Siamese KPConv: 3D multiple change detection from raw point clouds using deep learning

Iris de Gélis, Sébastien Lefèvre, Thomas Corpetti

Highlights

Siamese KPConv: 3D multiple change detection from raw point clouds using deep learning

Iris de Gélis,Sébastien Lefèvre,Thomas Corpetti

- Siamese KPConv, the first deep architecture for multiple change detection at point scale over 3D point clouds
- Urb3DCD-V2, a novel simulated dataset to evaluate 3D point cloud change detection methods
- Pre-training Siamese KPConv on simulated data greatly speeds up the training on real data

Siamese KPConv: 3D multiple change detection from raw point clouds using deep learning

Iris de Gélis[a,b,*], Sébastien Lefèvre[b] and Thomas Corpetti[c]

[a]Magellium, Toulouse, F-31000, France
[b]Université Bretagne Sud, IRISA UMR 6074, Vannes, F-56000, France
[c]CNRS, LETG UMR 6554, Rennes, F-35000, France

ARTICLE INFO

Keywords:
3D point clouds
Change detection
Deep learning
Siamese network
3D Kernel Point Convolution

ABSTRACT

This study is concerned with urban change detection and categorization in point clouds. In such situations, objects are mainly characterized by their vertical axis, and the use of native 3D data such as 3D Point Clouds (PCs) is, in general, preferred to rasterized versions because of significant loss of information implied by any rasterization process. Yet, for obvious practical reasons, most existing studies only focus on 2D images for change detection purpose. In this paper, we propose a method capable of performing change detection directly within 3D data. Despite recent deep learning developments in remote sensing, to the best of our knowledge there is no such method to tackle multi-class change segmentation that directly processes raw 3D PCs. Thereby, based on advances in deep learning for change detection in 2D images and for analysis of 3D point clouds, we propose a deep Siamese KPConv network that deals with raw 3D PCs to perform change detection and categorization in a single step. Experimental results are conducted on synthetic and real data of various kinds (LiDAR, multi-sensors). Tests performed on simulated low density LiDAR and multi-sensor datasets show that our proposed method can obtain up to 80% of mean of IoU over classes of changes, leading to an improvement ranging from 10% to 30% over the state-of-the-art. A similar range of improvements is attainable on real data. Then, we show that pre-training Siamese KPConv on simulated PCs allows us to greatly reduce (more than 3,000×) the annotations required on real data. This is a highly significant result to deal with practical scenarios. Finally, an adaptation of Siamese KPConv is realized to deal with change classification at PC scale. Our network overtakes the current state-of-the-art deep learning method by 23% and 15% of mean of IoU when assessed on synthetic and public Change3D datasets, respectively.

1. Introduction

Because of the constant growth of the world population and human activities, landscapes are continuously evolving, in particular within cities. The past decades have seen a regular increase in urban areas across the entire world. 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), change detection and categorization is a crucial issue.

In urban environments, most objects (buildings, vegetation, etc.) are mainly characterized by their vertical axis. Therefore, we advocate for the use of 3-Dimensional (3D) data such as 3D Point Clouds (PCs). While most existing studies concerning change detection only focus on 2-Dimensional (2D) images (Shi et al., 2020), we propose to concentrate on 3D data, which is better suited to urban 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 change detection and categorization (Waser et al., 2007; Guerin et al., 2014; Erdogan and Yilmaz, 2019).

Thanks to photogrammetric acquisition or light detection and ranging (LiDAR) acquisition, 3D PC data are becoming more popular. Yet, as reported in a recent survey (de Gélis et al., 2021b), most studies end up relying solely on 2D rasterization of PCs onto a Digital Surface Model (DSM). Indeed, dealing directly with 3D PCs involves more difficulties due to PC characteristics: sparsity, disorder, irregular point distribution. Furthermore, point locations and distribution can vary significantly in unchanged areas making the change detection task even harder. Therefore, rasterizing PCs onto regular grids of pixels of height facilitates the application of traditional 2D image processing approaches. But 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. Aside from 2D DSM rasterization, 3D rasterization into a 3D voxel grid is also a possibility and this facilitates the processing by using, for example, convolutions with 3D kernels. 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 characteristics of 3D environments. Lastly, it should be outlined that

iris.de-gelis@irisa.fr (I. de Gélis); sebastien.lefevre@irisa.fr (S. Lefèvre); thomas.corpetti@cnrs.fr (T. Corpetti)
ORCID(s): 0000-0003-3030-1907 (I. de Gélis); 0000-0002-2384-8202 (S. Lefèvre); 0000-0002-0257-138 (T. Corpetti)

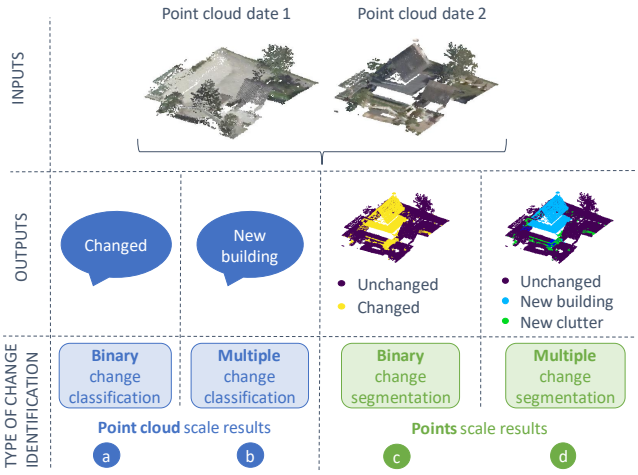


Figure 1: Different types of change detection results, at the scale of PCs (a and b) or points (c and d) and with binary (a and c) or multiple classes (c and d). In this study, we tackle the multiple change issue (d). In addition, for the purpose of comparison with the state-of-the-art, a method is also proposed to address multiple change classification (b).

the computation of 3D meshes or specific features related to PC is tricky with rasterized data, advocating for using raw 3D PCs.

Concerning 3D PCs change detection, binary (change/no-change) or multiple (nature of change) information can be extracted. In this study, change categorization 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 1. In this study, we mainly aim to tackle the multiple change segmentation task (Figure 1d) giving a finer precision of results. However, for the sake of comparison with state-of-art methods, we will also address the multiple change classification task.

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. Therefore, in this paper, the following contributions are proposed:

1. A 3D PCs Siamese Kernel Point convolution (KPConv) network able to deal with multiple change segmentation, the first deep method to provide results at point scale[1]. This network is also adapted for a multiple change classification task in order to allow comparison with the state-of-the-art;
2. Three new datasets for 3D change segmentation and classification, publicly available to foster research in the field and ease reproducibility.

After a presentation of related works in the following section, we introduce in Section 3 both of our proposed networks, named Siamese KPConv and its Cls variant, for 3D point clouds change segmentation and classification, respectively. Then, in Section 4 we propose the three new datasets: the synthetic Urb3DCD-V2 dataset as well as its classification version and Actueel Hoogtebestand Nederland Change Detection (AHN-CD), a change detection version of the real AHN dataset. Section 4 also presents Change3D, a public dataset for change detection at PCs scale. We conduct experiments on these datasets and report the results in Section 5. Section 6 is devoted to discussion, and we show that pre-training on simulated data greatly reduces the cost of manual annotation on real data. Finally, we provide a conclusion and perspectives in Section 7.

2. Related work

In the following section, we review existing works on 3D PCs change detection. We also discuss some representative works on Siamese architectures and deep learning for 3D PCs.

2.1. 3D PCs change detection

Existing traditional methods dealing directly with 3D PC change detection and categorization can be divided into two groups. Post-classification methods firstly perform a semantic segmentation of each PC and then compare obtained labels in order to retrieve changes (Awrangjeb et al., 2015; Roynard et al., 2016; Siddiqui and Awrangjeb, 2017; Xu et al., 2015b; Dai et al., 2020; Voelsen et al., 2021). Conversely, pre-classification methods immediately highlight changes and then classify them (Xu et al., 2015a). Both pre- and post-classification methods embed errors coming from each step, thus leading to errors in the final results (Xu et al., 2015b). To counter this issue, Tran et al. (2018) suggest performing change detection and categorization in a single step by using a Random Forest (RF) algorithm trained on handcrafted features related to point distribution, geometrical attributes, terrain elevation, multi-target capability of LiDAR and a "between date" feature.

While numerous works dealing with feature extraction, object detection, and segmentation in 3D PCs are available, the change detection issue still remains largely unexplored with deep learning (de Gélis et al., 2021b). Indeed, some studies apply Siamese or Feed-Forward with early fusion (EF) networks on 2.5D DSM. The results obtained simply consist of a binary classification at patch level (Zhang et al., 2019). 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 Shape Retrieval Challenge 2021 (SHREC21)

[1] A preliminary version of this method was presented at the ISPRS Congress 2021 (de Gélis et al., 2021a).

on Change3D dataset. 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). Thus, the expected result is provided at the PCs scale corresponding to the multiple change classification task (Figure 1b).

2.2. Siamese architectures

Firstly developed in computer vision (Chopra et al., 2005; Zagoruyko and Komodakis, 2015), Siamese networks belong to the double-stream architecture family. 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., 2020; Xu et al., 2020). The latter lead to more flexibility 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 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 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.

2.3. Deep learning for 3D PCs

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

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, as will be also considered in this study.

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 learning 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). 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 Aerial LiDAR Survey (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,b), 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.

In our study, we aim to tackle the 3D PC change detection task through inspiring from previous success in deep learning for change detection, that remains limited to the 2D case. With this objective in mind, relying on point convolutions is a relevant choice, and we now focus our review of related work on point convolutions. Let us emphasize that relying on point convolutions allows us to easily adapt popular architectures, such as the encoder/decoder widely-used for numerous tasks, including semantic segmentation or change detection, the latter being successfully addressed with Siamese networks (see Section 2.2).

Point convolution is defined for a point $x \in \mathbb{R}^3$ as:

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{B}^3_{R,x_i}} g(x_i - x) f_i \tag{1}$$

where x_i is a point in $\mathcal{P} \in \mathbb{R}^{N\times 3}$, f_i its corresponding features in $\mathcal{F} \in \mathbb{R}^{N\times D}$, with D the number of input features, $\mathcal{B}^3_{R,x_i} = \left\{x \in \mathbb{R}^3 s.t. \|x - x_i\| \leq R\right\}$ is the neighborhood of size $R \in \mathbb{R}$ and g is the kernel function.

A crucial issue remains in the definition of the kernel function g. g returns weight matrix in $\mathbb{R}^O$, with O the number of output features. A first category is based on MLP (Wang et al., 2018; Li et al., 2018; Hermosilla et al., 2018; Boulch,

2020) while another family relies on geometric kernels. Convolutions involving MLP are more complex, require more trainable parameters and exhibit limited performances. Geometric kernels can be defined with linear functions on k-Nearest Neighbors (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. Based on these overall performances and its intuitive principles, we chose to rely on KPConv to deal with PCs.

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 $\left\{W_k \mid k \leq K\right\} \subset \mathbb{R}^{D_{in} \times D_{out}}$ 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 $x \in \mathcal{B}^3_{R,x_i}$:

$$g(y) = \sum_{k \leq K} h(y, \widetilde{x}_k) W_k \tag{2}$$

where h is the correlation function between $\widetilde{x}_k$ and y, as defined by equation (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:

$$h(y, \widetilde{x}_k) = \max\left(0, 1 - \frac{\|y - \widetilde{x}_k\|}{\sigma}\right) \tag{3}$$

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 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).

Let us now introduce the Siamese network based on KPConv proposed in this paper.

3. 3D point cloud change detection

The following section describes the proposed methods for change detection between bi-temporal 3D PCs whether at PC or points scale (see Figure 1). 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 2.3, 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 number of points from each of the input PCs.

3.1. 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 2. 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 2.3. 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 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 2). 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 nearest spatial 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:

$$(\mathcal{P}_1, \mathcal{F}_1) \ominus (\mathcal{P}_2, \mathcal{F}_2) = f_{2i} - f_{1j|j=\arg\min(\|x_{2i}-x_{1j}\|)} \quad (4)$$

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 Red-Green-Blue (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 RGB, intensity, etc.) as early done by the authors of PointNet (Qi et al., 2017a). These supplementary features can be easily added as inputs by modifying the input dimension of weights matrix of kernel points of the first layer.

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 5. Usually, pseudo-Siamese networks are used when data to be compared come with different characteristics.

3.2. 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 1b), henceforth referred as Siamese KPConv Cls. The architecture is presented in Figure 3. It is composed of the same encoder part as in Siamese KPConv

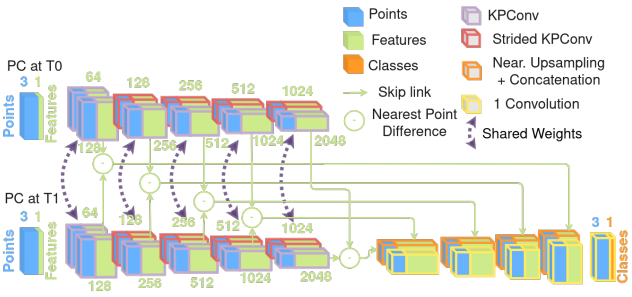


Figure 2: Our Siamese KPConv network architecture. The Pseudo-Siamese version of the network is the same without shared weights symbolized by dotted purple arrows.

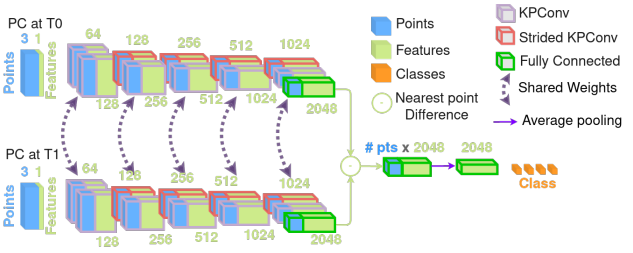


Figure 3: Our Siamese KPConv Cls network architecture for classification.

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, before these feature differences are given as input to 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. Furthermore, the position of the average pooling has been also assessed by testing to perform an average pooling after each encoder and then perform a simple feature difference, but this leads to worse results.

In the next section we present the datasets used to conduct our experiment.

4. 3D point clouds change detection datasets

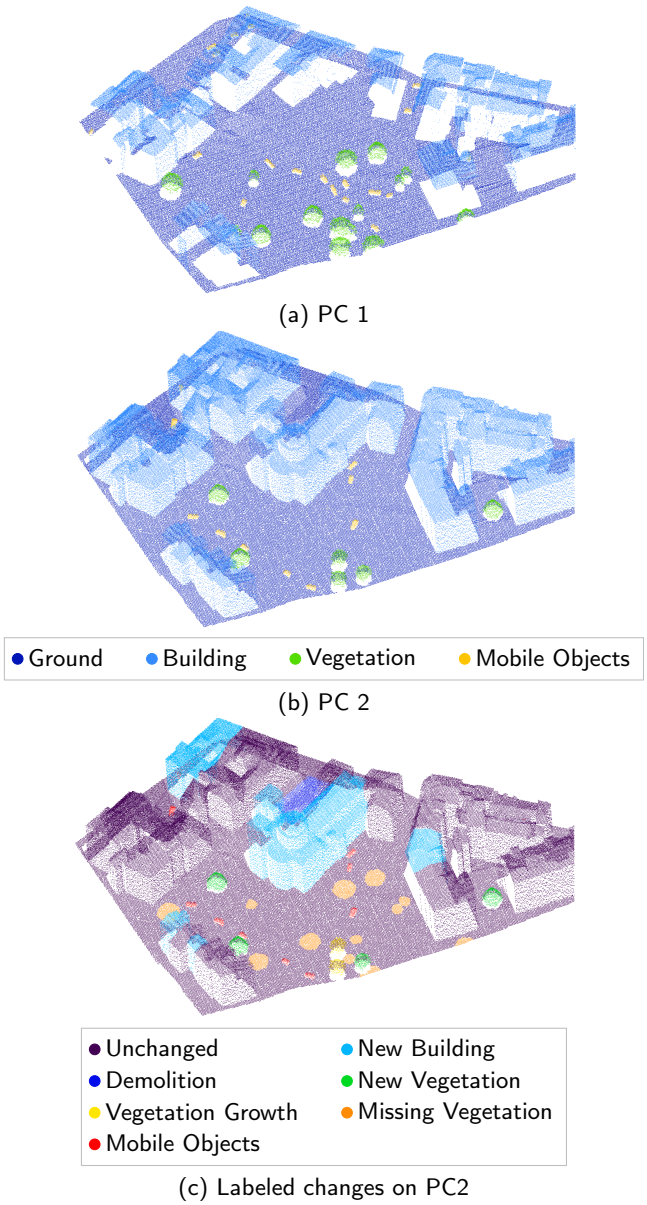
To the best of our knowledge, there is no public dataset with bi-temporal PCs annotated as a function of the change that occurred between the two dates at the point level, except for our recent Urb3DCD proposal (de Gélis et al., 2021b). This dataset contains only 3 classes (related to ground and buildings) where 2 different kinds of changes can be simulated. In order to assess our method in tougher and more realistic conditions, we propose here a new version of this dataset with up to 7 classes by adding vegetation and mobile objects. The description of the enhanced simulator and of this new Urb3DCD-V2 is given in sub-section 4.1. The latter

also introduces Urb3DCD-V2-Cls, a simulated dataset for multiple change classification. Then, to analyze the network's performances in real conditions, we also adapted the Actueel Hoogtebestand Nederland (AHN) dataset to change detection and categorization, leading to the new AHN-CD dataset as described in sub-section 4.2. Finally, let us note that in order to compare our method with other state-of-art methods in change classification at PCs scale, we also use the public Change3D dataset (Ku et al., 2021) that will be briefly presented in sub-section 4.3.

4.1. Simulated urban change detection dataset

To evaluate the performances of our method in tougher conditions than in de Gélis et al. (2021a), we have improved our simulator of urban 3D PCs (de Gélis et al., 2021b) 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 time steps. Moreover, some mobile objects (cars and trucks) have also been added in streets. The size of mobile objects and in particular their length, are also set randomly, within a realistic range. All mobile objects are randomly placed in the 3D model so that there is no collision between 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. We illustrate a pair of simulated bi-temporal PCs in Figure 4. Based on this simulator, we generated several datasets.

Let us recall that the simulator allows us to choose the configuration of LiDAR. A 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 de Gélis et al. (2021b), the most challenging data configuration for change detection methods is the multi-sensor (MS) 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 4.1. Figure 9(a-b) and 11(a-b) give examples of parts of input PCs over the same area. In Figure 11(a-b), the difference of quality between the two input PCs is clearly visible. Training, validation and test areas are similar to the dataset presented in de Gélis et al. (2021b). Similarly to the previous version of the dataset, we run 10, 1 and 3 simulations over each training, validation and test areas respectively, in order to constitute a large enough dataset 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 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).



(a) PC 1

(b) PC 2

(c) Labeled changes on PC2

Figure 4: Sample PCs at two timestamps (a,b) with the corresponding 7 types of changes simulated in (c).

Another particularity of this simulated dataset 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 10(a-b) where building facades without point (hidden facades) are not at the same location in the two acquisitions.

Parameters	Sub-datasets			
	LiDAR low dens. Urb3DCD-V2-1	MS Urb3DCD-V2-2		Urb3DCD-V2-Cls
	Both PCs	PC1	PC2	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
Acquisition configurations for the three sub-datasets of Urb3DCD-V2. Dens. stands for density.

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

Table 2
Class distribution for the Urb3DCD-Cls training, validation and testing splits. For each class, the number of samples along with the class proportion (in %) is given.

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

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 (Figure 1b). To build this dataset, pairs of cylinders of 15 m in radius have been extracted in simulated acquisitions over the train, validation and testing areas presented in de Gélis et al. (2021b). The configuration of acquisition of the PCs are given in Table 4.1 in Urb3DCD-V2-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 vegetation removed. The class distribution of this dataset is given in Table 2.

Notice that Urb3DCD-V2 as well as its classification variant are available at the following link: `https://ieee-dataport.org/open-access/urb3dcd-urban-point-clouds-simulated-dataset-3d-change-detection`.

4.2. Change detection dataset from real ALS data

The Netherlands was the first country to have full coverage of their country by ALS data (Sande et al., 2010). Since 2003, a total of four surveys have been delivered, making change detection possible. In addition to 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.

We consider some bi-temporal data coming from AHN3 and AHN4 in order to establish a change annotation according to the given classification. We call this new dataset AHN-CD. Relying on the AHN classification, 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 the class "bridge" is fused with the class "clutter" since it contains only a few points. Similarly, over selected areas, there are almost no points concerning water and remaining water points are thus deleted.

As already mentioned, when dealing with 3D PCs, comparison of point labels is not obvious since there is no direct corresponding pair of points between PCs. One could have taken the nearest point in PC from AHN3 for comparison with labels of points in AHN4, but this yields very noisy results. As a consequence, a more complex processing chain has been chosen, illustrated in Figure 5. In particular, to obtain smoother annotation, 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 required to distinguish between real demolition and building shadows.

It will be shown in Section 6.1 that this annotation is unfortunately still not perfect.

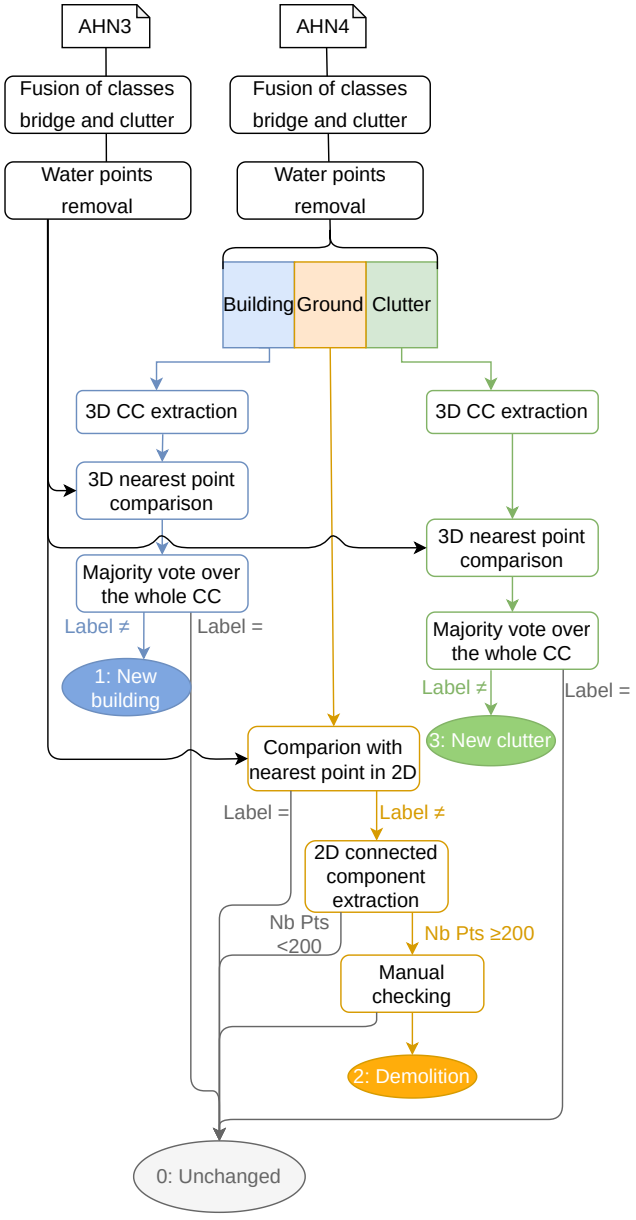


Figure 5: Flowchart for change detection annotation of AHN pairs (a.k.a. AHN-CD) into four classes: unchanged, new building, demolition and new clutter. CC stands for connected component.

The density of AHN3 varies from 10 to 14 points/m^2 while AHN4 varies from 20 to 24 points/m^2. Height and planimetric stochastic errors are 5 cm. In addition to point coordinates, AHN data also includes RGB colors, LiDAR intensity and the number of returns. Since we focus here on raw 3D PCs only, we rely solely on the 3D point coordinates to feed the network. As previously mentioned, AHN provides full coverage of the Netherlands. So we select some tiles to define our training, validation and test sets. Selected areas are shown in Figure 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/`. 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.

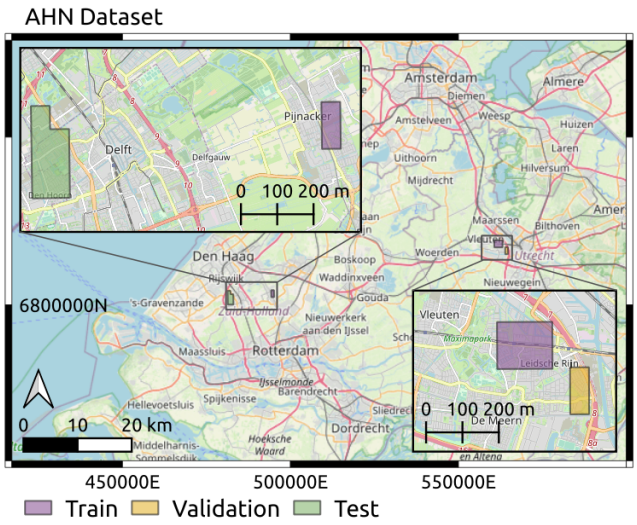


Figure 6: Selected parts of AHN-CD dataset for training, validation and testing.

4.3. Change3D dataset

Change3D is a dataset provided by CycloMedia Technology for the 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 interests, are identified by their coordinates and an associated label (see Figure 7 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 split is given in Table 3. As it 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.

As 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. As far as our study is concerned, we decided to extract vertical cylinders of 3 m in radius, as done by the SiamGCN deep learning method.

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

Table 3
Class distribution for the Change3D training and testing splits. For each class, the number of samples along with the class proportion (in %) is given.

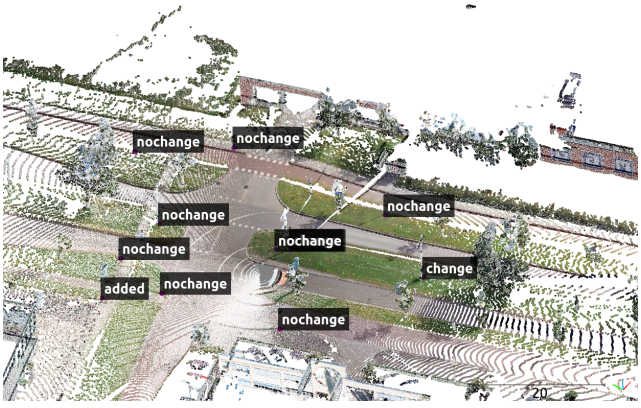


Figure 7: Example of a scene from the Change3D dataset with the points of interests and their corresponding labels.

5. Experimental results

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

5.1. 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 Random Forest (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 dataset (de Gélis et al., 2021b). We re-implemented feature extraction of all features of Tran et al. (2018) except those using LiDAR's multi-target capability because Urb3DCD does 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 by 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.

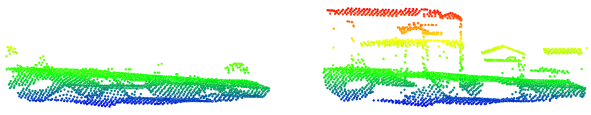
Concerning change classification, we propose to compare our Siamese KPConv Cls with SiamGCN network (Ku et al., 2021). This siamese network relies on graph convolution, in particular edge convolution operator (EdgeConv) (Wang et al., 2019b). From input PCs, graphs are constructed from the kNN connections. 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 subtraction 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 are then limited to the Change3D dataset.

As for quantitative parameters, for each class of change, the Intersection over Union (IoU) is reported. 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 accuracy (mAcc) and the mean of IoU over classes of changes (mIoU_{ch}) for reliable quantitative assessment of the different methods.

5.2. 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 8. Let us consider a point in the center of the building's roof (center of Figure 8b). 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 valid the registration, the same rotation angle is applied both PCs in the pair.



(a) First cylinder (b) Second cylinder

Figure 8: 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.

As mentioned in Section 3.1, 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. 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.

Parameter settings (summarized in Table 4) have been largely influenced by the original KPConv proposed values. We thus use a Stochastic Gradient Descent (SGD) with a momentum of 0.98, to minimize a point-wise Negative Log Likelihood (NLL) loss, given by the following equation:

$$NLL(y_t, y_p) = -(y_t \log(y_p) + (1 - y_t) \log(1 - y_p)) \quad (5)$$

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. Regarding Siamese KPConv Cls, the same training parameters 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.

Unlike KP-FCNN, we included a probability dropout of 0.5 in the last classification layers. Conversely, no dropout is used for Siamese KPConv Cls. In addition, in order to prevent over-fitting, 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, 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. 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).[2] Concerning the nearest point feature difference (Equation 4), the nearest point is

[2]The implementation will be made available upon acceptance of this paper.

	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 (Ku et al., 2021)	Adam	10^{-3}	Step	No	NLL	16	EdgeConv
DSM-based deep learning	Adam	10^{-3}	Exponential	No	NLL	32	2D convolution

Table 4
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.

	Deep learning 3D				Deep learning 2D	RF
		Siamese KPConv (Cls)				
	Input cylinders radius (m)	dl_0 (m)	Input samples Train	Val	DSM resolution (m)	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 5
Summary of input parameters according to the five datasets and the three families of methods.

determined thanks to the kNN implementation available in PyTorch Geometric, which is Graphic Process Unit (GPU) compliant for faster computation.

For the RF comparison, let us remark that some features are 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 5.

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 4. Let us note that the difference between our results and the original SiamGCN paper Ku et al. (2021) come from the methodology used for train/validation/test data splitting. In our experiments, we have followed the standard approach in machine learning, i.e. put apart the test 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.

5.3. Results

5.3.1. Semantic change on synthetic datasets

Quantitative results concerning the Urb3DCD-V2 dataset for low density LiDAR are presented in Table 6 and 8 and qualitative results are shown in Figures 9 and 10. 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 6, we can notice that the direct processing of 3D PC 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 over-fitting. 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. And this leads to high variation observed within the results.

Concerning per-class performance in Table 8, 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 9f. 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

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 6
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.

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 (9f-10f) are consistent with the ground truth (9c-10c). Conversely, the RF method (9d-10d) 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 9d (see the ellipse showing the region of interest). In Figure 10, 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 10(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 10d). As our method learns deep features from raw 3D PCs, it seems to be able to understand 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 reprojection 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 10e), 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.

Quantitative results for the MS dataset are presented in Tables 7 and 9. 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 from de Gélis et al. (2021b), 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 over-fitting 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., 2020; Zhou et al., 2021).

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 6 and 7, 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 9. When looking at qualitative results in Figures 11 and 12, the missing vegetation class is almost always mixed with demolition in RF results (11d-12d). Changed objects boundaries are not precise in DSM-FC-EF results (11e-12e) due to the rasterization process. Despite the difference of quality between the two input PCs (11(a-b)-12(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 12, we can draw the same conclusion as already made on the Urb3CDCD-V2 low density dataset.

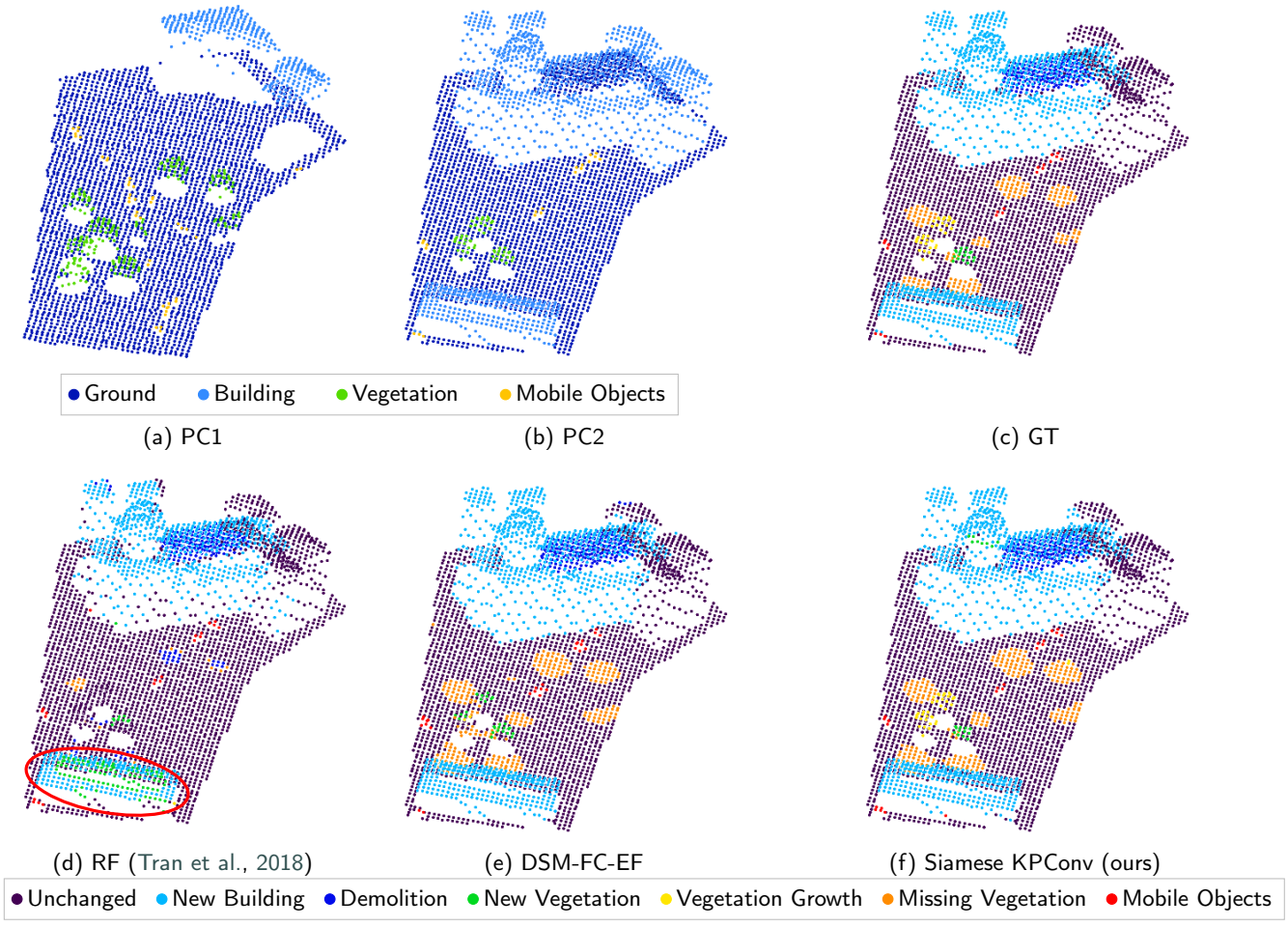


Figure 9: Visual change detection results on Urb3DCD-V2 low density LiDAR sub-dataset: (a-b) the two input point clouds; (c) Ground truth: simulated changes; (d) RF (Tran et al., 2018) results; (e) DSM-FC-EF (adaptation of Daudt et al. (2018) FC-EF to DSM inspired by Zhang et al. (2019) works) results; (f) our results with Siamese KPConv. Region of interest specifically discussed in the text is highlighted with an ellipse.

Method	mAcc	$mIoU_{ch}$
Siamese KPConv (ours)	73.24 ± 5.7	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 7
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.

5.3.2. Semantic change on real dataset

Results on AHN-CD dataset are presented in Table 10. 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 FC network with early fusion performs better than Siamese networks on DSMs, with lower scores, however, than our method. As can be seen in Figure 13, Pseudo-Siamese KPConv predictions (13d) are globally similar to the ground truth (13c). Finally, scores of all methods are lower compared to results on Urb3DCD-V2 datasets, and this will be further discussed in Section 6.1.

As far as computation time is concerned, 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 test area of Figure 6. The test set corresponds to ∼ 27,000 cylinders extracted from the pair of original PCs, i.e. a total of around 34 and 81 millions 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.

5.3.3. Change classification results

Results regarding the change classification task for both synthetic and real datasets are presented in Table 11 and 12, respectively. Our architecture is reaching some quite reliable

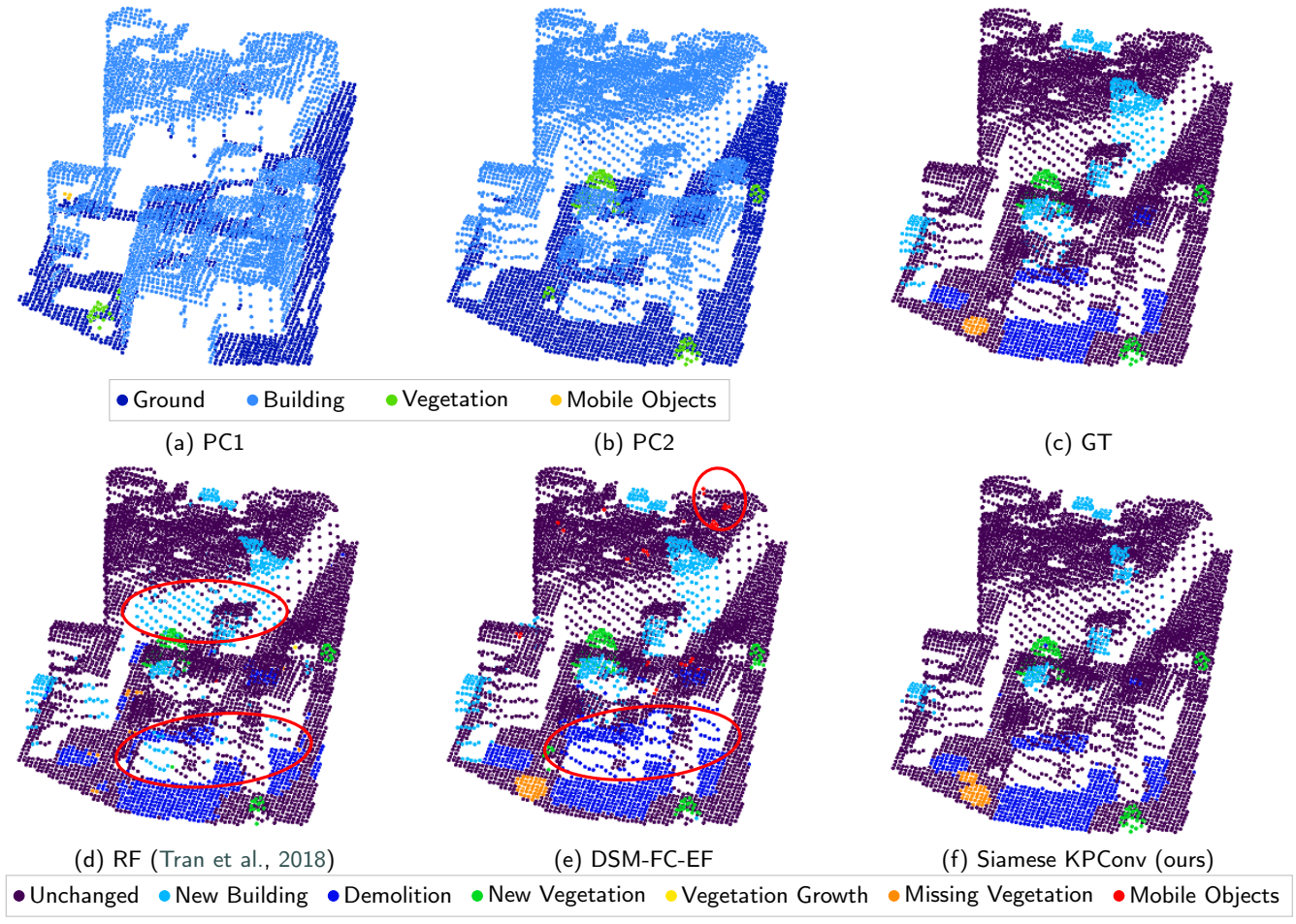


Figure 10: Visual change detection results on Urb3DCD-V2 low density LiDAR sub-dataset in an area containing occlusions: (a-b) the two input point clouds; (c) Ground truth: simulated changes; (d) RF (Tran et al., 2018) results; (e) DSM-FC-EF (adaptation of Daudt et al. (2018) FC-EF to DSM inspired by 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	Unchanged	New building	Demolition	New veg.	Veg. growth	Missing veg.	Mobile Object
Siamese KPConv (ours)	**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 (ours)	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 (Tran et al., 2018)	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 8
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	Unchanged	New building	Demolition	New veg.	Veg. growth	Missing veg.	Mobile Object
Siamese KPConv (ours)	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 (ours)	**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 (Tran et al., 2018)	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 9
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.

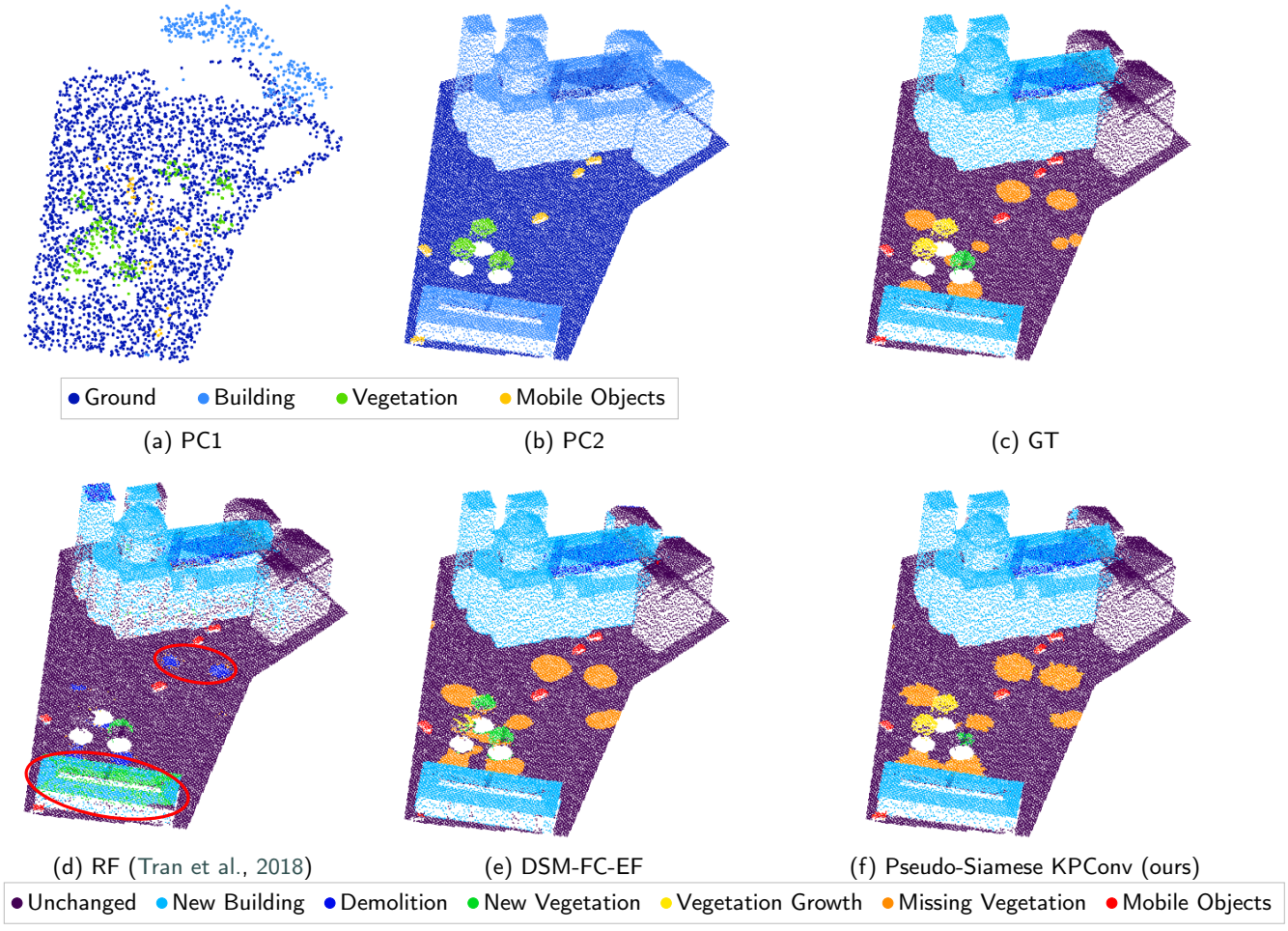


Figure 11: Visual change detection results on Urb3DCD-V2 MS sub-dataset: (a-b) the two input point clouds; (c) Ground truth: simulated changes; (d) RF (Tran et al., 2018) results; (e) DSM-FC-EF (adaptation of Daudt et al. (2018) FC-EF to DSM inspired by Zhang et al. (2019) works) results; (f) our results with Pseudo-Siamese KPConv. Regions of interest specifically discussed in the text are highlighted with ellipses.

Method	mAcc	mIoU_{ch}	Per class IoU			
			Unchanged	New building	Demolition	New clutter
Siamese KPConv (ours)	81.86 ± 0.72	**59.93** ± 0.14	**95.94** ± 0.06	**83.19** ± 1.54	**56.05** ± 1.74	**40.53** ± 0.56
Pseudo-Siamese KPConv (ours)	**84.44** ± 1.24	52.32 ± 4.31	92.96 ± 1.34	76.54 ± 11.39	43.67 ± 1.88	36.76 ± 2.95
DSM-Siamese	62.85 ± 1.13	33.18 ± 3.56	88.58 ± 2.53	60.95 ± 5.54	18.04 ± 1.59	20.54 ± 3.59
DSM-Pseudo-Siamese	67.04 ± 0.77	41.40 ± 0.62	92.25 ± 0.11	73.26 ± 0.68	22.91 ± 1.82	28.02 ± 0.73
DSM-FC-EF	74.98 ± 0.80	44.73 ± 2.16	92.95 ± 1.49	74.21 ± 0.37	33.68 ± 6.84	26.32 ± 0.04
RF (Tran et al., 2018)	50.11 ± 0.01	28.56 ± 0.02	93.13 ± 0.00	70.5 ± 0.21	2.04 ± 0.04	13.27 ± 0.02

Table 10
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.

results for each class of the Urb3DCD-Cls dataset. It also strongly outperforms SiamGCN. When looking at Table 12 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 3 the "color change" class is under-represented (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 class 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

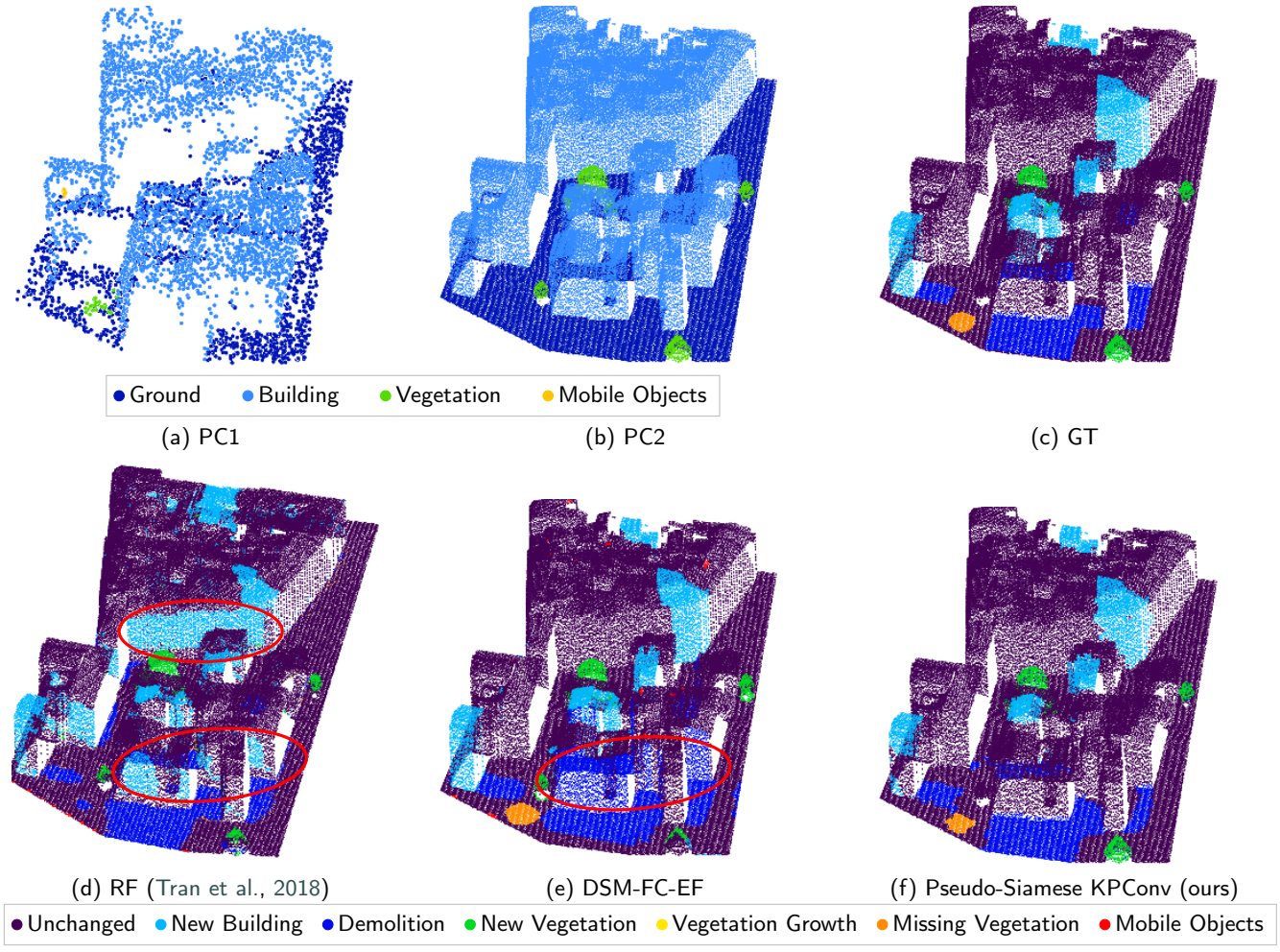


Figure 12: Visual change detection results on Urb3DCD-V2 MS sub-dataset in an area containing occlusions: (a-b) the two input point clouds; (c) Ground truth: simulated changes; (d) RF (Tran et al., 2018) results; (e) DSM-FC-EF (adaptation of Daudt et al. (2018) FC-EF to DSM inspired by Zhang et al. (2019) works) results; (f) our results with Pseudo-Siamese KPConv. Regions of interest specifically discussed in the text are highlighted with ellipses.

Method	mAcc	mIoU	Per class IoU				
			No change	New building	Demolition	New veg.	Veg. removed
Siamese KPconv Cls (ours)	**88.75** ± 1.59	**80.30** ± 1.58	**82.10** ± 0.98	**73.65** ± 1.56	**80.50** ± 1.60	**85.81** ± 1.64	**79.45** ± 2.87
SiamGCN (Ku et al., 2021)	76.45 ± 1.14	57.27 ± 0.52	68.63 ± 0.97	61.43 ± 0.79	70.29 ± 1.08	38.31 ± 0.59	47.69 ± 0.92

Table 11
Change classification results on Urb3DCD-Cls synthetic dataset. Results are given in %. Veg. stands for vegetation.

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.

6. Discussion

In the following section, we focus on the quality of AHN-CD and we discuss the transfer learning capacity of the network.

6.1. AHN-CD quality assessment

As seen in Table 10, 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 13. In our opinion, the main difficulty comes from change annotation. First of all, in order to obtain our annotations, we performed an automatic comparison of the two PCs, leading to a lot of mis-classifications, 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 13.

Method	mAcc	mIoU	Per class IoU				
			No change	New building	Demolition	New veg.	Veg. removed
Siamese KPconv Cls (ours)	**49.64** ± 1.35	**34.64** ± 1.18	55.35 ± 2.80	**43.41** ± 3.71	**47.93** ± 4.74	**19.85** ± 9.25	6.67 ± 11.55
PoChaDeHH (Ku et al., 2021)	45.18	30.22	**61.06**	31.58	40.00	4.17	**14.29**
HGI-CD (Ku et al., 2021)	25.82	17.17	55.30	16.28	14.29	0.00	0.00
SiamGCN (Ku et al., 2021)	32.04 ± 6.49	19.18 ± 1.03	42.56 ± 1.78	24.33 ± 0.83	11.27 ± 3.07	14.00 ± 2.19	3.70 ± 4.94

Table 12
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 %.

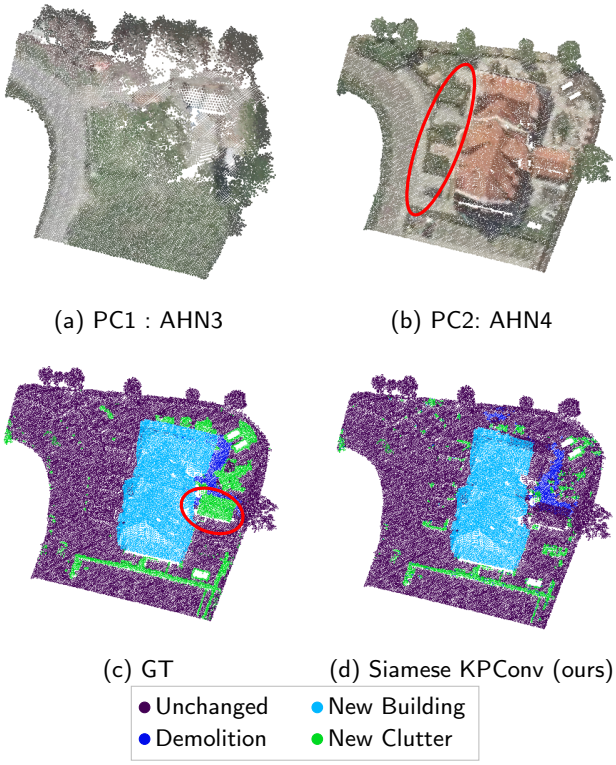


Figure 13: Qualitative results on AHN-CD dataset. See the discussion regarding the quality of the GT. Regions of interest specifically discussed in the text is highlighted with ellipses.

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 (see region of interest in Figure 13a), yielding difficulties in practice. Another example is given in Figure 14 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. As can be seen, our method correctly predicted the majority of all new buildings. 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 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 13, the shed is marked as clutter in the annotation whereas it is sometimes predicted as new building or even unchanged because of the glasshouse present in the older PC as explained before (Figure 14). Also, notice 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 remark should be made on the demolition class. Indeed, this class is largely under-represented: 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 14d (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 on the left side of Figure 14.

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 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 14, 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).

Furthermore, a sub-part of the AHN-CD test 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: 60.95% 'unchanged', 29.06% 'new building', 7.04% 'demolition' and 2.95% 'new clutter'. The selected area is about 12,400 m^2. Results are given in Table 13. Again,

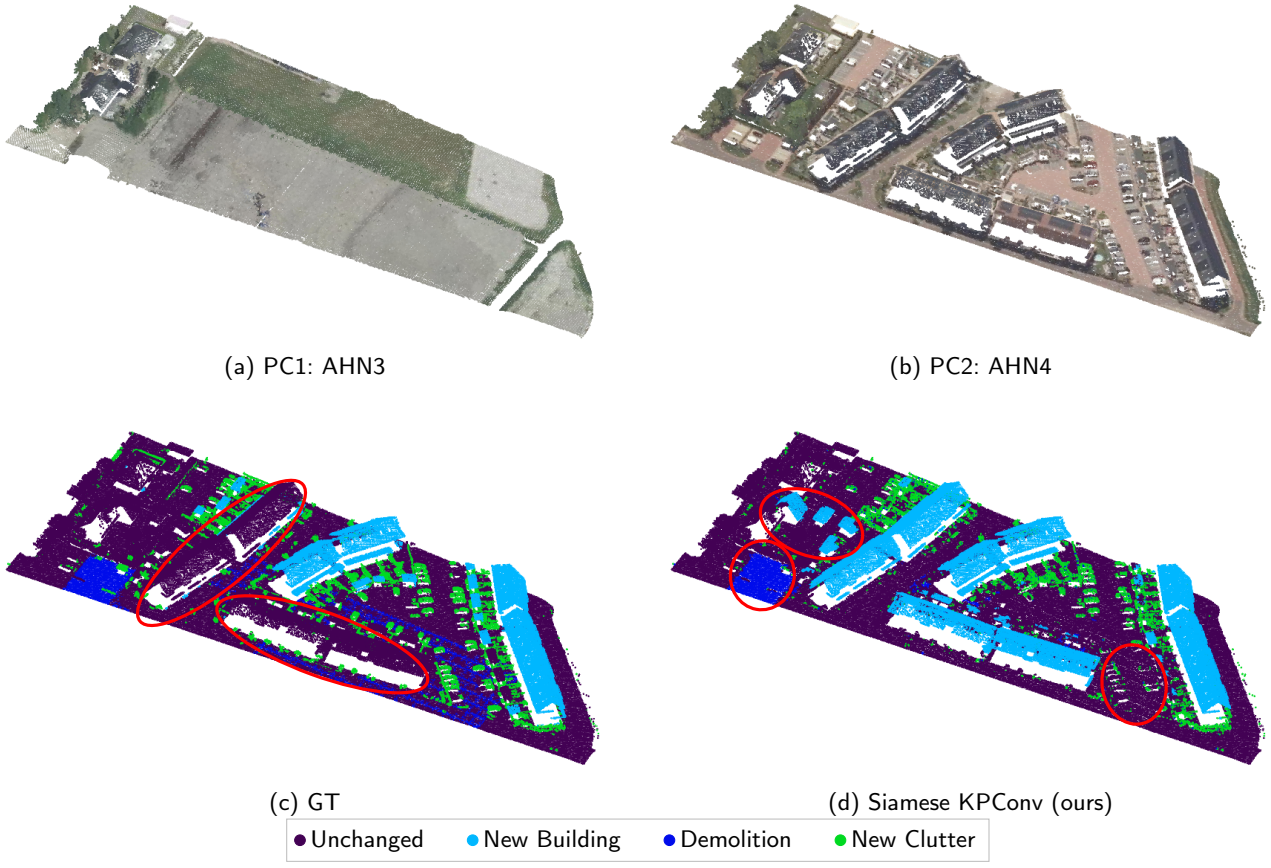


Figure 14: Qualitative results on AHN-CD dataset, illustrating some ground truth errors contrasting with relevant prediction by our method. Regions of interest specifically discussed in the text are highlighted with ellipses.

Method	mAcc	$mIoU_{ch}$	Per class IoU			
			Unchanged	New building	Demolition	New clutter
Siamese KPConv (ours)	85.65 ± 1.55	**72.95** ± 2.05	**89.75** ± 2.18	82.77 ± 5.38	**86.44** ± 0.88	**46.65** ± 0.16
Pseudo-Siamese KPConv (ours)	**87.87** ± 1.89	69.33 ± 1.99	88.90 ± 1.89	**86.93** ± 5.32	84.01 ± 0.87	37.08 ± 2.85
DSM-Siamese	50.87 ± 1.15	30.96 ± 2.48	77.10 ± 1.51	76.77 ± 0.79	4.91 ± 8.33	11.20 ± 1.71
DSM-Pseudo-Siamese	70.71 ± 5.09	48.85 ± 7.03	78.00 ± 5.09	75.32 ± 8.59	47.46 ± 11.92	23.76 ± 0.56
DSM-FC-EF	71.47 ± 1.43	45.57 ± 0.98	70.77 ± 1.13	90.32 ± 0.61	30.58 ± 1.76	15.81 ± 0.81
RF (Tran et al., 2018)	47.94 ± 0.02	29.45 ± 0.02	78.24 ± 0.00	74.64 ± 0.03	0.00 ± 0.00	13.72 ± 0.06

Table 13
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.

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, 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.

6.2. Transfer learning

In the following section, we aim at assessing the transfer capacity of our method compared to others, 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 Table 14, we report the transfer results between a training on Urb3DCD-V2 MS sub-dataset and a test on the

Method	$mIoU_{ch}$	Per class IoU						
		Unchanged	New build.	Demol.	New veg.	Veg. growth	Missing veg.	M.O.
Pseudo-Siamese KPConv (ours)	**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-Pseudo-Siamese	35.77	91.55	69.36	56.02	36.3	4.76	30.11	17.94
DSM-FC-EF	42.01	92.87	67.11	55.63	33.41	1.14	39.1	29.72
RF (Tran et al., 2018)	14.48	87.74	54.03	21.91	8.24	0.47	0.02	2.19

Table 14
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 %. Build., demol., veg. and M.O. stand for building, demolition, vegetation and mobile object respectively.

low-density LiDAR sub-dataset. Notice that no retraining has been done to adapt to the other dataset. As expected, results are worse than when the training is performed on a training set containing the same types of PCs as the test 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 7) 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 8). Table 14 recalls the poor generalization capacity of the RF method, even though it requires a smaller training set than deep learning methods (de Gélis et al., 2021b).

The issue of the size of the training set is crucial since automatic data annotation is tricky (see Section 6.1) 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 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 set (symbolized by the number of cylinders given as input) and compared results with the network pre-trained on a simulated dataset and fine-tuned on real data. The results are depicted in Figure 15. 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 Section 5.3, where, for each training epoch, 6,000 cylinders are chosen randomly in the training set according to class balance (see Section 5.2). 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 and 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=1m). 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 360,000 cylinders in total (considering 60 epochs) and that was followed in Section 5. 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.

7. Conclusion

In this study 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 the first, to our knowledge, 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 real and synthetic datasets. The latter were generated thanks to a simulator also introduced in this paper, that extends de Gélis et al. (2021b) to 6 different classes of change concerning buildings, vegetation and mobile objects. This enabled us to create different datasets with various quality gathered in the Urb3DCD-V2 product.

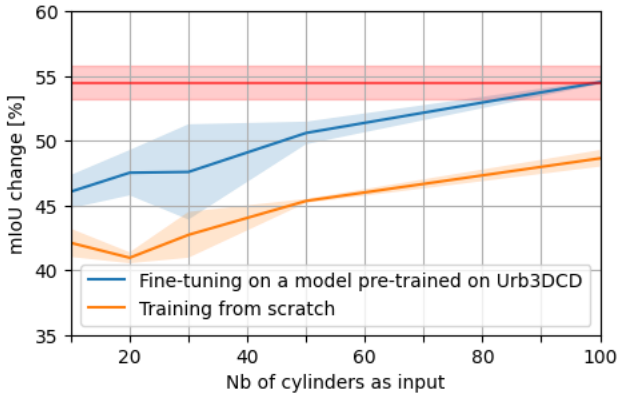


Figure 15: Comparison between training from scratch and using pre-trained weights learnt on a simulated dataset of Siamese KPConv. The mean of IoU over classes of change is given as a function of the number of cylinders of 50 m in diameter given as input. In red, the best results obtained with Siamese KPConv trained from scratch over 6,000 cylinders with random drawing.

In addition, tests were carried out on AHN-CD, a real dataset we built from AHN products, a series of national surveys on the Netherlands. 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 the traditional machine learning algorithm trained on handcrafted features, as reported in de Gélis et al. (2021b), 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 (FC) 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. Furthermore, an adapted version of Siamese KPConv for the change classification task outperforms state-of-the-art methods including deep learning based networks on both synthetic and real (Change3D) datasets.

In a last part, we assessed the transfer learning capacity of the network when trained on different conditions of acquisition than those faced in the test set. When directly transferring without retraining from the multi-sensor dataset to the LiDAR with low density, obtained results are higher than the traditional machine learning results without transfer, i.e. trained on the target data. 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.

To the best of our knowledge, our work is the first to deal with deep learning for multiple change segmentation over 3D PCs. While our method shows some promising results on both synthetic and real datasets, and good generalization capabilities, it remains dependent on the amount and quality of the labels in the training set. Thus, in future work, we plan to deal with the challenging annotation issue by exploring semi-supervised or even unsupervised learning approaches through self-supervision.

Acknowledgements

This research was funded by Magellium, Toulouse and the CNES, Toulouse. This work was granted access to the HPC resources of IDRIS under the allocation 2021-AD011011754R1 made by GENCI.

References

Atzmon, M., Maron, H., Lipman, Y., 2018. Point convolutional neural networks by extension operators. arXiv preprint arXiv:1803.10091 .

Awrangjeb, M., Fraser, C.S., Lu, G., 2015. Building change detection from lidar point cloud data based on connected component analysis. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences 2.

Boulch, A., 2020. Convpoint: Continuous convolutions for point cloud processing. Computers & Graphics 88, 24–34.

Boulch, A., Guerry, J., Le Saux, B., Audebert, N., 2018. Snapnet: 3d point cloud semantic labeling with 2d deep segmentation networks. Computers & Graphics 71, 189–198.

Champion, N., Boldo, D., Pierrot-Deseilligny, M., Stamon, G., 2010. 2d building change detection from high resolution satelliteimagery: A two-step hierarchical method based on 3d invariant primitives. Pattern Recognition Letters 31, 1138–1147.

Chaton, T., Chaulet, N., Horache, S., Landrieu, L., 2020. Torch-points3d: A modular multi-task framework for reproducible deep learning on 3d point clouds, in: 2020 International Conference on 3D Vision (3DV), IEEE. pp. 1–10.

Chopra, S., Hadsell, R., LeCun, Y., 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), IEEE. pp. 539–546.

Dai, C., Zhang, Z., Lin, D., 2020. An object-based bidirectional method for integrated building extraction and change detection between multimodal point clouds. Remote Sensing 12, 1680.

Daudt, R.C., Le Saux, B., Boulch, A., 2018. Fully convolutional siamese networks for change detection, in: 2018 25th IEEE International Conference on Image Processing (ICIP), IEEE. pp. 4063–4067.

Dechesne, C., Lassalle, P., Lefèvre, S., 2021. Bayesian u-net: Estimating uncertainty in semantic segmentation of earth observation images. Remote Sensing 13, 3836.

Diestel, W., 2003. Arbaro—tree generation for povray.

Dong, H., Ma, W., Wu, Y., Gong, M., Jiao, L., 2018. Local descriptor learning for change detection in synthetic aperture radar images via convolutional neural networks. IEEE Access 7, 15389–15403.

Erdogan, M., Yilmaz, A., 2019. Detection of building damage caused by van earthquake using image and digital surface model (dsm) difference. International Journal of Remote Sensing 40, 3772–3786.

Feranec, J., Hazeu, G., Christensen, S., Jaffrain, G., 2007. Corine land cover change detection in europe (case studies of the netherlands and slovakia). Land use policy 24, 234–247.

de Gélis, I., Lefèvre, S., Corpetti, T., 2021a. 3d urban change detection with point cloud siamese networks. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 43, 879–886.

de Gélis, I., Lefèvre, S., Corpetti, T., 2021b. Change detection in urban point clouds: An experimental comparison with simulated 3d datasets. Remote Sensing 13, 2629.

Groh, F., Wieschollek, P., Lensch, H.P., 2018. Flex-convolution, in: Asian Conference on Computer Vision, Springer. pp. 105–122.

Guerin, C., Binet, R., Pierrot-Deseilligny, M., 2014. Automatic detection of elevation changes by differential dsm analysis: Application to urban areas. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 7, 4020–4037.

Guiotte, F., Pham, M.T., Dambreville, R., Corpetti, T., Lefèvre, S., 2020. Semantic segmentation of lidar points clouds: rasterization beyond digital elevation models. IEEE Geoscience and Remote Sensing Letters 17, 2016–2019.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M., 2020. Deep learning for 3d point clouds: A survey. IEEE transactions on pattern analysis and machine intelligence 43, 4338–4364.

Hedjam, R., Abdesselam, A., Melgani, F., 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, P., Ritschel, T., Vázquez, P.P., Vinacua, À., Ropinski, T., 2018. Monte carlo convolution for learning on non-uniformly sampled point clouds. ACM Transactions on Graphics (TOG) 37, 1–12.

Hua, B.S., Tran, M.K., Yeung, S.K., 2018. Pointwise convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 984–993.

Jiang, H., Hu, X., Li, K., Zhang, J., Gong, J., Zhang, M., 2020. Pga-siamnet: Pyramid feature-based attention-guided siamese network for remote sensing orthoimagery building change detection. Remote Sensing 12, 484.

Ku, T., Galanakis, S., Boom, B., Veltkamp, R.C., Bangera, D., Gangisetty, S., Stagakis, N., Arvanitis, G., Moustakas, K., 2021. Shrec 2021: 3d point cloud change detection for street scenes. Computers & Graphics 99, 192–200. URL: https://www.sciencedirect.com/science/article/pii/S0097849321001369, doi:https://doi.org/10.1016/j.cag.2021.07.004.

Landrieu, L., Simonovsky, M., 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, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O., 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.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B., 2018. Pointcnn: Convolution on x-transformed points. Advances in neural information processing systems 31, 820–830.

Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G., Johnson, B.A., 2019. Deep learning in remote sensing applications: A meta-analysis and review. ISPRS journal of photogrammetry and remote sensing 152, 166–177.

Qi, C.R., Su, H., Mo, K., Guibas, L.J., 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, C.R., Yi, L., Su, H., Guibas, L.J., 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems 30.

Qin, R., Tian, J., Reinartz, P., 2016. 3d change detection–approaches and applications. ISPRS Journal of Photogrammetry and Remote Sensing 122, 41–56.

Rethage, D., Wald, J., Sturm, J., Navab, N., Tombari, F., 2018. Fully-convolutional point networks for large-scale point clouds, in: Proceedings of the European Conference on Computer Vision (ECCV), pp. 596–611.

Rottensteiner, F., 2008. Automated updating of building data bases from digital surface models and multi-spectral images: Potential and limitations, in: ISPRS Congress, Beijing, China, pp. 265–270.

Roynard, X., Deschaud, J.E., Goulette, F., 2016. Fast and robust segmentation and classification for change detection in urban point clouds, in: ISPRS 2016-XXIII ISPRS Congress, pp. 693–699.

Rußwurm, M., Wang, S., Korner, M., Lobell, D., 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.

Sande, C.V.D., Soudarissanane, S., Khoshelham, K., 2010. Assessment of relative accuracy of ahn-2 laser scanning data using planar features. Sensors 10, 8198–8214.

Sandric, I., Mihai, B., Savulescu, I., Suditu, B., Chitu, Z., 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.

Shi, S., Wang, X., Li, H., 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, W., Zhang, M., Zhang, R., Chen, S., Zhan, Z., 2020. Change detection based on artificial intelligence: State-of-the-art and challenges. Remote Sensing 12, 1688.

Siddiqui, F.U., Awrangjeb, M., 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.

Sofina, N., Ehlers, M., 2016. Building change detection using high resolution remotely sensed data and gis. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 9, 3430–3438.

Tchapmi, L., Choy, C., Armeni, I., Gwak, J., Savarese, S., 2017. Segcloud: Semantic segmentation of 3d point clouds, in: 2017 international conference on 3D vision (3DV), IEEE. pp. 537–547.

Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J., 2019. Kpconv: Flexible and deformable convolution for point clouds, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6411–6420.

Touati, R., Mignotte, M., Dahmane, M., 2020. Partly uncoupled siamese model for change detection from heterogeneous remote sensing imagery. Journal of Remote sensing and GIS 9.

Tran, T.H.G., Ressl, C., Pfeifer, N., 2018. Integrated change detection and classification in urban areas based on airborne laser scanning point clouds. Sensors 18, 448.

Varney, N., Asari, V.K., Graehling, Q., 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.

Vetrivel, A., Gerke, M., Kerle, N., Nex, F., Vosselman, G., 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. ISPRS journal of photogrammetry and remote sensing 140, 45–59.

Voelsen, M., Schachtschneider, J., Brenner, C., 2021. Classification and change detection in mobile mapping lidar point clouds. PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science 89, 195–207.

Wang, L., Huang, Y., Hou, Y., Zhang, S., Shan, J., 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, S., Suo, S., Ma, W.C., Pokrovsky, A., Urtasun, R., 2018. Deep parametric continuous convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2589–2597.

Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M., 2019b. Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) 38, 1–12.

Waser, L., Baltsavias, E., Eisenbeiss, H., Ginzler, C., Grün, A., Kuechler, M., Thee, P., 2007. Change detection in mire ecosystems: assessing changes of forest area using airborne remote sensing data. International archives of the photogrammetry, remote sensing and spatial information sciences 36, 313–318.

Widyaningrum, E., Bai, Q., Fajari, M.K., Lindenbergh, R.C., 2021. Airborne laser scanning point cloud classification using the dgcnn deep learning method. Remote Sensing 13, 859.

Wu, B., Wan, A., Yue, X., Keutzer, K., 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.

Xu, H., Cheng, L., Li, M., Chen, Y., Zhong, L., 2015a. Using octrees to detect changes to buildings and trees in the urban environment from airborne lidar data. Remote Sensing 7, 9682–9704.

Xu, Q., Chen, K., Sun, X., Zhang, Y., Li, H., Xu, G., 2020. Pseudo-siamese capsule network for aerial remote sensing images change detection. IEEE Geoscience and Remote Sensing Letters .

Xu, S., Vosselman, G., Oude Elberink, S., 2015b. Detection and classification of changes in buildings from airborne laser scanning data. Remote sensing 7, 17051–17076.

Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y., 2018. Spidercnn: Deep learning on point sets with parameterized convolutional filters, in: Proceedings of the European Conference on Computer Vision (ECCV), pp. 87–102.

Zagoruyko, S., Komodakis, N., 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.

Zhan, Y., Fu, K., Yan, M., Sun, X., Wang, H., Qiu, X., 2017. Change detection based on deep siamese convolutional network for optical aerial images. IEEE Geoscience and Remote Sensing Letters 14, 1845–1849.

Zhang, Z., Vosselman, G., Gerke, M., Persello, C., Tuia, D., Yang, M.Y., 2019. Detecting building changes between airborne laser scanning and photogrammetric data. Remote sensing 11, 2417.

Zhou, L., Ye, Y., Tang, T., Nan, K., Qin, Y., 2021. Robust matching for sar and optical images using multiscale convolutional gradient features. IEEE Geoscience and Remote Sensing Letters 19, 1–5.

Zhu, X.X., Tuia, D., Mou, L., Xia, G.S., Zhang, L., Xu, F., Fraundorfer, F., 2017. Deep learning in remote sensing: A comprehensive review and list of resources. IEEE Geoscience and Remote Sensing Magazine 5, 8–36. doi:10.1109/MGRS.2017.2762307.