povray*.

Ding, Jian, Nan Xue, Gui-Song Xia, Xiang Bai, Wen Yang, Michael Ying Yang, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, et al. (2021), «Object detection in aerial images: A large-scale benchmark and challenges», *in*: *IEEE transactions on pattern analysis and machine intelligence* 44.11, pp. 7778–7796.

Dini, GR, Karsten Jacobsen, Franz Rottensteiner, M Al Rajhi, and Christian Heipke (2012), «3D building change detection using high resolution stereo images and a GIS database», *in*: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives 39 (2012)* 39, pp. 299–304.

Dong, Huihui, Wenping Ma, Yue Wu, Maoguo Gong, and Licheng Jiao (2018), «Local descriptor learning for change detection in synthetic aperture radar images via convolutional neural networks», *in*: *IEEE Access* 7, pp. 15389–15403.

Dong, Huihui, Wenping Ma, Yue Wu, Jun Zhang, and Licheng Jiao (2020), «Self-supervised representation learning for remote sensing image change detection based on temporal prediction», *in*: *Remote Sensing* 12.11, p. 1868.

Du, Bo, Lixiang Ru, Chen Wu, and Liangpei Zhang (2019), «Unsupervised deep slow feature analysis for change detection in multi-temporal remote sensing images», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 57.12, pp. 9976–9992.

Enríquez, Alejandra R, Marta Marcos, Albert Falqués, and Dano Roelvink (2019), «Assessing beach and dune erosion and vulnerability under sea level rise: a case study in the Mediterranean Sea», *in*: *Frontiers in Marine Science* 6, p. 4.

Erdogan, Mustafa and Altan Yilmaz (2019), «Detection of building damage caused by Van Earthquake using image and Digital Surface Model (DSM) difference», *in*: *International Journal of Remote Sensing* 40.10, pp. 3772–3786.

Ericsson, Linus, Henry Gouk, Chen Change Loy, and Timothy M Hospedales (2022), «Self-Supervised Representation Learning: Introduction, advances, and challenges», *in*: *IEEE Signal Processing Magazine* 39.3, pp. 42–62.

Fan, Haoqiang, Hao Su, and Leonidas J Guibas (2017), «A point set generation network for 3D object reconstruction from a single image», *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 605–613.

Fang, Sheng, Kaiyu Li, Jinyuan Shao, and Zhe Li (2021), «SNUNet-CD: A densely connected Siamese network for change detection of VHR images», *in*: *IEEE Geoscience and Remote Sensing Letters* 19, pp. 1–5.

Feranec, Jan, Gerard Hazeu, Susan Christensen, and Gabriel Jaffrain (2007), «Corine land cover change detection in Europe (case studies of the Netherlands and Slovakia)», *in*: *Land use policy* 24.1, pp. 234–247.

Fiorucci, Marco, Peter Naylor, and Makoto Yamada (2023), *Optimal Transport for Change Detection on LiDAR Point Clouds*, DOI: 10.48550/ARXIV.2302.07025, URL: https://arxiv.org/abs/2302.07025.

Gadelha, Matheus, Rui Wang, and Subhransu Maji (2018), «Multiresolution tree networks for 3D point cloud processing», *in*: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 103–118.

Gao, Feng, Junyu Dong, Bo Li, and Qizhi Xu (2016), «Automatic change detection in synthetic aperture radar images based on PCANet», *in*: *IEEE geoscience and remote sensing letters* 13.12, pp. 1792–1796.

Gao, Xiang, Wei Hu, and Guo-Jun Qi (2020), «GraphTER: Unsupervised learning of graph transformation equivariant representations via auto-encoding node-wise transformations», *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7163–7172.

García-Rodríguez, M. J. and J. A. Malpica (2010), «Assessment of earthquake-triggered landslide susceptibility in El Salvador based on an Artificial Neural Network model», *in*: *Natural Hazards and Earth System Sciences* 10.6, pp. 1307–1315, DOI: 10.5194/nhess-10-1307-2010, URL: https://nhess.copernicus.org/articles/10/1307/2010/.

Gehrung, Joachim, Marcus Hebel, Michael Arens, and Uwe Stilla (2017), «An approach to extract moving objects from MLS data using a volumetric background representation», *in*: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* 4.

Geng, Jie, Hongyu Wang, Jianchao Fan, and Xiaorui Ma (2017), «Change detection of SAR images based on supervised contractive autoencoders and fuzzy clustering», *in*: *2017 International workshop on remote sensing with intelligent processing (RSIP)*, IEEE, pp. 1–3.

Gharibbafghi, Zeinab, Jiaojiao Tian, and Peter Reinartz (2019), «Superpixel-Based 3D Building Model Refinement and Change Detection, Using VHR Stereo Satellite Im-

agery», *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*.

Girardeau-Montaut, Daniel (2016), *CloudCompare*.

Girardeau-Montaut, Daniel, Michel Roux, Raphaël Marc, and Guillaume Thibault (2005), «Change detection on points cloud data acquired with a ground laser scanner», *in*: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 36.part 3, W19.

Girdhar, Rohit, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta (2016), «Learning a predictable and generative vector representation for objects», *in*: *European Conference on Computer Vision*, Springer, pp. 484–499.

Gong, Maoguo, Hailun Yang, and Puzhao Zhang (2017), «Feature learning and change feature classification based on deep learning for ternary change detection in SAR images», *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 129, pp. 212–225.

Gong, Maoguo, Jiaojiao Zhao, Jia Liu, Qiguang Miao, and Licheng Jiao (2015), «Change detection in synthetic aperture radar images based on deep neural networks», *in*: *IEEE transactions on neural networks and learning systems* 27.1, pp. 125–138.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2020), «Generative adversarial networks», *in*: *Communications of the ACM* 63.11, pp. 139–144.

Grill, Jean-Bastien, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. (2020), «Bootstrap your own latent-a new approach to self-supervised learning», *in*: *Advances in neural information processing systems* 33, pp. 21271–21284.

Groh, Fabian, Patrick Wieschollek, and Hendrik PA Lensch (2018), «Flex-convolution», *in*: *Asian Conference on Computer Vision*, Springer, pp. 105–122.

Guerin, Cyrielle, Renaud Binet, and Marc Pierrot-Deseilligny (2014), «Automatic detection of elevation changes by differential DSM analysis: Application to urban areas», *in*: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7.10, pp. 4020–4037.

Guiotte, Florent, Minh-Tan Pham, Romain Dambreville, Thomas Corpetti, and Sébastien Lefèvre (2020), «Semantic segmentation of lidar points clouds: rasterization beyond

digital elevation models», *in*: *IEEE Geoscience and Remote Sensing Letters* 17.11, pp. 2016–2019.

Guo, Meng-Hao, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu (2021), «PCT: Point cloud transformer», *in*: *Computational Visual Media* 7, pp. 187–199.

Guo, Yulan, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun (2020), «Deep learning for 3D point clouds: A survey», *in*: *IEEE transactions on pattern analysis and machine intelligence* 43.12, pp. 4338–4364.

Hackel, Timo, N. Savinov, L. Ladicky, Jan D. Wegner, K. Schindler, and M. Pollefeys (2017), «SEMANTIC3D.NET: A new large-scale point cloud classification benchmark», *in*: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-1-W1, pp. 91–98.

Hassani, Kaveh and Mike Haley (2019), «Unsupervised multi-task feature learning on point clouds», *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8160–8171.

He, H., M. Chen, T. Chen, and D. Li (2018), «Matching of remote sensing images with complex background variations via Siamese convolutional neural network», *in*: *Remote Sensing* 10.2, p. 355.

Hebel, Marcus, Michael Arens, and Uwe Stilla (2013), «Change detection in urban areas by object-based analysis and on-the-fly comparison of multi-view ALS data», *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 86, pp. 52–64.

Hedjam, Rachid, Abdelhamid Abdesselam, and Farid Melgani (2019), «Change detection from unlabeled remote sensing images using siamese ANN», *in*: *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, IEEE, pp. 1530–1533.

Hermosilla, Pedro, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski (2018), «Monte carlo convolution for learning on non-uniformly sampled point clouds», *in*: *ACM Transactions on Graphics (TOG)* 37.6, pp. 1–12.

Hess, Georg, Johan Jaxing, Elias Svensson, David Hagerman, Christoffer Petersson, and Lennart Svensson (2023), «Masked Autoencoder for Self-Supervised Pre-Training on Lidar Point Clouds», *in*: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 350–359.

Hock, Regine (2005), «Glacier melt: a review of processes and their modelling», *in*: *Progress in physical geography* 29.3, pp. 362–391.

Hou, Ji, Benjamin Graham, Matthias Nießner, and Saining Xie (2021), «Exploring data-efficient 3D scene understanding with contrastive scene contexts», *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15587–15597.

Hsu, Pai-Hui and Zong-Yi Zhuang (2020), «Incorporating handcrafted features into deep learning for point cloud classification», *in*: *Remote Sensing* 12.22, p. 3713.

Hua, Binh-Son, Minh-Khoi Tran, and Sai-Kit Yeung (2018), «Pointwise convolutional neural networks», *in*: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 984–993.

Huang, Qidong, Xiaoyi Dong, Dongdong Chen, Hang Zhou, Weiming Zhang, Kui Zhang, Gang Hua, and Nenghai Yu (2022a), «PointCAT: Contrastive Adversarial Training for Robust Point Cloud Recognition», *in*: *arXiv preprint arXiv:2209.07788*.

Huang, Rong, Yusheng Xu, Ludwig Hoegner, and Uwe Stilla (2022b), «Semantics-aided 3D change detection on construction sites using UAV-based photogrammetric point clouds», *in*: *Automation in Construction* 134, p. 104057.

Huang, Siyuan, Yichen Xie, Song-Chun Zhu, and Yixin Zhu (2021), «Spatio-temporal self-supervised representation learning for 3D point clouds», *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6535–6545.

James, M. R. and S. Robson (Aug. 2012), «Straightforward reconstruction of 3D surfaces and topography with a camera: Accuracy and geoscience application», *in*: *Journal of Geophysical Research-Earth Surface* 117, F03017, ISSN: 2169-9003, DOI: 10.1029/2011JF002289, (visited on 01/03/2022).

James, Mike R. and Stuart Robson (Aug. 2014), «Mitigating systematic error in topographic models derived from UAV and ground-based image networks», *in*: *Earth Surface Processes and Landforms* 39.10, pp. 1413–1420, ISSN: 0197-9337, DOI: 10.1002/esp.3609, (visited on 01/05/2022).

Jang, Yeong Jae, Kwan Young Oh, Kwang Jae Lee, and Jae Hong Oh (2019), «A study on urban change detection using D-DSM from stereo satellite data», *in*: *Journal of the Korean Society of Surveying, Geodesy, Photogrammetry and Cartography* 37.5, pp. 389–395.

Jaud, Marion, Pauline Letortu, Claire Théry, Philippe Grandjean, Stéphane Costa, Olivier Maquaire, Robert Davidson, and Nicolas Le Dantec (June 2019), «UAV survey of a coastal cliff face – Selection of the best imaging angle», en, *in*: *Measurement* 139,

pp. 10–20, ISSN: 02632241, DOI: 10.1016/j.measurement.2019.02.024, (visited on 10/04/2021).

Jaud, Marion, Sophie Passot, Pascal Allemand, Nicolas Le Dantec, Philippe Grandjean, Jérôme Ammann, and Christophe Delacourt (May 2017), «Stratégies d'optimisation d'acquisition par drone pour limiter les distorsions lors de la reconstruction 3D par les logiciels Photoscan et MicMac.», *in*: *Journées CRITEX*. Grenoble.

Jiang, Haobo, Yaqi Shen, Jin Xie, Jun Li, Jianjun Qian, and Jian Yang (2021), «Sampling network guided cross-entropy method for unsupervised point cloud registration», *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6128–6137.

Jiang, Huiwei, Xiangyun Hu, Kun Li, Jinming Zhang, Jinqi Gong, and Mi Zhang (2020), «PGA-SiamNet: Pyramid feature-based attention-guided siamese network for remote sensing orthoimagery building change detection», *in*: *Remote Sensing* 12.3, p. 484.

Jiang, Huiwei, Min Peng, Yuanjun Zhong, Haofeng Xie, Zemin Hao, Jingming Lin, Xiaoli Ma, and Xiangyun Hu (2022a), «A Survey on Deep Learning-Based Change Detection from High-Resolution Remote Sensing Images», *in*: *Remote Sensing* 14.7, p. 1552.

Jiang, Kaixuan, Wenhua Zhang, Jia Liu, Fang Liu, and Liang Xiao (2022b), «Joint Variation Learning of Fusion and Difference Features for Change Detection in Remote Sensing Images», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–18.

Jing, Longlong and Yingli Tian (2020), «Self-supervised visual feature learning with deep neural networks: A survey», *in*: *IEEE transactions on pattern analysis and machine intelligence* 43.11, pp. 4037–4058.

Jing, Longlong, Ling Zhang, and Yingli Tian (2021), «Self-supervised feature learning by cross-modality and cross-view correspondences», *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1581–1591.

Kalinicheva, Ekaterina, Jérémie Sublime, and Maria Trocan (2019), «Change detection in satellite images using reconstruction errors of joint autoencoders», *in*: *Artificial Neural Networks and Machine Learning–ICANN 2019: Image Processing: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part III*, Springer, pp. 637–648.

Kharroubi, Abderrazzaq, Florent Poux, Zouhair Ballouch, Rafika Hajji, and Roland Billen (2022), «Three Dimensional Change Detection Using Point Clouds: A Review», *in*: *Geomatics* 2.4, pp. 457–485.

Kölle, Michael, Dominik Laupheimer, Stefan Schmohl, Norbert Haala, Franz Rottensteiner, Jan Dirk Wegner, and Hugo Ledoux (2021), «The Hessigheim 3D (H3D) benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from UAV LiDAR and Multi-View-Stereo», *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 1, p. 100001.

Kontogianni, Theodora, Michael Gygli, Jasper Uijlings, and Vittorio Ferrari (2020), «Continuous adaptation for interactive object segmentation by learning from corrections», *in*: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, Springer, pp. 579–596.

Kramer, Mark A (1991), «Nonlinear principal component analysis using autoassociative neural networks», *in*: *AIChE journal* 37.2, pp. 233–243.

Krishnan, Sriram, Christopher Crosby, Viswanath Nandigam, Minh Phan, Charles Cowart, Chaitanya Baru, and Ramon Arrowsmith (2011), «OpenTopography: a services oriented architecture for community access to LIDAR topography», *in*: *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*, pp. 1–8.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2017), «Imagenet classification with deep convolutional neural networks», *in*: *Communications of the ACM* 60.6, pp. 84–90.

Ku, Tao, Sam Galanakis, Bas Boom, Remco C. Veltkamp, Darshan Bangera, Shankar Gangisetty, Nikolaos Stagakis, Gerasimos Arvanitis, and Konstantinos Moustakas (2021), «SHREC 2021: 3D point cloud change detection for street scenes», *in*: *Computers & Graphics* 99, pp. 192–200, ISSN: 0097-8493, DOI: https://doi.org/10.1016/j.cag.2021.07.004, URL: https://www.sciencedirect.com/science/article/pii/S0097849321001369.

Kussul, Nataliia, Mykola Lavreniuk, Sergii Skakun, and Andrii Shelestov (2017), «Deep learning classification of land cover and crop types using remote sensing data», *in*: *IEEE Geoscience and Remote Sensing Letters* 14.5, pp. 778–782.

Lague, D., N. Brodu, and J. Leroux (2013), «Accurate 3D comparison of complex topography with terrestrial laser scanner: Application to the Rangitikei canyon (NZ)», *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 82, pp. 10–26.

Laignel, Benoît (2003), *Caractérisation et dynamique érosive de systèmes géomorphologiques continentaux sur substrat crayeux. Exemple de l'Ouest du Bassin de Paris dans le contexte nord-ouest européen.*

Landrieu, Loic and Martin Simonovsky (2018), «Large-scale point cloud semantic segmentation with superpoint graphs», *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4558–4567.

Lang, Alex H, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom (2019), «Pointpillars: Fast encoders for object detection from point clouds», *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12697–12705.

Lebègue, L, E Cazala-Hourcade, F Languille, S Artigues, and O Melet (2020), «CO3D, A worldwide one-meter accuracy DEM for 2025», *in*: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.

Lee, Mark and Alan Clark (Jan. 2002), *Investigation and management of soft rock cliffs*, Thomas Telford Publishing, ISBN: 978-0-7277-3812-7, DOI: 10.1680/iamosrc.29859, URL: https://www.icevirtuallibrary.com/doi/book/10.1680/iamosrc.29859 (visited on 01/03/2022).

Leenstra, Marrit, Diego Marcos, Francesca Bovolo, and Devis Tuia (2021), «Self-supervised pre-training enhances change detection in Sentinel-2 imagery», *in*: *International Conference on Pattern Recognition*, Springer, pp. 578–590.

Lefèvre, S., D. Tuia, J. D. Wegner, T. Produit, and A. S. Nassar (2017), «Toward seamless multiview scene analysis from satellite to street level», *in*: *Proceedings of the IEEE* 105.10, pp. 1884–1899.

Lei, Jie, Yijie Gu, Weiying Xie, Yunsong Li, and Qian Du (2022), «Boundary Extraction Constrained Siamese Network for Remote Sensing Image Change Detection», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–13.

Lei, Yu, Xiaodong Liu, Jiao Shi, Chao Lei, and Jing Wang (2019), «Multiscale superpixel segmentation with deep features for change detection», *in*: *Ieee Access* 7, pp. 36600–36616.

Lenczner, Gaston, Adrien Chan-Hon-Tong, Bertrand Le Saux, Nicola Luminari, and Guy Le Besnerais (2022), «DIAL: Deep Interactive and Active Learning for Semantic Segmentation in Remote Sensing», *in*: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 15, pp. 3376–3389.

Letortu, Pauline, Stéphane Costa, Abdelkrim Bensaid, Jean-Michel Cador, and Hervé Quénol (2014), «Vitesses et modalités de recul des falaises crayeuses de Haute-Normandie (France): méthodologie et variabilité du recul», English; French, *in*: *Geomorphologie:*

Relief, Processus, Environnement 20.2, pp. 133–144, ISSN: 12665304 (ISSN), DOI: 10.4000/geomorphologie.10872.

Letortu, Pauline, Stéphane Costa, Olivier Maquaire, and Robert Davidson (June 2019), «Marine and subaerial controls of coastal chalk cliff erosion in Normandy (France) based on a 7-year laser scanner monitoring», *in*: *Geomorphology* 335, pp. 76–91, ISSN: 0169-555X, DOI: 10.1016/j.geomorph.2019.03.005, (visited on 04/02/2019).

Letortu, Pauline, Stéphane Costa, Olivier Maquaire, Christophe Delacourt, Emmanuel Augereau, Robert Davidson, Serge Suanez, and Jean Nabucet (Sept. 2015), «Retreat rates, modalities and agents responsible for erosion along the coastal chalk cliffs of Upper Normandy: The contribution of terrestrial laser scanning», en, *in*: *Geomorphology* 245, pp. 3–14, ISSN: 0169555X, DOI: 10.1016/j.geomorph.2015.05.007, (visited on 10/04/2021).

Letortu, Pauline, Marion Jaud, Philippe Grandjean, Jerome Ammann, Stephane Costa, Olivier Maquaire, Robert Davidson, Nicolas Le Dantec, and Christophe Delacourt (2018), «Examining high-resolution survey methods for monitoring cliff erosion at an operational scale», *in*: *Giscience & Remote Sensing* 55.4, pp. 457–476, ISSN: 1548-1603, DOI: 10.1080/15481603.2017.1408931, (visited on 12/15/2021).

Li, Chun-Liang, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov (2018a), «Point cloud GAN», *in*: *arXiv preprint arXiv:1810.05795*.

Li, Kaiyu, Zhe Li, and Sheng Fang (2020), «Siamese NestedUNet networks for change detection of high resolution satellite image», *in*: *2020 International Conference on Control, Robotics and Intelligent System*, pp. 42–48.

Li, Qiuxia, Hang Gong, Haishan Dai, Chunlai Li, Zhiping He, Wenjing Wang, Yusen Feng, Feng Han, Abudusalamu Tuniyazi, Haoyang Li, et al. (2021), «Unsupervised Hyperspectral Image Change Detection via Deep Learning Self-Generated Credible Labels», *in*: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14, pp. 9012–9024.

Li, Ruihui, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng (2019a), «Pu-GAN: a point cloud upsampling adversarial network», *in*: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7203–7212.

Li, Yangyan, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen (2018b), «PointCNN: Convolution on x-transformed points», *in*: *Advances in neural information processing systems* 31, pp. 820–830.

Li, Yangyang, Cheng Peng, Yanqiao Chen, Licheng Jiao, Linhao Zhou, and Ronghua Shang (2019b), «A deep learning method for change detection in synthetic aperture radar images», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 57.8, pp. 5751–5763.

Li, Zhenglai, Chang Tang, Lizhe Wang, and Albert Y Zomaya (2022), «Remote sensing change detection via temporal feature interaction and guided refinement», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–11.

Liang, Hanxue, Chenhan Jiang, Dapeng Feng, Xin Chen, Hang Xu, Xiaodan Liang, Wei Zhang, Zhenguo Li, and Luc Van Gool (2021), «Exploring geometry-aware contrast and clustering harmonization for self-supervised 3D object detection», *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3293–3302.

Lin, Frank and William W Cohen (2010), «Power iteration clustering», *in*: *Proceedings of the 27th International Conference on Machine Learning*.

Lin, Zhi-Hao, Sheng-Yu Huang, and Yu-Chiang Frank Wang (2021), «Learning of 3D graph convolution networks for point cloud analysis», *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.8, pp. 4212–4224.

Liu, Hao, Ying Qu, and Liqiang Zhang (2022a), «Multispectral Scene Classification via Cross-Modal Knowledge Distillation», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–12.

Liu, Jia, Maoguo Gong, Kai Qin, and Puzhao Zhang (2016), «A deep convolutional coupling network for change detection based on heterogeneous optical and radar images», *in*: *IEEE transactions on neural networks and learning systems* 29.3, pp. 545–559.

Liu, Xinhai, Zhizhong Han, Xin Wen, Yu-Shen Liu, and Matthias Zwicker (2019), «L2g auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention», *in*: *Proceedings of the 27th ACM International Conference on Multimedia*, pp. 989–997.

Liu, Yikun, Xudong Kang, Yuwen Huang, Kuikui Wang, and Gongping Yang (2022b), «Unsupervised Domain Adaptation Semantic Segmentation for Remote-Sensing Images via Covariance Attention», *in*: *IEEE Geoscience and Remote Sensing Letters* 19, pp. 1–5.

Lv, Ning, Chen Chen, Tie Qiu, and Arun Kumar Sangaiah (2018), «Deep learning and superpixel feature extraction based on contractive autoencoder for change detection in SAR images», *in*: *IEEE transactions on industrial informatics* 14.12, pp. 5530–5538.

Lyu, Xuzhe, Ming Hao, and Wenzhong Shi (2020), «Building Change Detection Using a Shape Context Similarity Model for LiDAR Data», *in*: *ISPRS International Journal of Geo-Information* 9.11, p. 678.

Ma, Lei, Yu Liu, Xueliang Zhang, Yuanxin Ye, Gaofei Yin, and Brian Alan Johnson (2019), «Deep learning in remote sensing applications: A meta-analysis and review», *in*: *ISPRS journal of photogrammetry and remote sensing* 152, pp. 166–177.

MacQueen, J (1967), «Classification and analysis of multivariate observations», *in*: *5th Berkeley Symp. Math. Statist. Probability*, University of California Los Angeles LA USA, pp. 281–297.

Malamud, Bruce D, Donald L Turcotte, Fausto Guzzetti, and Paola Reichenbach (2004), «Landslide inventories and their statistical properties», *in*: *Earth Surface Processes and Landforms* 29.6, pp. 687–711.

Malila, William A (1980), «Change vector analysis: An approach for detecting forest changes with Landsat», *in*: *LARS symposia*, p. 385.

Mao, Zan, Xinyu Tong, Ze Luo, and Honghai Zhang (2022), «MFATNet: Multi-Scale Feature Aggregation via Transformer for Remote Sensing Image Change Detection», *in*: *Remote Sensing* 14.21, p. 5379.

Marc, Odin, Robert Behling, Christoff Andermann, Jens M Turowski, Luc Illien, Sigrid Roessner, and Niels Hovius (2019), «Long-term erosion of the Nepal Himalayas by bedrock landsliding: the role of monsoons, earthquakes and giant landslides», *in*: *Earth Surface Dynamics* 7.1, pp. 107–128.

Massey, CI, D Townsend, K Jones, B Lukovic, D Rhoades, R Morgenstern, B Rosser, W Ries, J Howarth, I Hamling, et al. (2020), «Volume characteristics of landslides triggered by the MW 7.8 2016 Kaikōura Earthquake, New Zealand, derived from digital surface difference modeling», *in*: *Journal of Geophysical Research: Earth Surface* 125.7, e2019JF005163.

Masson-Delmotte, Valérie, Panmao Zhai, Anna Pirani, Sarah L. Connors, C. Péan, Sophie Berger, Nada Caud, Y. Chen, Leah Goldfarb, Melissa I. Gomis, Mengtian Huang, Katherine Leitzell, Elisabeth Lonnoy, J. B. Robin Matthews, Thomas K. Maycock, Tim Waterfield, Özge Yelekçi, R. Yu, and Botao Zhou, eds. (2021), *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*, Cambridge University Press.

Mei, Guofeng, Xiaoshui Huang, Juan Liu, Jian Zhang, and Qiang Wu (2022a), «Unsupervised Point Cloud Pre-Training Via Contrasting and Clustering», *in*: *2022 IEEE International Conference on Image Processing (ICIP)*, IEEE, pp. 66–70.

Mei, Guofeng, Cristiano Saltori, Fabio Poiesi, Jian Zhang, Elisa Ricci, Nicu Sebe, and Qiang Wu (2022b), «Data Augmentation-free Unsupervised Learning for 3D Point Cloud Understanding», *in*: *arXiv preprint arXiv:2210.02798*.

Michoud, Clément, Dario Carrea, Stéphane Costa, Marc-Henri Derron, Michel Jaboyedoff, Christophe Delacourt, Olivier Maquaire, Pauline Letortu, and Robert Davidson (Dec. 2014), «Landslide detection and monitoring capability of boat-based mobile laser scanning along Dieppe coastal cliffs, Normandy», *in*: *Landslides* 12.2, pp. 403–418, DOI: 10.1007/s10346-014-0542-5, (visited on 01/03/2022).

Mittal, Himangi, Brian Okorn, Arpit Jangid, and David Held (2021), «Self-Supervised Point Cloud Completion via Inpainting», *in*: *arXiv preprint arXiv:2111.10701*.

Mortimore, Rory N. and A. Duperret (2004), *Coastal Chalk Cliff Instability*, en, Geological Society of London, ISBN: 978-1-86239-150-5.

Mou, L., M. Schmitt, Y. Wang, and X. X. Zhu (2017), «A CNN for the identification of corresponding patches in SAR and optical imagery of urban scenes», *in*: *2017 Joint Urban Remote Sensing Event (JURSE)*, IEEE, pp. 1–4.

Murakami, H., K. Nakagawa, H. Hasegawa, T. Shibata, and E. Iwanami (1999), «Change detection of buildings using an airborne laser scanner», *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 54.2-3, pp. 148–152.

Mustapha, Ahmad, Wael Khreich, and Wasim Masr (2022), «A Deep Dive into Deep Cluster», *in*: *arXiv preprint arXiv:2207.11839*.

Nanni, Loris, Stefano Ghidoni, and Sheryl Brahnam (2017), «Handcrafted vs. non-handcrafted features for computer vision classification», *in*: *Pattern Recognition* 71, pp. 158–172.

Nijhawan, Rahul, Josodhir Das, and Balasubramanian Raman (2019), «A hybrid of deep learning and hand-crafted features based approach for snow cover mapping», *in*: *International journal of remote sensing* 40.2, pp. 759–773.

Niu, Xudong, Maoguo Gong, Tao Zhan, and Yuelei Yang (2018), «A conditional adversarial network for change detection in heterogeneous images», *in*: *IEEE Geoscience and Remote Sensing Letters* 16.1, pp. 45–49.

Noroozi, Mehdi and Paolo Favaro (2016), «Unsupervised learning of visual representations by solving jigsaw puzzles», *in*: *Computer Vision–ECCV 2016: 14th European*

Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI, Springer, pp. 69–84.

Nuth, Christopher and Andreas Kääb (2011), «Co-registration and bias corrections of satellite elevation data sets for quantifying glacier thickness change», *in*: *The Cryosphere* 5.1, pp. 271–290.

Okyay, U., J. Telling, C.L. Glennie, and W.E. Dietrich (2019), «Airborne lidar change detection: An overview of Earth sciences applications», *in*: *Earth-Science Reviews* 198, p. 102929.

Otsu, Nobuyuki (1979), «A threshold selection method from gray-level histograms», *in*: *IEEE TSMC* 9.1, pp. 62–66.

Palazzolo, Emanuele and Cyrill Stachniss (2018), «Fast image-based geometric change detection given a 3D model», *in*: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 6308–6315.

Pang, S., X. Hu, Z. W., and Y. Lu (2014), «Object-based analysis of airborne LiDAR data for building change detection», *in*: *Remote Sensing* 6.11, pp. 10733–10749.

Pelletier, Charlotte, Geoffrey I Webb, and François Petitjean (2019), «Temporal convolutional neural network for the classification of satellite image time series», *in*: *Remote Sensing* 11.5, p. 523.

Peng, Daifeng and Yongjun Zhang (2016), «Building change detection by combining Lidar data and ortho image», *in*: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 41.

Peng, Daifeng, Yongjun Zhang, and Haiyan Guan (2019), «End-to-end change detection for high resolution satellite images using improved UNet++», *in*: *Remote Sensing* 11.11, p. 1382.

Peng, Xueli, Ruofei Zhong, Zhen Li, and Qingyang Li (2020), «Optical remote sensing image change detection based on attention mechanism and image difference», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 59.9, pp. 7296–7307.

Pollock, William and Joseph Wartman (2020), «Human vulnerability to landslides», *in*: *GeoHealth* 4.10, e2020GH000287.

Pomerol, B., H. W. Bailey, C. Monciardini, and R. N. Mortimore (Dec. 1987), «Lithostratigraphy and biostratigraphy of the lewes and seaford chalks: A link across the Anglo-Paris basin at the Turonian-Senonian boundary», en, *in*: *Cretaceous Research* 8.4, pp. 289–304, ISSN: 0195-6671, DOI: 10.1016/0195-6671(87)90001-2, (visited on 12/13/2021).

Poursaeed, Omid, Tianxing Jiang, Han Qiao, Nayun Xu, and Vladimir G Kim (2020), «Self-supervised learning of point clouds via orientation estimation», *in*: *2020 International Conference on 3D Vision (3DV)*, IEEE, pp. 1018–1028.

Qi, Charles R, Hao Su, Kaichun Mo, and Leonidas J Guibas (2017a), «Pointnet: Deep learning on point sets for 3D classification and segmentation», *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660.

Qi, Charles Ruizhongtai, Li Yi, Hao Su, and Leonidas J Guibas (2017b), «Pointnet++: Deep hierarchical feature learning on point sets in a metric space», *in*: *Advances in neural information processing systems* 30.

Qin, R., J. Tian, and P. Reinartz (2016), «3D change detection – approaches and applications», *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 122, pp. 41–56.

Remelli, Edoardo, Pierre Baque, and Pascal Fua (2019), «Neuralsampler: Euclidean point cloud auto-encoder and sampler», *in*: *arXiv preprint arXiv:1901.09394*.

Ren, Yazhou, Jingyu Pu, Zhimeng Yang, Jie Xu, Guofeng Li, Xiaorong Pu, Philip S Yu, and Lifang He (2022), «Deep clustering: A comprehensive survey», *in*: *arXiv preprint arXiv:2210.04142*.

Rethage, Dario, Johanna Wald, Jurgen Sturm, Nassir Navab, and Federico Tombari (2018), «Fully-convolutional point networks for large-scale point clouds», *in*: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 596–611.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015), «U-net: Convolutional networks for biomedical image segmentation», *in*: *International Conference on Medical image computing and computer-assisted intervention*, Springer, pp. 234–241.

Rottensteiner, Franz (2008), «Automated updating of building data bases from digital surface models and multi-spectral images: Potential and limitations», *in*: *ISPRS Congress, Beijing, China*, vol. 37, pp. 265–270.

Roynard, X., J.-E. Deschaud, and F. Goulette (2016), «Fast and robust segmentation and classification for change detection in urban point clouds», *in*: *ISPRS Archives* XLI-B3, pp. 693–699.

Roynard, Xavier, Jean-Emmanuel Deschaud, and François Goulette (2018), «Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification», *in*: *Int. J. of Robotics Research* 37.6, pp. 545–557, DOI: 10.1177/0278364918767506.

Rußwurm, Marc, Sherrie Wang, Marco Korner, and David Lobell (2020), «Meta-learning for few-shot land cover classification», *in*: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition workshops*, pp. 200–201.

Saha, Sudipan, Francesca Bovolo, and Lorenzo Bruzzone (2019), «Unsupervised deep change vector analysis for multiple-change detection in VHR images», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 57.6, pp. 3677–3693.

— (2020), «Change detection in image time-series using unsupervised LSTM», *in*: *IEEE Geoscience and Remote Sensing Letters*.

Saha, Sudipan, Patrick Ebel, and Xiao Xiang Zhu (2021), «Self-supervised multisensor change detection», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–10.

Sande, Corné Van Der, Sylvie Soudarissanane, and Kourosh Khoshelham (2010), «Assessment of relative accuracy of AHN-2 laser scanning data using planar features», *in*: *Sensors* 10.9, pp. 8198–8214.

Sandric, Ionut, Bogdan Mihai, Ionut Savulescu, Bogdan Suditu, and Zenaida Chitu (2007), «Change Detection Analysis for Urban Development in Bucharest-Romania, using High Resolution Satellite Imagery», *in*: *2007 Urban Remote Sensing Joint Event*, IEEE, pp. 1–8, DOI: 10.1109/URS.2007.371848.

Sanghi, Aditya (2020), «Info3d: Representation learning on 3D objects using mutual information maximization and contrastive learning», *in*: *European Conference on Computer Vision*, Springer, pp. 626–642.

Sauder, Jonathan and Bjarne Sievers (2019), «Self-supervised deep learning on point clouds by reconstructing space», *in*: *Advances in Neural Information Processing Systems* 32.

Sculley, David (2010), «Web-scale k-means clustering», *in*: *Proceedings of the 19th international conference on World wide web*, pp. 1177–1178.

Seydi, Seyd Teymoor and Mahdi Hasanlou (2021), «A new structure for binary and multiple hyperspectral change detection based on spectral unmixing and convolutional neural network», *in*: *Measurement* 186, p. 110137.

Shafique, Ayesha, Guo Cao, Zia Khan, Muhammad Asad, and Muhammad Aslam (2022), «Deep learning-based change detection in remote sensing images: a review», *in*: *Remote Sensing* 14.4, p. 871.

Sharma, Abhishek, Oliver Grau, and Mario Fritz (2016), «Vconv-dae: Deep volumetric shape learning without object labels», *in*: *European conference on computer vision*, Springer, pp. 236–250.

Shi, Shaoshuai, Xiaogang Wang, and Hongsheng Li (2019), «Pointrcnn: 3D object proposal generation and detection from point cloud», *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 770–779.

Shi, Wenzhong, Min Zhang, Rui Zhang, Shanxiong Chen, and Zhao Zhan (2020a), «Change detection based on artificial intelligence: State-of-the-art and challenges», *in*: *Remote Sensing* 12.10, p. 1688.

Shi, Yi, Mengchen Xu, Shuaihang Yuan, and Yi Fang (2020b), «Unsupervised deep shape descriptor with point distribution learning», *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9353–9362.

Shirowzhan, S., S.M.E Sepasgozar, H. Li, J. Trinder, and P. Tang (2019), «Comparative analysis of machine learning and point-based algorithms for detecting 3D changes in buildings over time using bi-temporal lidar data», *in*: *Automation in Construction* 105, p. 102841.

Siddiqui, F. U. and M. Awrangjeb (2017), «A novel building change detection method using 3D building models», *in*: *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, IEEE, pp. 1–8.

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

Slott, Jordan M., A. Brad Murray, Andrew D. Ashton, and Thomas J. Crowley (Sept. 2006), «Coastline responses to changing storm patterns», *in*: *Geophysical Research Letters* 33.18, p. L18404, ISSN: 0094-8276, DOI: 10.1029/2006GL027445, (visited on 01/03/2022).

Sofina, Natalia and Manfred Ehlers (2016), «Building change detection using high resolution remotely sensed data and GIS», *in*: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 9.8, pp. 3430–3438.

Song, Ahram, Jaewan Choi, Youkyung Han, and Yongil Kim (2018), «Change detection in hyperspectral images using recurrent 3D fully convolutional networks», *in*: *Remote Sensing* 10.11, p. 1827.

Song, Lei, Min Xia, Junlan Jin, Ming Qian, and Yonghong Zhang (2021), «SUACDNet: Attentional change detection network based on siamese U-shaped structure», *in*: *International Journal of Applied Earth Observation and Geoinformation* 105, p. 102597.

Stal, C., F. Tack, P. De Maeyer, A. De Wulf, and R. Goossens (2013), «Airborne photogrammetry and lidar for DSM extraction and 3D change detection over an urban area–a comparative study», *in*: *International Journal of Remote Sensing* 34.4, pp. 1087–1110.

Stilla, Uwe and Yusheng Xu (2023), «Change detection of urban objects using 3D point clouds: A review», *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 197, pp. 228–255.

Sun, Yongbin, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma (2020), «Pointgrow: Autoregressively learned point cloud generation with self-attention», *in*: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 61–70.

Swirad, Zuzanna M and Adam P Young (2021), «Automating coastal cliff erosion measurements from large-area LiDAR datasets in California, USA», *in*: *Geomorphology*, p. 107799.

Tang, Lulu, Ke Chen, Chaozheng Wu, Yu Hong, Kui Jia, and Zhi-Xin Yang (2020), «Improving semantic analysis on point clouds via auxiliary supervision of local geometric priors», *in*: *IEEE Transactions on Cybernetics*.

Tang, Xu, Huayu Zhang, Lichao Mou, Fang Liu, Xiangrong Zhang, Xiao Xiang Zhu, and Licheng Jiao (2021), «An unsupervised remote sensing change detection method based on multiscale graph convolutional network and metric learning», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–15.

Tchapmi, Lyne, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese (2017), «Segcloud: Semantic segmentation of 3D point clouds», *in*: *2017 international conference on 3D vision (3DV)*, IEEE, pp. 537–547.

Teo, Tee-Ann and Tian-Yuan Shih (2013), «Lidar-based change detection and change-type determination in urban areas», *in*: *International Journal of Remote Sensing* 34.3, pp. 968–981.

Teruggi, Simone, Eleonora Grilli, Michele Russo, Francesco Fassi, and Fabio Remondino (2020), «A hierarchical machine learning approach for multi-level and multi-resolution 3D point cloud classification», *in*: *Remote Sensing* 12.16, p. 2598.

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

Thomee, Bart, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li (2016), «YFCC100M: The new data in multimedia research», *in*: *Communications of the ACM* 59.2, pp. 64–73.

Touati, R, M Mignotte, and M Dahmane (2020a), «Partly Uncoupled Siamese Model for Change Detection from Heterogeneous Remote Sensing Imagery», *in*: *Journal of Remote sensing and GIS* 9.1.

Touati, Redha, Max Mignotte, and Mohamed Dahmane (2020b), «Anomaly feature learning for unsupervised change detection in heterogeneous images: A deep sparse residual model», *in*: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13, pp. 588–600.

Tran, T.H.G., C. Ressl, and N. Pfeifer (2018), «Integrated change detection and classification in urban areas based on airborne laser scanning point clouds», *in*: *Sensors* 18.2, p. 448.

Valsesia, Diego, Giulia Fracastoro, and Enrico Magli (2018), «Learning localized generative models for 3D point clouds via graph convolution», *in*: *International conference on learning representations*.

Vandenhende, Simon, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool (2021), «Multi-task learning for dense prediction tasks: A survey», *in*: *IEEE transactions on pattern analysis and machine intelligence* 44.7, pp. 3614–3633.

Varnes, David J (1978), «Slope movement types and processes», *in*: *Special report* 176, pp. 11–33.

Varney, Nina, Vijayan K Asari, and Quinn Graehling (2020), «DALES: a large-scale aerial LiDAR data set for semantic segmentation», *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 186–187.

Ventura, Guido, Giuseppe Vilardo, Carlo Terranova, and Eliana Bellucci Sessa (2011), «Tracking and evolution of complex active landslides by multi-temporal airborne LiDAR data: The Montaguto landslide (Southern Italy)», *in*: *Remote Sensing of Environment* 115.12, pp. 3237–3248.

Vetrivel, Anand, Markus Gerke, Norman Kerle, Francesco Nex, and George Vosselman (2018), «Disaster damage detection through synergistic use of deep learning and 3D point cloud features derived from very high resolution oblique aerial images, and multiple-kernel-learning», *in*: *ISPRS journal of photogrammetry and remote sensing* 140, pp. 45–59.

Vos, Sander, Katharina Anders, Mieke Kuschnerus, Roderik Lindenbergh, Bernhard Höfle, Stefan Aarninkhof, and Sierd de Vries (2022), «A high-resolution 4D terrestrial laser scan dataset of the Kijkduin beach-dune system, The Netherlands», *in*: *Scientific Data* 9.1, pp. 1–11.

Vu, T. T., M. Matsuoka, and F. Yamazaki (2004), «LIDAR-based change detection of buildings in dense urban areas», *in*: *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*, vol. 5, IEEE, pp. 3413–3416.

Wang, Hanchen, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt Kusner (2020a), «Pre-training by completing point clouds», *in*.

Wang, Jue, Yanfei Zhong, and Liangpei Zhang (2023), «Change Detection Based on Supervised Contrastive Learning for High-Resolution Remote Sensing Imagery», *in*: *IEEE Transactions on Geoscience and Remote Sensing*.

Wang, Lei, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan (2019a), «Graph attention convolution for point cloud semantic segmentation», *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10296–10305.

Wang, Lin and Kuk-Jin Yoon (2021), «Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks», *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, Moyang, Kun Tan, Xiuping Jia, Xue Wang, and Yu Chen (2020b), «A deep siamese network with hybrid convolutional feature extraction module for change detection based on multi-sensor remote sensing images», *in*: *Remote Sensing* 12.2, p. 205.

Wang, Peng-Shuai, Yu-Qi Yang, Qian-Fang Zou, Zhirong Wu, Yang Liu, and Xin Tong (2021), «Unsupervised 3D learning for shape analysis via multiresolution instance discrimination», *in*: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 4, pp. 2773–2781.

Wang, Shenlong, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun (2018), «Deep parametric continuous convolutional neural networks», *in*: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2589–2597.

Wang, Xue and Peijun Li (2020), «Extraction of urban building damage using spectral, height and corner information from VHR satellite images and airborne LiDAR data», *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 159, pp. 322–336.

Wang, Yi, Conrad M Albrecht, Nassim Ait Ali Braham, Lichao Mou, and Xiao Xiang Zhu (2022), «Self-supervised learning in remote sensing: A review», *in*: *arXiv preprint arXiv:2206.13188*.

Wang, Yue, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon (2019b), «Dynamic graph cnn for learning on point clouds», *in*: *ACM Transactions On Graphics (TOG)* 38.5, pp. 1–12.

Wang, Zhihao, Jian Chen, and Steven CH Hoi (2020c), «Deep learning for image super-resolution: A survey», *in*: *IEEE transactions on pattern analysis and machine intelligence* 43.10, pp. 3365–3387.

Warth, G., A. Braun, C. Bödinger, V. Hochschild, and F. Bachofer (2019), «DSM-based identification of changes in highly dynamic urban agglomerations», *in*: *European Journal of Remote Sensing* 52.1, pp. 322–334.

Waser, LT, Emmanuel Baltsavias, H Eisenbeiss, C Ginzler, Armin Grün, M Kuechler, and P Thee (2007), «Change detection in mire ecosystems: assessing changes of forest area using airborne remote sensing data», *in*: *International archives of the photogrammetry, remote sensing and spatial information sciences* 36.7/C50, pp. 313–318.

Wen, Xin, Tianyang Li, Zhizhong Han, and Yu-Shen Liu (2020), «Point cloud completion by skip-attention network with hierarchical folding», *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1939–1948.

Widyaningrum, Elyta, Qian Bai, Marda K Fajari, and Roderik C Lindenbergh (2021), «Airborne laser scanning point cloud classification using the DGCNN deep learning method», *in*: *Remote Sensing* 13.5, p. 859.

Wu, Bichen, Alvin Wan, Xiangyu Yue, and Kurt Keutzer (2018), «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)*, IEEE, pp. 1887–1893.

Wu, Chen, Hongruixuan Chen, Bo Du, and Liangpei Zhang (2021), «Unsupervised change detection in multitemporal VHR images based on deep kernel PCA convolutional mapping network», *in*: *IEEE Transactions on Cybernetics*.

Wu, Jiajun, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum (2016), «Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling», *in*: *Advances in neural information processing systems* 29.

Wu, Zhirong, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao (2015), «3D ShapeNets: A deep representation for volumetric shapes», *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920.

Xiang, Shao, Mi Wang, Xiaofan Jiang, Guangqi Xie, Zhiqi Zhang, and Peng Tang (2021), «Dual-task semantic change detection for remote sensing images using the generative change field module», *in*: *Remote Sensing* 13.16, p. 3336.

Xiao, Aoran, Jiaxing Huang, Dayan Guan, Xiaoqin Zhang, Shijian Lu, and Ling Shao (2023), «Unsupervised Point Cloud Representation Learning with Deep Neural Networks: A Survey», *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xie, Saining, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany (2020), «Pointcontrast: Unsupervised pre-training for 3D point cloud understanding», *in*: *European conference on computer vision*, Springer, pp. 574–591.

Xu, Hao, Liang Cheng, Manchun Li, Yanming Chen, and Lishan Zhong (2015a), «Using octrees to detect changes to buildings and trees in the urban environment from airborne LiDAR data», *in*: *Remote Sensing* 7.8, pp. 9682–9704.

Xu, Quanfu, Keming Chen, Xian Sun, Yue Zhang, Hao Li, and Guangluan Xu (2020), «Pseudo-Siamese Capsule Network for Aerial Remote Sensing Images Change Detection», *in*: *IEEE Geoscience and Remote Sensing Letters*.

Xu, Sudan, George Vosselman, and Sander Oude Elberink (2015b), «Detection and classification of changes in buildings from airborne laser scanning data», *in*: *Remote sensing* 7.12, pp. 17051–17076.

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

Yang, Guandao, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan (2019), «Pointflow: 3D point cloud generation with continuous normalizing flows», *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4541–4550.

Yang, Juyoung, Pyunghwan Ahn, Doyeon Kim, Haeil Lee, and Junmo Kim (2021), «Progressive seed generation auto-encoder for unsupervised point cloud learning», *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6413–6422.

Yang, Xiangli, Zixing Song, Irwin King, and Zenglin Xu (2022), «A survey on deep semi-supervised learning», *in*: *IEEE Transactions on Knowledge and Data Engineering*.

Yang, Yaoqing, Chen Feng, Yiru Shen, and Dong Tian (2018), «Foldingnet: Point cloud auto-encoder via deep grid deformation», *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 206–215.

Yin, Hongyang, Liguo Weng, Yan Li, Min Xia, Kai Hu, Haifeng Lin, and Ming Qian (2023), «Attention-guided siamese networks for change detection in high resolution remote sensing images», *in*: *International Journal of Applied Earth Observation and Geoinformation* 117, p. 103206, ISSN: 1569-8432, DOI: https://doi.org/10.1016/j.jag.2023.103206, URL: https://www.sciencedirect.com/science/article/pii/S1569843223000286.

Young, A. P. and S. A. Ashford (Mar. 2006), «Application of airborne LIDAR for seacliff volumetric change and beach-sediment budget contributions», *in*: *Journal of Coastal Research* 22.2, pp. 307–318, ISSN: 0749-0208, DOI: 10.2112/05-0548.1, (visited on 01/04/2022).

Young, Adam P., R. E. Flick, R. Gutierrez, and R. T. Guza (Nov. 2009), «Comparison of short-term seacliff retreat measurement methods in Del Mar, California», *in*: *Geomorphology* 112.3-4, pp. 318–323, ISSN: 0169-555X, DOI: 10.1016/j.geomorph.2009.06.018, (visited on 01/03/2022).

Young, Adam P., M. J. Olsen, N. Driscoll, R. E. Flick, R. Gutierrez, R. T. Guza, E. Johnstone, and F. Kuester (Apr. 2010), «Comparison of Airborne and Terrestrial Lidar Estimates of Seacliff Erosion in Southern California», *in*: *Photogrammetric Engineering and Remote Sensing* 76.4, pp. 421–427, ISSN: 0099-1112, DOI: 10.14358/PERS.76.4.421, (visited on 01/03/2022).

Yu, Xiao, Junfu Fan, Peng Zhang, Liusheng Han, Dafu Zhang, and Guangwei Sun (2019), «Multi-scale convolutional neural network for remote sensing image change detection», *in*: *Geoinformatics in Sustainable Ecosystem and Society*, Springer, pp. 234–242.

Yuan, Wentao, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert (2018), «Pcn: Point completion network», *in*: *2018 International Conference on 3D Vision (3DV)*, IEEE, pp. 728–737.

Zagoruyko, Sergey and Nikos Komodakis (2015), «Learning to compare image patches via convolutional neural networks», *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4353–4361.

Zahs, Vivien, Katharina Anders, Julia Kohns, Alexander Stark, and Bernhard Höfle (2023), «Classification of structural building damage grades from multi-temporal pho-

togrammetric point clouds using a machine learning model trained on virtual laser scanning data», *in*: *arXiv preprint arXiv:2302.12591*.

Zahs, Vivien, Lukas Winiwarter, Katharina Anders, Jack G Williams, Martin Rutzinger, and Bernhard Höfle (2022), «Correspondence-driven plane-based M3C2 for lower uncertainty in 3D topographic change quantification», *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 183, pp. 541–559.

Zhan, Tao, Maoguo Gong, Xiangming Jiang, and Shuwei Li (2018), «Log-based transformation feature learning for change detection in heterogeneous images», *in*: *IEEE Geoscience and Remote Sensing Letters* 15.9, pp. 1352–1356.

Zhan, Yang, Kun Fu, Menglong Yan, Xian Sun, Hongqi Wang, and Xiaosong Qiu (2017), «Change detection based on deep siamese convolutional network for optical aerial images», *in*: *IEEE Geoscience and Remote Sensing Letters* 14.10, pp. 1845–1849.

Zhang, Jinming, Xiangyun Hu, and Hengming Dai (2021a), «Unsupervised Learning of ALS Point Clouds for 3-D Terrain Scene Clustering», *in*: *IEEE Geoscience and Remote Sensing Letters* 19, pp. 1–5.

Zhang, Ling and Zhigang Zhu (2019), «Unsupervised feature learning for point cloud understanding by contrasting and clustering using graph convolutional neural networks», *in*: *2019 international conference on 3D vision (3DV)*, IEEE, pp. 395–404.

Zhang, Mengya, Guangluan Xu, Keming Chen, Menglong Yan, and Xian Sun (2018a), «Triplet-based semantic relation learning for aerial remote sensing image change detection», *in*: *IEEE Geoscience and Remote Sensing Letters* 16.2, pp. 266–270.

Zhang, Min and Wenzhong Shi (2020), «A feature difference convolutional neural network-based change detection method», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 58.10, pp. 7232–7246.

Zhang, Puzhao, Maoguo Gong, Linzhi Su, Jia Liu, and Zhizhou Li (2016), «Change detection based on deep feature representation and mapping transformation for multi-spatial-resolution remote sensing images», *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 116, pp. 24–41.

Zhang, Xinzheng, Hang Su, Ce Zhang, Xiaowei Gu, Xiaoheng Tan, and Peter M Atkinson (2021b), «Robust unsupervised small area change detection from SAR imagery using deep learning», *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 173, pp. 79–94.

Zhang, Yu and Qiang Yang (2021), «A survey on multi-task learning», *in*: *IEEE Transactions on Knowledge and Data Engineering* 34.12, pp. 5586–5609.

Zhang, Z., G. Vosselman, M. Gerke, C. Persello, D. Tuia, and M.Y. Yang (2019), «Detecting building changes between airborne laser scanning and photogrammetric data», *in*: *Remote sensing* 11.20, p. 2417.

Zhang, Z., G. Vosselman, M. Gerke, D. Tuia, and M. Y. Yang (2018b), «Change detection between multimodal remote sensing data using siamese CNN», *in*: *arXiv preprint arXiv:1807.09562*.

Zhang, Zaiwei, Rohit Girdhar, Armand Joulin, and Ishan Misra (2021c), «Self-supervised pretraining of 3D features on any point-cloud», *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10252–10263.

Zhao, Wenzhi, Lichao Mou, Jiage Chen, Yanchen Bo, and William J Emery (2019a), «Incorporating metric learning and adversarial network for seasonal invariant change detection», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 58.4, pp. 2720–2731.

Zhao, Yongheng, Tolga Birdal, Haowen Deng, and Federico Tombari (2019b), «3D point capsule networks», *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1009–1018.

Zheng, Xiangtao, Xiumei Chen, Xiaoqiang Lu, and Bangyong Sun (2021), «Unsupervised change detection by cross-resolution difference learning», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–16.

Zheng, Xiaolong, Dongdong Guan, Bangjie Li, Zhengsheng Chen, and Lefei Pan (2023), «Global and Local Graph-Based Difference Image Enhancement for Change Detection», *in*: *Remote Sensing* 15.5, p. 1194.

Zhiheng, Kang and Li Ning (2019), «PyramNet: Point cloud pyramid attention network and graph embedding module for classification and segmentation», *in*: *arXiv preprint arXiv:1906.03299*.

Zhou, Haoran, Yidan Feng, Mingsheng Fang, Mingqiang Wei, Jing Qin, and Tong Lu (2021a), «Adaptive graph convolution for point cloud analysis», *in*: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4965–4974.

Zhou, Huaping, Minglong Song, and Kelei Sun (2023), «A Full-Scale Feature Fusion Siamese Network for Remote Sensing Change Detection», *in*: *Electronics* 12.1, p. 35.

Zhou, Liang, Yuanxin Ye, Tengfeng Tang, Ke Nan, and Yao Qin (2021b), «Robust Matching for SAR and Optical Images Using Multiscale Convolutional Gradient Features», *in*: *IEEE Geoscience and Remote Sensing Letters* 19, pp. 1–5.

Zhou, Sheng, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, Martin Ester, et al. (2022), «A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions», *in*: *arXiv preprint arXiv:2206.07579*.

Zhou, Zongwei, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang (2018), «Unet++: A nested u-net architecture for medical image segmentation», *in*: *Deep learning in medical image analysis and multimodal learning for clinical decision support*, Springer, pp. 3–11.

Zhu, Jingwei, Joachim Gehrung, Rong Huang, Björn Borgmann, Zhenghao Sun, Ludwig Hoegner, Marcus Hebel, Yusheng Xu, and Uwe Stilla (2020), «TUM-MLS-2016: An annotated mobile LiDAR dataset of the TUM city campus for semantic point cloud interpretation in urban areas», *in*: *Remote Sensing* 12.11, p. 1875.

Zhu, Xiao Xiang and Richard Bamler (2010), «Very high resolution spaceborne SAR tomography in urban environment», *in*: *IEEE Transactions on Geoscience and Remote Sensing* 48.12, pp. 4296–4308.

Zhu, Xiao Xiang, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer (2017), «Deep learning in remote sensing: A comprehensive review and list of resources», *in*: *IEEE Geoscience and Remote Sensing Magazine* 5.4, pp. 8–36.

List of abbreviations

2D two-dimensional

3D three-dimensional

3D-PcD 3D point cloud differencing

AE auto-encoder

AHD average height difference

AHN Actueel Hoogtebestand Nederland

AHN-CD AHN Change Detection

AI artificial intelligence

ALS Aerial Laser Scanning

BYOL bootstrap your own latent

C2C cloud-to-cloud

CC connected component

CPU central processing unit

CNN Convolutional Neural Network

CO3D 3D Optical Constellation

CVA Change Vector Analysis

DC3DCD DeepCluster 3D Change Detection

DCVA Deep Change Vector Analysis

DEM Digital Elevation Model

DGPS Differential Global Positioning System

DoD Difference of DEM

DSM Digital Surface Model

DSMd DSMs difference

DTM Digital Terrain Model

EF early fusion

EdgeConv edge convolution

FCN fully convolutional network

FF feed forward

GAN Generative Adversarial Network

GCN Graph Convolution Network

GCP ground control point

GPU graphics processing unit

GSD ground sampling distance

H3D Hessigheim 3D

HD high density

HGI-CD Hybrid Graph Inception Change Detection

ICP iterative closest point

IoU Intersection over Union

kNN k-Nearest Neighbors

KPConv Kernel Point Convolution

KP-CNN Kernel Point – Convolutional Neural Network

KP-FCNN Kernel Point – Fully Convolutional Neural Network

LETG Littoral, Environnement, Géomatique, Télédétection

LiDAR Light Detection And Ranging

LoD2 Level of Detail 2

M3C2 Multi-Scale Model-to-Model Cloud Comparison

mAcc mean of accuracy

MC Multi-Class

mIoU mean of IoU

MLP Multi-Layer Perceptron

MLS Mobile Laser Scanning

MS Multi-Sensor

NDVI Normalized Difference Vegetation Index

NLL negative log-likelihood

NMI normalized mutual information

OSUR Observatoire des Sciences de l'Univers de Rennes

PC Point Cloud

PCA Principal Component Analysis

PDAL Point Data Abstraction Library

PIC power iteration clustering

PoChaDeHH Point Cloud Change Detection with Hierarchical Histograms

RANSAC Random Sample Consensus

RF Random Forest

RGB Red Green Blue

RGB-D Red Green Blue – Depth

SAR Synthetic Aperture Radar

SfM Structure-from-Motion

SfM-MVS Structure-from-Motion Multiview Stereo Photogrammetry

SGD Stochastic Gradient Descent

SHREC21 Shape Retrieval Challenge 2021

SiamGCN Siamese Graph Convolutional Network

SSL Self-Supervised Learning

SSL-DCVA Self-Supervised Learning - Deep Change Vector Analysis

SSST Supervised Semantic Segmentation Training

SSST-DCVA Supervised Semantic Segmentation Training - Deep Change Vector Analysis

TLS Terrestrial Laser Scanning

TP terrestrial photogrammetry

UAV unmanned aerial vehicle

VTK Visualisation ToolKit

List of Figures

List of Tables

COLLEGE MATHSTIC
DOCTORAL BRETAGNE
BRETAGNE OCEANE

Université Bretagne Sud
ubs

Titre : Apprentissage profond pour la détection de changements dans des nuages de points 3D

Mot clés : Détection de changements, nuages de points 3D, apprentissage profond

Résumé : L'époque contemporaine s'accompagne de changements toujours plus rapides et fréquents de nos paysages, qu'ils soient causés par des processus géomorphologiques ou par des activités humaines. Le suivi de ces évolutions nécessite une modélisation régulière de notre environnement. Plutôt que se limiter à une conception bidimensionnelle, il parait judicieux d'utiliser des nuages de points 3D. La complexité de ce format de données rend néanmoins nécessaire la création de méthodologies spécifiques pour leur analyse. Aussi, l'apprentissage profond apparait comme la solution adéquate pour traiter les observations 3D de la Terre. Cette thèse se concentre donc sur la détection de changements dans des nuages de points 3D par apprentissage profond.

Dans un premier temps, un simulateur de nuages de points 3D en milieu urbain a été développé pour générer aléatoirement des jeux de données avec une évolution réaliste de l'environnement urbain. Après une comparaison expérimentale des méthodes existantes, des architectures Siamoises sont proposées pour la détection supervisée de changements tant dans le milieu urbain qu'en géosciences en utilisant des convolutions à points noyaux (KPConv). Afin de réduire l'annotation fastidieuse des données, la thèse s'intéresse aussi aux méthodes faiblement supervisées avec l'apprentissage par transfert, l'auto-supervision et le clustering profond. Bien que ces méthodes se révèlent prometteuses, une importance particulière doit être portée à la conception de l'architecture profonde.

Title: Deep learning for change detection in 3D point clouds

Keywords: Change detection, 3D point clouds, deep learning

Abstract: Whether caused by geomorphic processes or by human activities, contemporary times are accompanied by ever more rapid and frequent changes in our landscapes. Monitoring these changes requires regular modeling of our environment. Rather than limiting ourselves to a two-dimensional representation, it seems appropriate to use 3D data to embody our world, using point clouds for example. However, the complexity of this data format makes it necessary to create specific methodologies for their analysis. Therefore, deep learning appears to be the appropriate solution to process 3D observations of the Earth. This thesis focuses on change detection in 3D point clouds with deep learning.

First, a 3D point cloud simulator in an urban environment has been developed. It allows to randomly generate datasets with a realistic evolution of the urban environment. After an experimental comparison of the state-of-the-art methods, this thesis proposes Siamese architectures for supervised change detection both in urban environment and in geosciences using kernel point convolutions (KPConv). Finally, in order to reduce tedious data annotation, the thesis focuses on weakly supervised methods with transfer learning, self-supervision and deep clustering. These methods are promising in this context, nevertheless a particular importance must be given to the design of the deep architecture.