remote sensing

MDPI

Article

PBFormer: Point and Bi-Spatiotemporal Transformer for Pointwise Change Detection of 3D Urban Point Clouds

Ming Han [1,2], Jianjun Sha [1,2,*], Yanheng Wang [1,2] and Xiangwei Wang [2]

[1] College of Intelligent Systems Science and Engineering, Harbin Engineering University, Harbin 150001, China
[2] Qingdao Innovation and Development Base, Harbin Engineering University, Qingdao 266000, China
* Correspondence: shajianjun_hh@163.com

Abstract: Change detection (CD) is a technique widely used in remote sensing for identifying the differences between data acquired at different times. Most existing 3D CD approaches voxelize point clouds into 3D grids, project them into 2D images, or rasterize them into digital surface models due to the irregular format of point clouds and the variety of changes in three-dimensional (3D) objects. However, the details of the geometric structure and spatiotemporal sequence information may not be fully utilized. In this article, we propose PBFormer, a transformer network with Siamese architecture, for directly inferring pointwise changes in bi-temporal 3D point clouds. First, we extract point sequences from irregular 3D point clouds using the k-nearest neighbor method. Second, we uniquely use a point transformer network as an encoder to extract point feature information from bitemporal 3D point clouds. Then, we design a module for fusing the spatiotemporal features of bi-temporal point clouds to effectively detect change features. Finally, multilayer perceptrons are used to obtain the CD results. Extensive experiments conducted on the Urb3DCD benchmark show that PBFormer outperforms other excellent approaches for 3D point cloud CD tasks.

Keywords: change detection; crossing fusion; point clouds; Siamese networks; transformer

Citation: Han, M.; Sha, J.; Wang, Y.; Wang, X. PBFormer: Point and Bi-Spatiotemporal Transformer for Pointwise Change Detection of 3D Urban Point Clouds. *Remote Sens.* **2023**, *15*, 2314. https://doi.org/10.3390/rs15092314

Academic Editor: Mohammad Awrangjeb

Received: 27 February 2023
Revised: 20 April 2023
Accepted: 26 April 2023
Published: 27 April 2023

1. Introduction

The acquisition of large-scale three-dimensional (3D) point clouds has become increasingly convenient with the development of 3D sensors such as depth cameras and lidar. This has led to the widespread use of 3D point clouds as a data source for change detection (CD) tasks. The aim of 3D CD studies is to identify changes in the geometric structure of objects. Effective 3D CD is critical to meet the growing demand in remote sensing fields, such as for updating the geographic information database [1], surveying urban development [2,3], monitoring forest change [4], and assessing landslide or building damage [5].

Classification and CD are the two main steps involved in existing 3D CD techniques. According to the steps of CD and classification in an algorithm, 3D CD methods can be generalized into three categories: pre-classification, post-classification, and integrated methods [6]. Pre-classification CD approaches first detect changes between bitemporal 3D point clouds, then classify and characterize them. Xu et al. [7] first built an octree to separate changed regions, which were classified by the AutoClust algorithm. Du et al. [8] calculated grey-scale similarity and a height difference of bitemporal 3D data, then utilized the graph cuts method to extract changed regions and refined the results by separating ground points from non-ground points. Bangera et al. [9] proposed a hybrid learning-based approach called HGI-CD. It first extracts significantly changed regions from the scene, which are then used to construct change graphs, fed into a graph convolution network (GCN) [10] with an inception network [11], and finally classified by multilayer perceptrons (MLPs).

On the other hand, post-classification CD approaches segment the whole 3D point clouds into several components, based on categories of objects, then these components are matched one by one in two periods, and the difference between them is detected.

Awrangjeb et al. [12] first classified two-dimensional (2D) footprints of buildings extracted from LiDAR data and aerial images, then compared them on a 2D map to obtain significant changes. The method in [13] first segmented each point cloud by extracting buildings from a scene. Then, a 3D surface difference map was created for CD by computing the distance between a 3D point in one set and the nearest plane in the other set.

The accuracy of the previous CD limits the final results of pre-classification CD methods. Similarly, the result of segmentation is a limiting factor in the performance of post-classification CD methods. Some integrated approaches are studied for 3D CD to overcome the above problems. Tran et al. [14] attempted to integrate CD and classification to form a unified procedure. The study in [14] is based on a hand-crafted feature called 'Stability', which is defined as the ratio of the number of points in the spherical neighborhood of a point to the number of points in the cylindrical neighborhood of the same point, but in the other point cloud. These features extracted from bitemporal point clouds are used to train a random forest model to solve CD tasks. Although this method is a comparatively novel approach, it still cannot overcome the limitations of hand-crafted features.

The feature extraction capability of deep learning Is stronger than that of traditional approaches in most vision tasks [6,15]. Therefore, a better research direction is to investigate one-stage CD methods based on end-to-end deep learning networks. Pioneering work in deep learning on 3D point clouds has focused on shape classification [16–18], semantic and instance segmentation [19,20], object detection and tracking [21,22], etc. Nevertheless, to date, there has been relatively little CD research based solely on 3D point clouds, and even less focused on directly processing 3D point clouds without any pre/post-processing procedure. SiamGCN [9] is an end-to-end, Siamese, and shared dynamic graph convolutional neural network (DGCNN) [23] designed to identify the change of a pair of point sets. Feature to Feature Supervoxel-based Spatial Smoothing (F2S3) is a deep learning framework developed by Gojcic et al. [24]. F2S3 first estimates a 3D displacement vector field, then filters and smooths the 3D displacement vector field to analyze displacements and changes. However, the pipeline of F2S3 is not fully automatic. It requires a complex process of selecting appropriate hyper-parameters. A follow-up method proposed by Gojcic et al. [25] attempts to realize the automation of the F2S3 workflow. The hyper-parameters are replaced by values from the input data. However, there are three main issues that have not been addressed by the above methods:

(1) A key issue in single-point CD approaches is the extremely sparse semantics of a single point in 3D Euclidean space, making it difficult to use a single point as the object of CD study.

(2) 3D point clouds from distinct periods or different devices always lack exact spatial correspondence, making it challenging to extract fine-grained and robust features from a pair of unrestricted 3D points.

(3) A 3D point cloud can be treated as a sequence distributed in a 3D Euclidean space, and the point cloud of different periods can be considered as a time series. However, current 3D CD methods have rarely been studied from the perspective of handling spatiotemporal series.

Transformers [26,27] have been successfully applied to natural language processing (NLP) tasks, and have the ability to capture the rich semantic relationships between two sequences of words in machine translation. Inspired by research in NLP [28,29], several works [30–33] have applied transformers to computer vision (CV) tasks to learn the spatial sequence information in images. In the context of 3D CD tasks, transformers could be a suitable network structure if the 3D point cloud is treated as a spatiotemporal sequence.

Motivated by the above analysis, in this paper, we propose a **p**oint and **b**i-spatiotemporal trans**former** (PBFormer) that effectively detects detail changes in 3D point clouds. PBFormer directly processes each point in bitemporal point clouds. First, we select the spatial neighborhoods of each point in two periods as the analysis objects, since changes in 3D space usually occur in an area, rather than at an isolated point without volume. Second, our Siamese structure encoder consist of a pair of point transformers (PTs) sharing weights, which ex-

tract the spatial sequence features from bitemporal 3D point clouds. Third, we designed a bi-spatiotemporal crossing transformer (BT) to fuse two representations of bitemporal point clouds and output the changed type of a point pair using MLPs. Extensive experiments have been conducted on the Urb3DCD benchmark [34], demonstrating that our method outperforms other excellent approaches, including C2C, M3C2, random forest, etc. The main contributions of our work are as follows:

(1) PBFormer, an end-to-end Siamese transformer network, is proposed to solve the above three issues of the existing methods. PBFormer can directly detect pointwise changes in raw 3D point clouds. It considers two neighborhoods as the change detection objects, which contain more abundant semantics than isolated points. Experimental results on the public dataset show that PBFormer is suitable for point cloud CD.

(2) PT is designed to extract point features using transformer encoders, which project 3D coordinates into position embeddings. Our proposed PT considers point clouds as spatial sequences and models them, enabling it to more fully extract fine-grained and robust geometric features from raw 3D point clouds.

(3) BT is designed to fuse a pair of point features in intersection form to effectively detect changes in bitemporal point clouds. Our proposed BT considers the bitemporal point cloud as a temporal sequence and models it to obtain the temporal sequence information and capture subtle differences between bitemporal point clouds.

2. Related Works

- Point-Based Networks

Among the early networks designed for point cloud perception tasks, PointNet [16] is a pioneering work, inspiring a number of methods that can directly consume points. These approaches can be divided into two categories: pointwise MLPs-based, and kernel point convolutions based. PointNet utilized the pointwise shared MLP strategy, and the methods in Refs. [18,20,21] all belong to this family. KPConv [35] and PAConv [36] are typical methods based on kernel point convolutions. The prominent advantages of transformers, such as long-distance characteristics, massively parallel computing, and less inductive bias, have inspired some explorations [17,37–39] to extend transformers to point cloud perception tasks. In this paper, we attempt to apply transformers to 3D CD tasks with minimal inductive bias.

- Siamese Networks

Siamese networks [40] were first proposed for signature verification. Many subsequent studies have explored the use of Siamese networks for metric learning and other applications, such as image information retrieval [41], image comparing [42,43], and object tracking [44,45]. Siamese networks have a coupling architecture based on two network branches that share the same weights. This network structure is particularly suitable for CD because of its ability to maximize the representation with different labels and minimize others with identical labels under the supervised learning paradigm [46–48].

- Transformers

Transformers are highly scalable learners for language and vision tasks [49–51]. The encoder of the original transformer consists of two elementary modules, self-attention (SA) and MLP. In SA, each element of an input sequence is linearly projected into three vectors, namely query, key, and value. Then, these three vectors participate in the calculation called scaled dot-product attention [26] and output a more abstract feature. As an improvement of SA, multi-head self-attention (MSA) can project query, key, and value vectors linearly in parallel to enhance the feature extraction ability. The calculation process of MSA is detailed in Ref. [26].

In each encoder, the first layer is a multi-head self-attention (MSA) mechanism, and the second layer is a simple MLP composed of fully connected networks (FCN) and rectified linear unit (ReLU) activation functions. Each MSA and MLP, followed by layer normalization (LN) and residual connections, form a transformer encoder. Moreover, to incorporate the relative or absolute position information of the input sequence, positional embeddings $X_{\text{pos}} = \{X_{\text{pos-1}}, X_{\text{pos-2}}, X_{\text{pos-3}}, \ldots, X_{\text{pos-n}}\}$, where n is the length of the input sequence, are added to the input embeddings X_{input} before the first encoder. Let $X_{\text{i}} = \{x_1, x_2, x_3, \ldots, x_{\text{n}}\}$ be the token sequence, and Y_{i} be the output of the i^{th} encoder, where $i = 1, 2, 3, \ldots, L$, so the calculation process can be expressed as:

$$X_1 = X_{input} + X_{pos}, \tag{1}$$

$$X_i^{'} = X_i + LN(MSA(X_i)), \tag{2}$$

$$Y_i = X_i^{'} + LN(MLP(X_i^{'})), \tag{3}$$

and

$$X_{i+1} = Y_i \tag{4}$$

In the model design, Vision Transformer (ViT) [30] is as close as possible to the original transformer [26]. ViT reshapes a 2D image into a sequence of flattened 2D patches in order to obtain sequential input and concatenates the sequence with a learnable embedding X_{CLS}, which is similar to the classification token (*CLS*) in BERT [28]. In addition, LN is adjusted before MSA and MLP in each encoder. Therefore, the calculation process of the i^{th} encoder in ViT can be expressed as:

$$X_1 = X_{CLS} \oplus (X_{input} + X_{pos}), \tag{5}$$

$$X_i^{'} = X_i + MSA(LN(X_i)), \tag{6}$$

$$Y_i = X_i^{'} + MLP(LN(X_i^{'})), \tag{7}$$

and

$$X_{i+1} = Y_i \tag{8}$$

where $\oplus$ is a concatenation operator. The final embedding X_{CLS} is separated from Y_{L} and fed into a classification head implemented by MLPs.

However, due to the different densities of information and the data format [33], transformers cannot be used directly for CV tasks. Moreover, this issue is even more challenging to solve when processing 3D point clouds than when processing images. Considering the sparse semantics and irregular format of point clouds, we introduced a transformer suitable for CD tasks on 3D point clouds. We also investigated the effect of positional encoding in point cloud learning, since point clouds contain sufficient spatial information.

3. Methodology

In this section, we first briefly introduce the process of extracting point sequences. Then, we present how the PT encoder extracts geometric features from point sequences. Finally, we show how the BT decoder learns differential features. The architecture of the proposed PBFormer is shown in Figure 1.

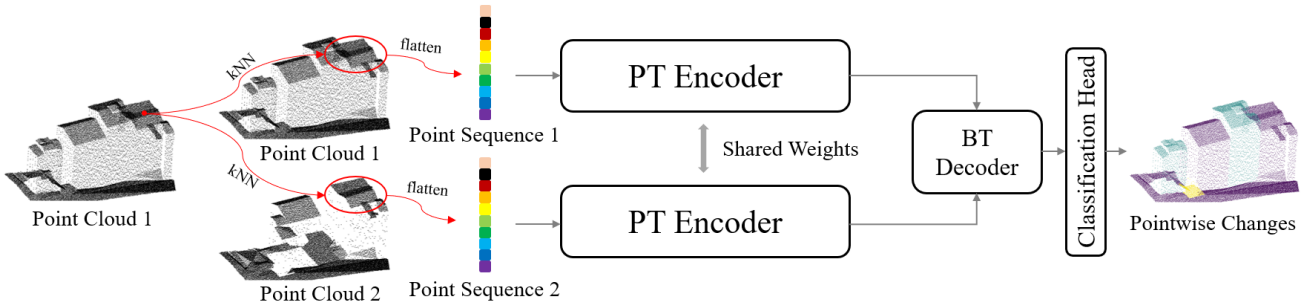


Figure 1. The overall architecture of our proposed PBFormer. We use kNN to cut two fixed-size sequences from the original point clouds and feed them to both shared encoders. The process of detecting a point in the former period is described in the figure, and the process for detecting a point in the later period is similar.

3.1. Generating Point Sequences

To solve the problem of lacking semantic information at a single point, we calculate the change category of each 3D point from two sequences generated by searching neighbor points in two periods, respectively, with kNN [52,53]. The idea of kNN is intuitive, simple, and efficient, retrieving the k-nearest points as a neighborhood by Euclidean distances.

Suppose we have two sets of point clouds $S_1 \in \mathbb{R}^{n_1 \times c}$ and $S_2 \in \mathbb{R}^{n_2 \times c}$, where c is the number of channels and n_1 and n_2 are the number of points in S_1 and S_2, respectively. First, we build kd-trees [54] for S_1 and S_2 to accelerate the retrieval computation. Then, according to the k value of kNN, we use kd-trees to retrieve two neighborhoods of a point $p \in S_1$ in S_1 and S_2, respectively. The two neighborhoods form a point sequence pair, $P_1 \in \mathbb{R}^{k \times c}$ ($P_1 \subset S_1$) and $P_2 \in \mathbb{R}^{k \times c}$ ($P_2 \subset S_2$). If there are almost no matching points in S_2 in the same spatial space, kNN is still being executed to generate the second point sequence. Although the neighborhood represented by the second point sequence may be inconsistent with the first one, this difference matches the actual situation (except for occlusion) and provides a pair of point sequences with differences. Finally, this pair of point sequences is fed into PBFormer for CD. The procedure is the same for detecting points in S_2.

Since the bitemporal point clouds do not have strict correspondence, the consistency of the two sequences may also be affected. However, when the appropriate k value is set, the region represented by the two sequences may be approximately the same, and the geometric structure information contained in them will also be approximately the same. If similar regions cannot be obtained, there are likely obvious structural changes.

3.2. Point Transformer Encoder

- Positional Embedding

In order to capture the relationship between points in a sequence, it is necessary to adopt an appropriate method for adding positional embeddings. There are three common choices: none, fixed, and learned [55]. Without additional positional information, embeddings of the input sequence are immediately fed into the encoder. The fixed method generates a series of positional encodings using a specific function, such as sine and cosine functions. The learned method usually uses learned positional embeddings to encode spatial information, and the dimension of the learned positional embeddings can vary according to the data structure.

Transferring transformers to point cloud learning, we first consider a point sequence $P \in \mathbb{R}^{k \times c}$ as a sequence with relative position information, where k is the number of points in P and c is the number of the feature dimensions of a point $p \in \mathbb{R}^c$ in P. In the raw point clouds, the format of p is usually the coordinate (x, y, z) concatenated with the intensity *Int.* and the true RGB color (r, g, b). Note that the positional information is contained in the original feature of p. Suppose we further consider P as a sequence distributed in 3D space; (x, y, z) can serve as positional encoding. It is also noteworthy that (x, y, z) represents

the absolute positional information of p in the entire point cloud. If we only consider the geometric structure at the local spatial scale, the absolute positional encoding changes with the rigid transformation, and it is thus not very robust to jitter, while the relative positional encoding merely reflects the distance x_{dist} between p and a reference point p_{r}, which is invariant to the rigid transformation.

In GCN [10], the combinatorial Laplacian matrix $\mathbf{L}$ is introduced to replace the adjacency matrix $\mathbf{A}$, i.e., $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the degree matrix. The Laplacian matrix $\mathbf{L}$ can help GCN reduce the complexity and simplify the process of computing convolution because convolution is easier to calculate in the Fourier domain, and the process of the spectral decomposition of the Laplacian matrix is relatively simple. A recent work [37] also defines an operator analogous to the discrete Laplacian matrix that uses the identity matrix $\mathbf{I}$ to approximate the degree matrix and the product of the query matrix $\mathbf{Q}$ and the key matrix $\mathbf{K}$ in self-attention networks to the adjacency matrix. This operator can also effectively improve the performance of the model in [37]. In fact, we find that a feature representing offset also plays an important role in other point cloud learning algorithms [20,38], regardless of whether we derive this offset feature in low-dimensional or high-dimensional feature space. Since arbitrary points in P can be moved to p_{r} by a certain corresponding transformation matrix $\mathbf{T}$, x_{dist} can be expressed as:

$$\begin{aligned} x_{dist} &= p - p_r \\ &= p - \mathbf{T}p \\ &= (\mathbf{I} - \mathbf{T})p \\ &\approx \mathbf{L}_{dist}p, \end{aligned} \tag{9}$$

where $\mathbf{T}$ is comparable to the adjacency matrix $\mathbf{A}$. Similarly, we could obtain an approximate discrete Laplacian $\mathbf{L}_{\text{dist}}$, which provides a way to add positional encoding. Therefore, we simply revise an original coordinate as a positional embedding, called self-positional embedding (SPE), as follows:

$$x_{s-pos} = \varphi(LN((x_p, y_p, z_p) - (x_{p_r}, y_{p_r}, z_{p_r}))) \tag{10}$$

where φ is a simple mapping function such as FCN to approximate $\mathbf{L}_{\text{dist}}$, and p_{r} is the point used to generate P. We also add LN to improve stability and convergence during training.

- Point Transformer

The efficient implementation of the original transformer is almost out of the box in ViT. We also purposely followed the mainstream transformer architecture to introduce as little inductive bias as possible.

Our proposed network is illustrated in Figure 2. To handle 3D point clouds, we consider the neighborhood $P = \{p_1, p_2, \ldots, p_{\text{k}}\}$ of each point p comparable to a flattened 2D patch in ViT and map P into a linear projection to make point embedding $E \in \mathbb{R}^{k \times d}$, where d is the number of channels after mapping. We have

$$E = \theta(P), \tag{11}$$

where θ represents simple mapping functions, such as FCN.

Then, we extract the coordinates of all points in P, map them into SPE $E_{\text{s-pos}} \in \mathbb{R}^{k \times d}$, and add $E_{\text{s-pos}}$ to E. Thereafter, an extra learnable embedding $X_{\text{CLS}-0} \in \mathbb{R}^{\text{d}}$, similar to BERT's special $[CLS]$ token, is prepended to the beginning of the E. During training, $X_{\text{CLS}-0}$ could learn the category of p_{r} from the geometric structure of P. We have

$$E_{s-pos} = \psi(P_{xyz}), \tag{12}$$

and

$$X = X_{CLS-0} \oplus (E + E_{s-pos}), \tag{13}$$

where ψ represents simple mapping functions such as FCN.

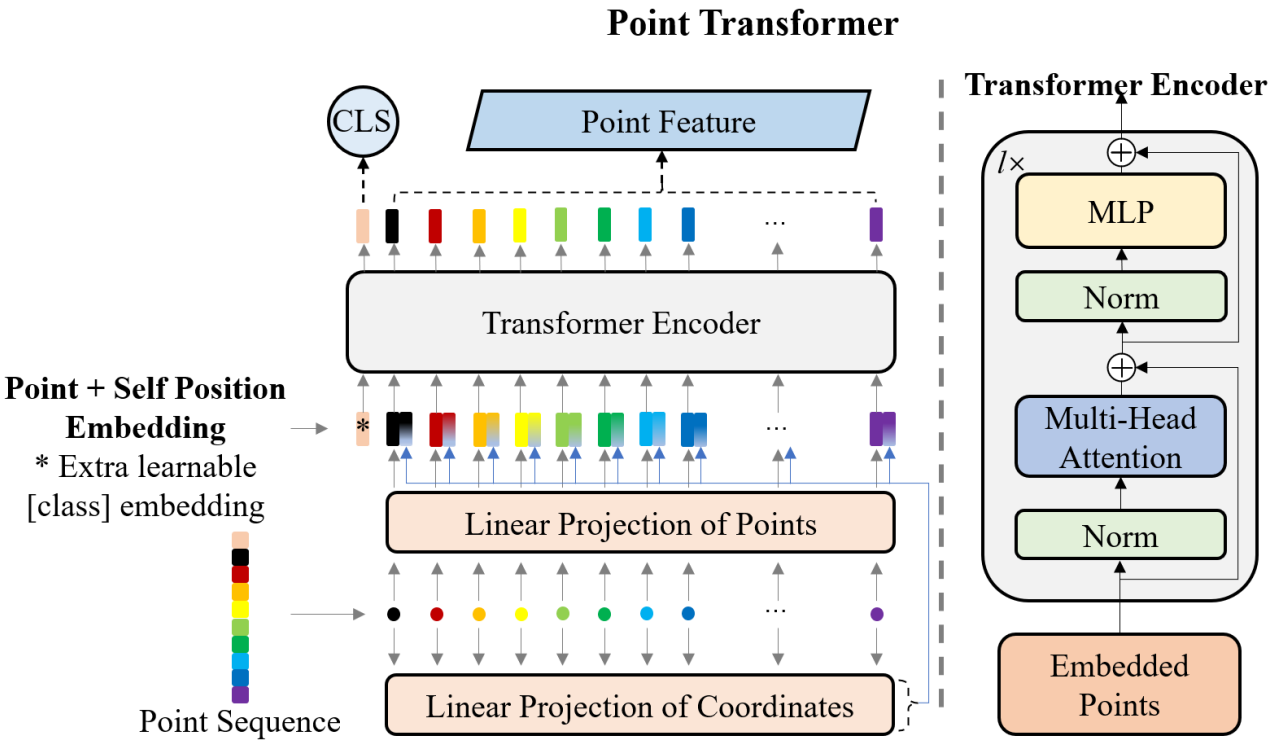


Figure 2. The illustration of the PT encoder. We linearly embed raw features and coordinates of each point into two sequences, respectively, then add the two embeddings and feed the sequence of embeddings into a mainstream transformer encoder to obtain more effective representations. In order to distinguish changes, we use an approach where an extra learnable [*CLS*] token is prepended to the sequence. The structure of the transformer encoder on the right is inspired by other common transformer networks. finally, the output representation is split into two parts: the [*CLS*] token and the point feature.

Eventually, X is fed into the transformer encoder, which outputs a [*CLS*] token $X_{\mathrm{CLS}} \in \mathbb{R}^{\mathrm{d}}$ and point features $X_{\mathrm{PF}} \in \mathbb{R}^{(k-1)\times d}$. The transformer encoder is composed of l basic blocks. Each block has two elementary layers, MSA and MLP, followed by LN and residual connections. The operation in the lth block can be expressed as:

$$X_l^{'} = X_l + MSA(LN(X_l)), \tag{14}$$

and

$$X_{l+1} = X_l^{'} + MLP(LN(X_l^{'})), \tag{15}$$

Let $X = \{x_1, x_2, x_3, \ldots, x_{\mathrm{k}}\}$, so $X_{\mathrm{CLS}} = \{x_1\}$ and $X_{\mathrm{PF}} = \{x_2, x_3, \ldots, x_{\mathrm{k}}\}$. Therefore, the outputs of two branches in PBFormer can be expressed as $X_{\mathrm{b}} = \{X_{\mathrm{CLS\text{-}b}}, X_{\mathrm{PF\text{-}b}}\}$ and $X_{\mathrm{a}} = \{X_{\mathrm{CLS\text{-}a}}, X_{\mathrm{PF\text{-}a}}\}$, where b and a mean *before* and *after*.

3.3. Bi-Spatiotemporal Crossing Transformer Decoder

After acquiring two features in different periods, the difference in space between them needs to be further analyzed. The change result y can be expressed as:

$$y = f(X_b, X_a), \tag{16}$$

where f is a function for learning the difference.

Nevertheless, f implemented by simple network architectures may be laborious for learning complex changes because they only perform comparison operations, and scarcely

establish contextual relationships between different temporal features. Therefore, we attempt to take an appropriate approach of fusing X_b and X_a, and propose a well-designed feature fusion network based on the powerful sequence encoding ability of transformers.

- Differential Features Fusion

There are mainly three types of fusion algorithms: full-fused, part-fused and crossing-fused [32]. Full-fused means that all features in X_b and X_a participate in each operation of f, as shown in Figure 3a. However, it is unacceptable because such fusion requires considerable computing time. Figure 3b,c shows two types of part-fused processing, one with only the point features fused, and the other with only the [*CLS*] token fused. Obviously, the key information of the two representations is not fully exploited.

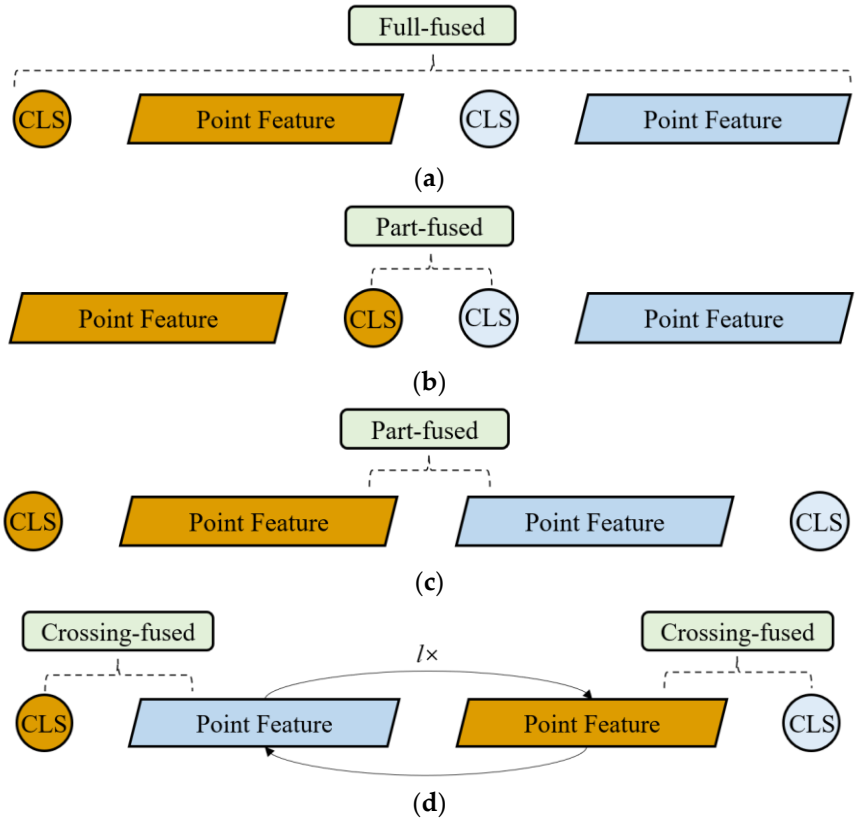


Figure 3. The comparison of different fusion styles: (**a**) full-fused; (**b**,**c**) part-fused; (**d**) crossing-fused. We use amber and blue to indicate the differences in time and space between the two representations.

According to the correspondence of the scene before and after changes, the change in the representation in one period could be captured by the other representation. Thus, we try to achieve this goal by exchanging information in two representations, as shown in Figure 3d. This scheme, called crossing-fused, considers two point features as mediums for exchanging information. Throughout the fusion process, the two mediums successively transmit their information to two [*CLS*] tokens.

First, $X_{\text{PF-b}}$ carries the spatial structure information before changes, and $X_{\text{CLS-a}}$ indicates that the changes are fused; then $X_{\text{CLS-a}}$ is fused with $X_{\text{PF-a}}$ again. In this process, $X_{\text{CLS-a}}$ learns useful spatial features before and after the change. Second, $X_{\text{CLS-b}}$ is subjected to the same computation as $X_{\text{CLS-a}}$, where $X_{\text{PF-b}}$ and $X_{\text{PF-a}}$ replace each other. Third, the above process is repeated l times to obtain two adequately fused [*CLS*] tokens.

- Bi-spatiotemporal Crossing Transformer

The proposed feature fusion module based on a crossing-fused scheme is shown in Figure 4. Since transformers have the ability to capture long-range interactions and

dependencies, we implement each fusion operation using a simple transformer network called the crossing block. Our crossing block consists of MSA, followed by LN and residual connections. After l times of crossing fusion, we obtain two fused [*CLS*] tokens. During this process of crossing fusion, the two [*CLS*] tokens could learn geometric structure information at the other branch. The operational process of a branch in the BT decoder can be expressed as:

$$X_{b-a} = (X_{CLS-b}^{i-1} \oplus X_{PF-a}), \tag{17}$$

$$\begin{aligned} X_{b-a}^{'} &= X_{b-a} + MSA(LN(X_{b-a})) \\ &= (X_{CLS-b}^{'} \oplus X_{PF-a}), \end{aligned} \tag{18}$$

$$X_{b} = (X_{CLS-b}^{'} \oplus X_{PF-b}), \tag{19}$$

$$\begin{aligned} X_{b}^{'} &= X_{b} + MSA(LN(X_{b})) \\ &= (X_{CLS-b}^{'} \oplus X_{PF-b}), \end{aligned} \tag{20}$$

where $i = 1, 2, 3, \ldots, l$. Finally, the l-th $X_{\text{CLS-b}} \in \mathbb{R}^{\text{d}}$ is added to the l-th $X_{\text{CLS-a}} \in \mathbb{R}^{\text{d}}$ and fed into a classification head implemented by MLPs.

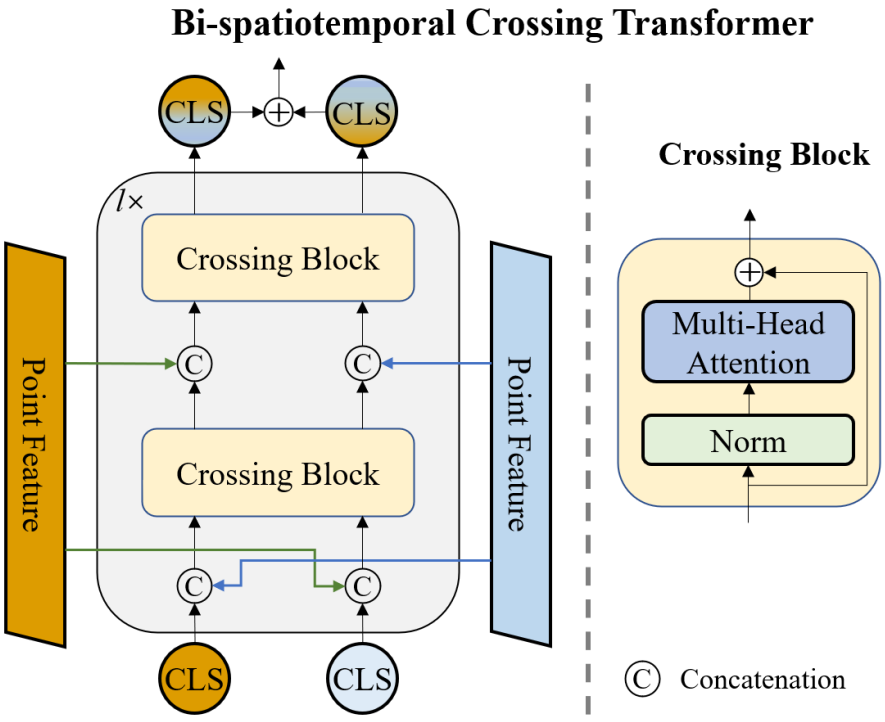


Figure 4. The network structure of our proposed BT decoder. Two [*CLS*] tokens are first cross-fused with the other point feature and then with their own. This process is then repeated l times, outputting two fused mixed-color tokens. The crossing block on the right is a naive transformer encoder without MLP.

4. Experiments

In this section, we evaluate PBFormer through extensive experiments. First, the experimental setup is presented in detail. Second, PBFormer is compared to several methods that present results at different levels. Third, the relationship between training set size and model performance is analyzed. Finally, ablation studies are performed for our SPE and BT decoder.

4.1. Setup

- Dataset

The Urb3DCD [34] is an urban point cloud dataset based on the 3D model of a real city with level of detail 2 (LoD2) precision. Urb3DCD consists of 5 sub-datasets, which are divided according to the resolution and noise level. The sub-dataset-1 provides 3 different training sets: 1-a is a small training set, with only 1 pair of point clouds; 1-b is a normal training set, with 10 pairs of point clouds like the others in sub-datasets-2, 3, 4 and 5; 1-c is the largest training set, with 50 pairs of point clouds. There is 1 pair of point clouds in the validation set and 3 pairs of point clouds in the test set for all sub-datasets. The raw 3D points in Urb3DCD contain only 3D coordinates, without color and intensity information, and can be divided into 3 classes: new, demolished, and unchanged.

- Implementation

Our proposed PBFormer was implemented with PyTorch. All experiments were conducted on the same machine with an Intel i9-10900K @3.7GHz CPU and an NVIDIA RTX3060 GPU. The AdamW optimizer was used, without changing any parameters. The initial learning rate was set to 0.0002 and scheduled by cosine annealing, with warm restarts at each step. For the evaluation on Urb3DCD and analyzing the influence of the training set size, the model was trained for 50 epochs. The batch size was set to 32, and the number for cutting point sequences, k, was set to 256. Cross-entropy was adopted as the loss function. We stacked 4 transformer encoders in the PT encoder and 4 crossing blocks in the BT decoder. During training and testing, the whole raw point clouds are fed into PBFormer to detect pointwise changes without projecting, rasterizing, and tiling.

The mean intersection over union (mIoU) can be used to measure the similarity between predicted results and the ground truth and evaluate an algorithm's performance. Therefore, our experiments use the mIoU of all change categories as the standard metric. Let $M_{\text{confusion}}$ be the n $\times$ n confusion matrix; then, mIoU can be expressed as:

$$mIoU = \frac{1}{n}\sum_{i=1}^{n}\frac{TP}{FN + FP + TP}, \tag{21}$$

where true positive (TP) indicates the number of points that change in ground truth and as well as in the predicted point clouds, false negative (FN) represents the number of points that change in ground truth but do not change in the predicted point clouds, and false positive (FP) is the number of points that do not change in ground truth but do change in the predicted point clouds.

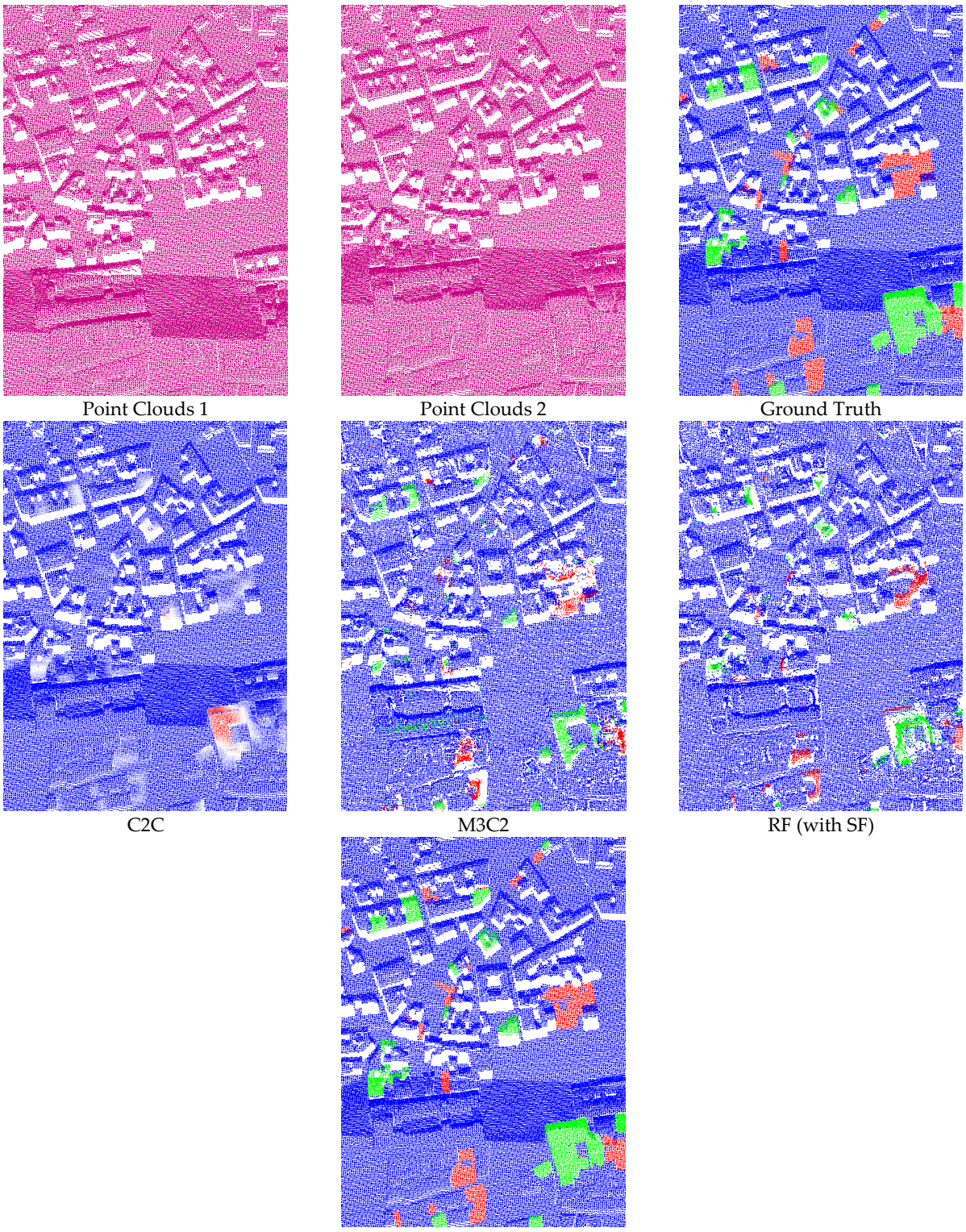
4.2. Change Detection on Benchmark

- Evaluation on Urb3DCD

Five well-established methods are selected for comparison and verification of the effect of the proposed PBFormer. (1) Methods based on DSM difference (DSMd) [56] with empirical thresholds are applied to process 2D images generated from DSMs and are displayed at the 2D pixel and patch level. (2) A Siamese FFN [48] and a Siamese convolutional neural network (CNN) [47] are applied to detect 2D images, but they are only displayed at the 2D patch level. (3) The cloud-to-cloud (C2C) [57] is a comparison method at the 3D points level which is based on point-to-point Hausdorff distance. C2C cannot discriminate the change category, providing only binary results. (4) The multi-scale model-to-model cloud comparison (M3C2) [58] method uses a hand-crafted feature based on the distance along the normal direction between two point clouds, and M3C2 also provides results at the 3D point level. (5) A random forest (RF) algorithm with the hand-crafted stability feature (SF) [14] can provide results at the 3D point level.

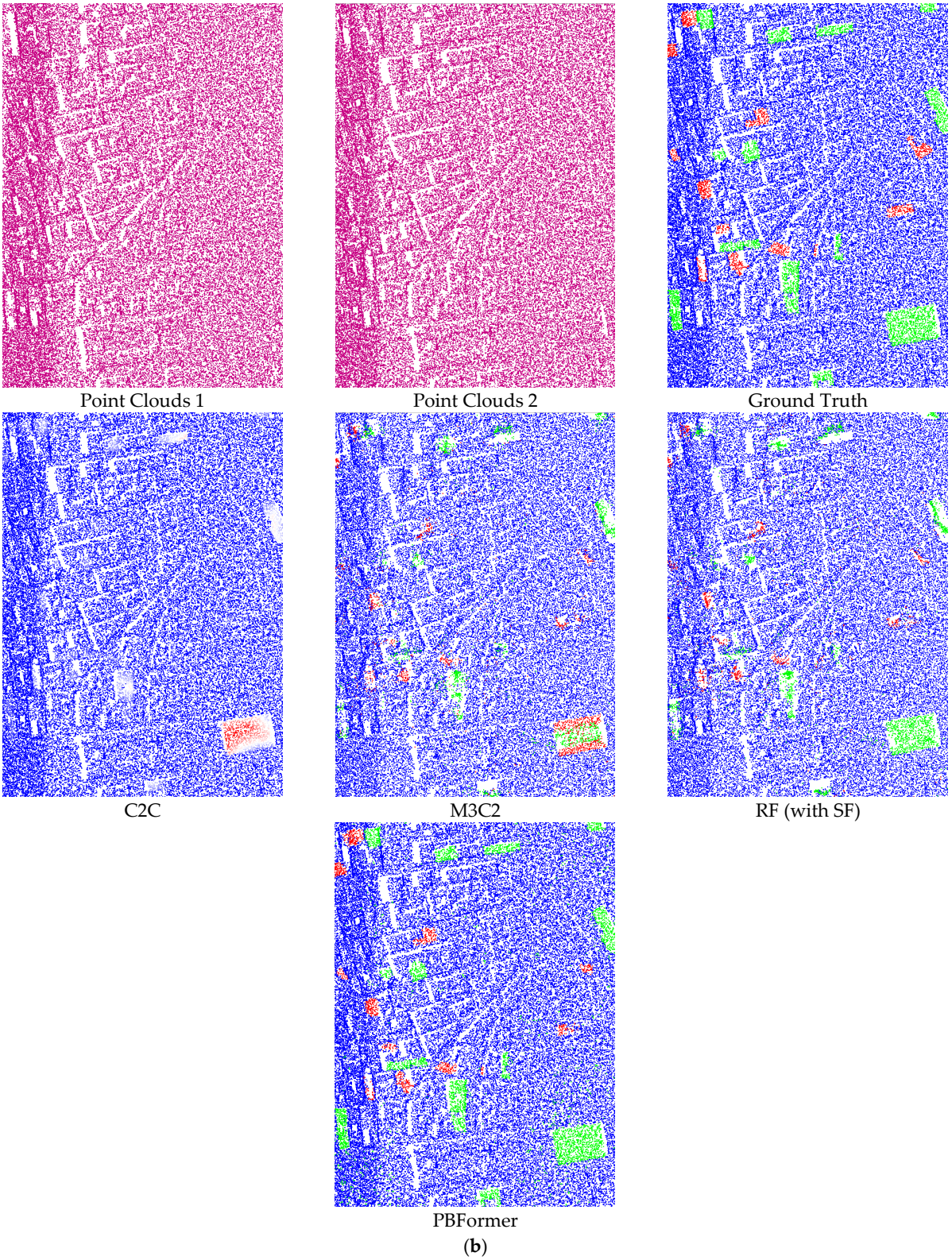
The effectiveness of PBFormer is evaluated on all standard-sized sub-datasets by comparing our method with other excellent methods, both qualitatively and visually.

The quantitative comparison is shown in Table 1, and the qualitative results are visually presented in Figures 5 and 6.



(**a**)

Figure 5. *Cont.*



(**b**)

Figure 5. *Cont.*

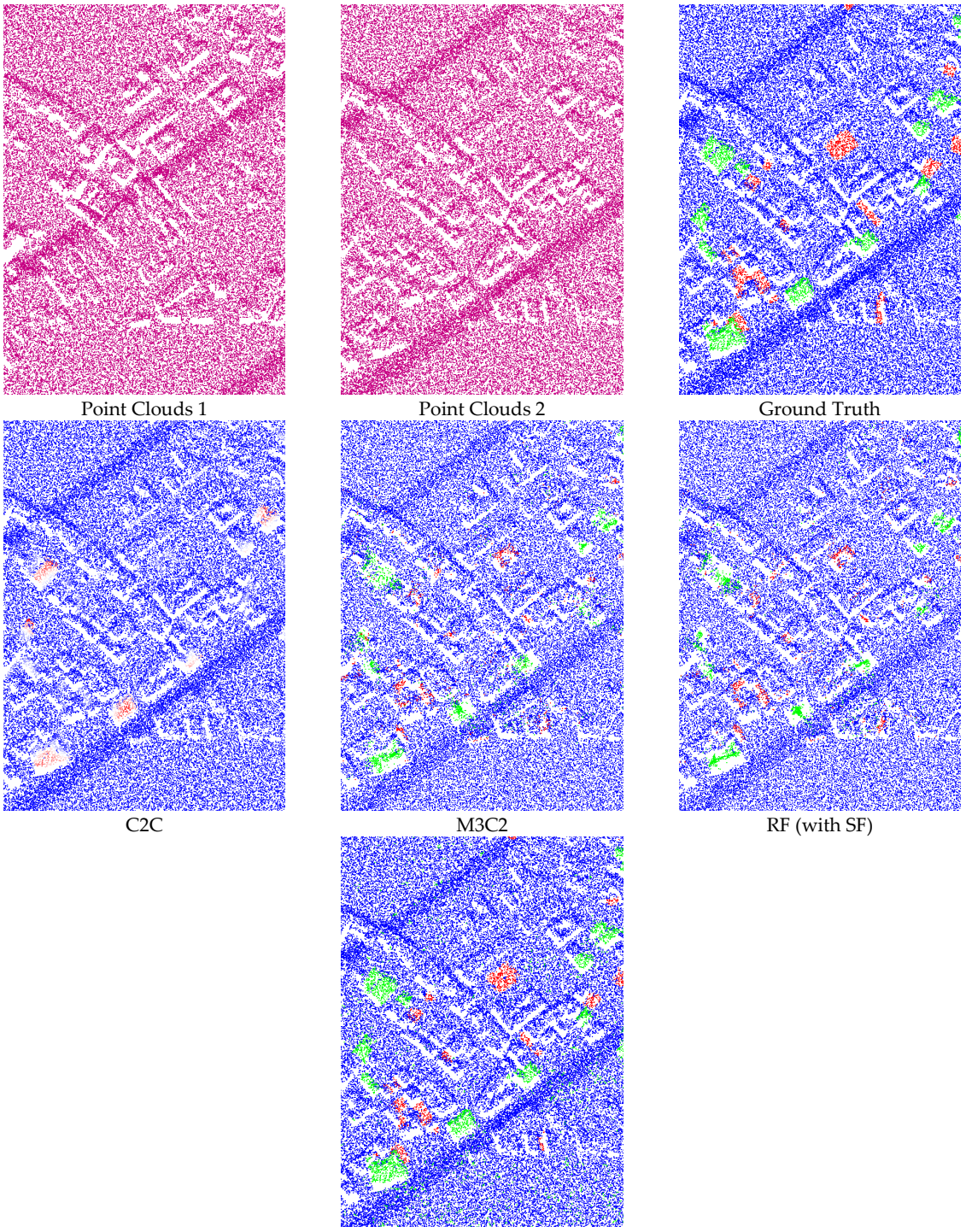


Figure 5. Qualitative results of PBFormer on different test sub-sets of Urb3DCD at the building group level: (**a**) noise scan direction: 0.05 m, scan angle: -20° to 20°; (**b**) noise scan direction: 1 m, scan angle: -20° to 20°; (**c**) noise scan direction: 1 m, scan angle: -10° to 10°. All unchanged points are shown in blue, the new points are indicated in green, and the demolished points appear in red.

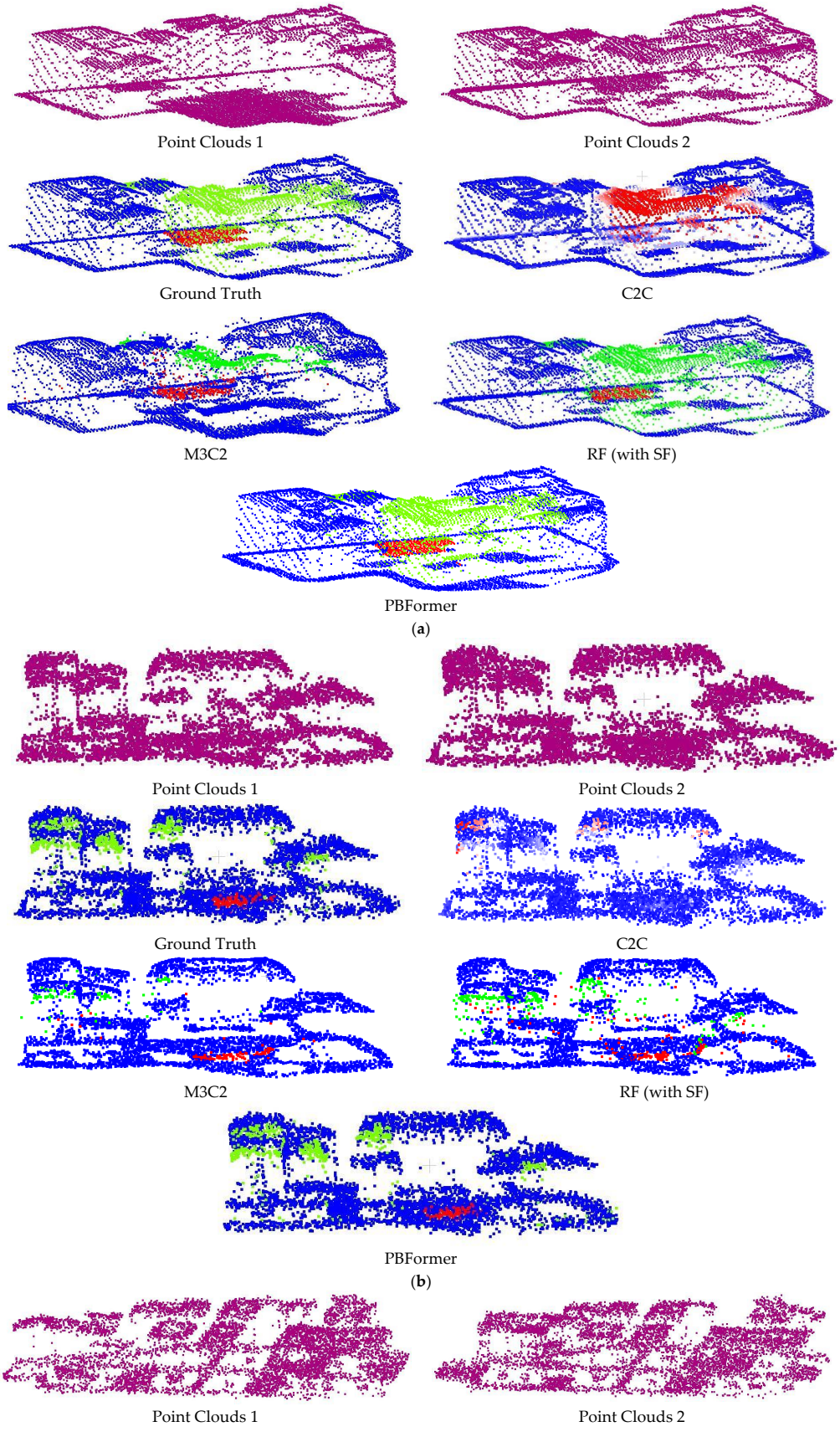


Figure 6. *Cont.*

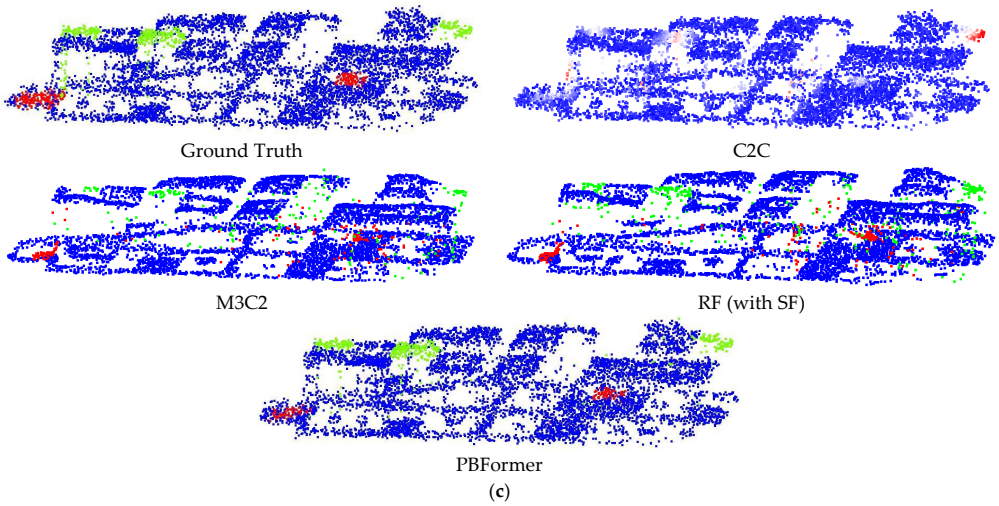


Figure 6. Qualitative results of PBFormer on different test sub-sets of Urb3DCD at the single building level: (**a**) noise scan direction: 0.05 m, scan angle: -20° to 20°; (**b**) noise scan direction: 1 m, scan angle: -20° to 20°; (**c**) noise scan direction: 1 m, scan angle: -10° to 10°. All unchanged points, new points, and demolished points are marked in blue, green, and red, respectively.

It can be seen that: (1) The results of methods processing DSMd with empirical thresholds consistently failed to perform satisfactorily on all sub-datasets. Otsu's threshold algorithm [59] and the opening filter [60] can significantly improve the performance, meaning that nearly all of their results achieved more than 80% of mIoU. However, in these results, we can observe the change of 3D point clouds only from the top view. (2) FFN and Siamese CNN showed inconspicuous results despite being trained. A possible reason is that the two networks have simple architectures and are not specially designed for point clouds. (3) C2C only provided binary results, and the accuracy was directly determined by the empirical threshold. Although we additionally used the fuzzy threshold, it is still unsatisfactory, even in terms of visual effect (as shown in Figure 5a–c). (4) M3C2 achieved results close to the lowest mIoU. This hand-crafted feature is not robust enough, especially when there is excessive noise in the point clouds. (5) The RF (with SF) algorithm also obtained merely ordinary results due to the limitation of hand-crafted SF. (6) Last but not least, our PBFormer outperforms the above approaches regarding mIoU and performs better on 5 sub-datasets.

- Influence of Training Set Size

We further explored the relationship between the size of the training set and the performance of the model, as shown in Table 2. It can be seen that: (1) When training models on sub-dataset-1-a (including only 1 pair of point clouds), the mIoU score of our PBFormer is lower than those of the FFN and RF (with SF). The reason for this is that transformers have more powerful modeling flexibility and less inductive bias, so they generally needs to be supported by large-scale datasets. Therefore, it is challenging to train PBFormer on a small-scale training set, which leads to slightly poor results in our approach. (2) Increasing the training set size is an effective solution to improve the detection accuracy, which is more obvious in FFN and our PBFormer. Although the result obtained on sub-dataset-1-b (10 pairs of point clouds) was used as a baseline in our work, our PBFormer may achieve better performance when supported by more data, such as training on sub-dataset-1-c (50 pairs of point clouds).

Table 1. Quantitative results of different methods on Urb3DCD. E-threshold refers to an empirical threshold and Opening represents post-processing with an opening filter. The mIoU over change categories are given (%). The best results on each sub-dataset are shown in bold.

Level of Output	Methods	Sub-Dataset-1-b	Sub-Dataset-2	Sub-Dataset-3	Sub-Dataset-4	Sub-Dataset-5
		ALS Low Res	ALS High Res	ALS High Noise	Photogrammetry	Multi Sensor
2D pixel level	DSMd + E-threshold [56]	36.89	50.72	29.97	30.37	33.16
	DSMd + Otsu [56,59]	59.14	66.96	52.67	54.77	52.61
	DSMd + Otsu + Opening [56,59,60]	71.92	79.09	70.34	76.89	70.99
2D patch level	DSMd + E-threshold [56]	34.59	48.78	29.90	30.76	31.86
	DSMd + Otsu [56,59]	60.40	75.23	56.59	71.00	55.02
	DSMd + Otsu + Opening [56,59,60]	80.22	86.62	80.71	86.83	79.59
	FFN [48]	75.47	79.66	76.37	79.95	76.77
	Siamese CNN [47]	64.16	72.18	71.36	69.50	67.41
3D point level	C2C/(Binary) [57]	-/(49.59)	-/(61.65)	-/(49.22)	-/(54.75)	-/(49.76)
	M3C2 [58]	29.87	53.73	38.72	35.01	37.78
	RF (with SF) [14]	63.41	70.12	58.83	68.87	63.64
	PBFormer (Ours)	**85.05**	**87.14**	**81.77**	**87.34**	**82.60**

Table 2. Quantitative results of different methods on sub-dataset-1-a–c. The mIoU over change categories are given (%). The best results on each sub-dataset are shown in bold.

Methods	Sub-Dataset-1 ALS Low Res		
	a	b	c
FFN [48]	58.43	75.47	77.28
Siamese CNN [47]	55.07	64.16	66.58
RF (with SF) [14]	**61.34**	63.41	67.10
PBFormer (Ours)	57.29	**85.05**	**88.10**

4.3. Ablation Studies

In this section, we first compared the different positional encoding strategies to illustrate the necessity of our proposed SPE. Then, we analyzed the BT decoder's effectiveness by training ablated networks with other feature fusion modules. All ablated networks were trained on the sub-dataset-1-b and evaluated on the test set in sub-dataset-1.

Expecting to evaluate the full network (PBFormer), we conducted eight ablation experiments: (1) Removing SPE and replacing BT by differencing. After removing SPE and BT, we used a pair of sharing transformer encoders of ViT, without position embedding, as Siamese encoders and differencing as the method of feature fusion. (2) Replacing SPE with learned positional encoding. Compared to (1), we added widely used learned 1D positional embeddings to the point embedding sequence in the transformer encoder of ViT. (3) Applying SPE and differencing. In this ablation experiment, SPE was added to the ablated network in (1). (4) Removing SPE and replacing differencing by MLPs. In this ablation experiment, differencing was replaced, and MLPs were considered as the feature fusion module. (5) Applying learned positional encoding and MLPs. We added learned 1D positional embeddings to the ablated network in (4). (6) Applying SPE and MLPs. SPE was added to the ablated network in (4). (7) Removing SPE and applying BT decoder only. In this ablation experiment, MLPs were replaced by our BT decoder. (8) Applying learned 1D positional embeddings and the BT decoder. In the last ablation experiment, learned 1D positional encoding was added to the ablated network in (7).

Table 3 shows the mIoU scores in all ablation experiments. Figure 7 shows the comparative results obtained by varying the positional encoding strategy, and Figure 8 shows how varying feature fusion modules affect the results.

Table 3. Ablation studies of the contribution of SPE and BT decoder. The best results are shown in bold.

	mIoU (%)
(1) None Positional Encoding + Differencing	71.02
(2) Learned Positional Encoding + Differencing	73.57
(3) SPE + Differencing	80.68
(4) None Positional Encoding + MLPs	74.48
(5) Learned Positional Encoding + MLPs	76.53
(6) SPE + MLPs	82.25
(7) None Positional Encoding + BT	80.76
(8) Learned Positional Encoding + BT	82.12
(9) SPE + BT (The Full Network)	**85.05**

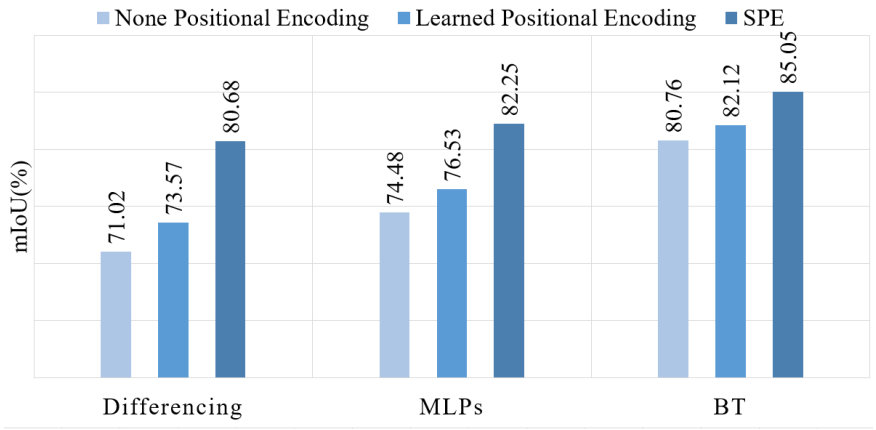


Figure 7. Comparison of different positional encoding strategies.

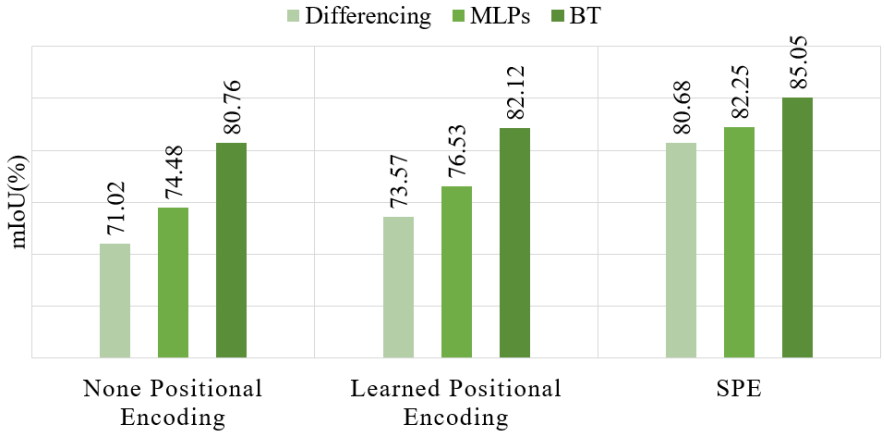


Figure 8. Comparison of different feature fusion modules.

It can be seen that transformer networks have a strong feature extraction ability, which is the prerequisite for the design of PBFormer. Even using only the basic transformer encoder, the mIoU score can easily exceed 70. Moreover, positional embeddings facilitate transformer networks learning point sequence, and a widely used method of positional encoding can improve the mIoU score. In addition, our proposed PT encoder with SPE has one of the most significant impacts on performance. Regardless of the type of feature fusion module used, mIoU scores above 80 demonstrate that SPE is necessary for transformer networks to effectively learn spatial structure information from 3D point clouds.

Moreover, the replacement of the feature fusion modules shows that the BT decoder has another significant impact on performance. The mIoU score also exceeds 80 when only the BT decoder is used in PBFormer. Similarly, based on three positional encoding

strategies, the mIoU scores over 80 prove that the BT decoder could effectively learn the difference between bitemporal point clouds.

Finally, the complete model was evaluated, and it achieved superior performance. The role of the SPE and BT decoder and how they cooperate with each other to achieve the best results was demonstrated in the above ablation experiments.

5. Discussion

This section provides a discussion to explain the previous experimental results. The discussion includes further elaboration of the experimental results and the influence analysis of the experimental dataset.

In this paper, PBFormer was compared with other methods regarding the Urb3DCD benchmark. In addition to the better results achieved with our network, several issues remain to be addressed. With sufficient prior information, C2C is still a method worth considering due to its high efficiency. However, due to its over-dependence on the threshold and its ignoring of the local geometric structure, C2C usually fails to achieve satisfactory results. For a better visual effect, we tried to blur the threshold and obtained some faded areas in Figures 5 and 6, but the result was not perfect. Although the centers of the changed regions can be distinguished, the boundaries between the changed and unchanged regions are vaguer. The result of M3C2 is also unsatisfactory, both visually and qualitatively. The visual results of M3C2 seem to be smoothed because all corresponding points of the points distributed on the edge are not found by M3C2, and they remain in the gray area. RF (with SF) is more reliable than C2C and M3C2, which are relatively sensitive. While all three belong to the one-stage strategy, the quantitative and visual result of RF (with SF) reflects the gap between machine learning-based and traditional methods. A possible reason for the poor results of M3C2 and RF (with SF) is the lower resolution of the dataset. Even for the sub-dataset-2, the resolution is about 10 points/m^2. In this case, where the detection (LODetection) level may exceed the magnitude change, M3C2 applied to detect changes at the centimetric scale will perform less well on a dataset at the metric scale. This could explain the less adequate result of RF (with SF) on sub-dataset-1 (0.5 point/m^2) and the improvement in sub-dataset-2. PBFormer is also affected by the resolution, and point clouds with a higher resolution help PBFormer to achieve better CD quality. However, when processing sparse point clouds, PBFormer could still obtain enough points to learn the characteristics between two inputs by kNN, which is part of the reason why PBFormer works better than other methods.

The establishment of the Urb3DCD benchmark is a pioneering work that provides a dataset with pointwise annotations for training deep learning-based models. However, the scenario in Urb3DCD is simplified, where only ground data appear (with the exception of buildings), and there are few occluded building point clouds. Due to the gap between the dataset and the real world, the generalization ability of PBFormer might be lower than expected. In addition, the categories of changes only include 'new buildings' and 'demolished buildings', which may cause PBFormer to merely learn whether there has been a change, but not the semantics of the point clouds. Moreover, a problem that cannot be ignored and avoided is the data imbalance. In the data used for CD, there are usually fewer changed areas with respect to unchanged areas, which is unfavorable for PBFormer, which is designed based on the assumption that the distribution of each class of samples is uniform. Finally, it should be noted that PBFormer may have poor performance at the boundaries between changed and unchanged objects, especially when the noise is high. PBFormer is a network that outputs pointwise changes. If the edges of buildings are blurred by the high noise, PBFormer would not adequately understand the structure of buildings.

6. Conclusions

In this paper, an end-to-end Siamese transformer network named PBFormer is proposed for 3D CD on urban point clouds. Unlike most existing algorithms that preprocess 3D point clouds into other data types, 3D point clouds are fed directly into PBFormer to

obtain pointwise changes. PT encoder and BT decoder are two parts of PBFormer. PT encoder is used to extract geometric structure features from point clouds, and BT decoder can model the spatiotemporal sequence features and learn the difference between bitemporal point clouds. Extensive experiments on public datasets demonstrate the effectiveness, exceptional performance, and robustness of our method, which could be applied to deformation detection, morphological change estimation, and other 3D CD tasks. Meanwhile more efficient transformer structures for real-time CD in large-scale urban scenes could be studied, based on our network.

Author Contributions: Conceptualization, M.H., J.S. and Y.W.; methodology, M.H.; software, M.H.; validation, M.H.; formal analysis, M.H. and J.S.; investigation, M.H.; resources, J.S. and X.W.; data curation, M.H.; writing—original draft preparation, M.H.; writing—review and editing, M.H., J.S. and Y.W.; visualization, M.H.; supervision, J.S. and X.W.; project administration, X.W.; funding acquisition, J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partly funded by the National Key Laboratory on Electromagnetic Environmental Effects and Electro-optical Engineering under Grant KY3240020001.

Data Availability Statement: The data used in this study are available on IEEE Dataport at the following link http://ieee-dataport.org/open-access/urb3dcd-urban-point-clouds-simulated-dataset-3d-change-detection.

Acknowledgments: The authors sincerely thank the researchers for providing the Urb3DCD datasets and the anonymous reviewers for their valuable comments and suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, Y.; Hong, D.; Sha, J.; Gao, L.; Liu, L.; Zhang, Y.; Rong, X. Spectral–Spatial–Temporal Transformers for Hyperspectral Image Change Detection. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–14. [CrossRef]
2. Yuan, Y.; Lv, H.; Lu, X. Semi-supervised change detection method for multi-temporal hyperspectral images. *Neurocomputing* **2015**, *148*, 363–375. [CrossRef]
3. Lei, J.; Li, M.; Xie, W.; Li, Y.; Jia, X. Spectral mapping with adversarial learning for unsupervised hyperspectral change detection. *Neurocomputing* **2021**, *465*, 71–83. [CrossRef]
4. Bu, S.; Li, Q.; Han, P.; Leng, P.; Li, K. Mask-CDNet: A mask based pixel change detection network. *Neurocomputing* **2020**, *378*, 166–178. [CrossRef]
5. Qin, R.; Tian, J.; Reinartz, P. 3D change detection—Approaches and applications. *ISPRS J. Photogramm. Remote Sens.* **2016**, *122*, 41–56. [CrossRef]
6. Kharroubi, A.; Poux, F.; Ballouch, Z.; Hajji, R.; Billen, R. Three Dimensional Change Detection Using Point Clouds: A Review. *Geomatics* **2022**, *2*, 457–485. [CrossRef]
7. Xu, H.; Cheng, L.; Li, M.; Chen, Y.; Zhong, L. Using Octrees to Detect Changes to Buildings and Trees in the Urban Environment from Airborne LiDAR Data. *Remote Sens.* **2015**, *7*, 9682–9704. [CrossRef]
8. Du, S.; Zhang, Y.; Qin, R.; Yang, Z.; Zou, Z.; Tang, Y.; Fan, C. Building Change Detection Using Old Aerial Images and New LiDAR Data. *Remote Sens.* **2016**, *8*, 1030. [CrossRef]
9. Ku, T.; Galanakis, S.; Boom, B.; Veltkamp, R.C.; Bangera, D.; Gangisetty, S.; Stagakis, N.; Arvanitis, G.; Moustakas, K. SHREC 2021: 3D point cloud change detection for street scenes. *Comp. Graph.* **2021**, *99*, 192–200. [CrossRef]
10. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
11. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D. Going Deeper With Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
12. Awrangjeb, M.; Fraser, C.S.; Lu, G. Building Change Detection from Lidar Point Cloud Data Based on Connected Component Analysis. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2015**, *II-3/W5*, 393–400. [CrossRef]
13. Xu, S.; Vosselman, G.; Oude Elberink, S. Detection and Classification of Changes in Buildings from Airborne Laser Scanning Data. *Remote Sens.* **2015**, *7*, 17051–17076. [CrossRef]
14. Tran, T.H.G.; Ressl, C.; Pfeifer, N. Integrated Change Detection and Classification in Urban Areas Based on Airborne Laser Scanning Point Clouds. *Sensors* **2018**, *18*, 448. [CrossRef]
15. Guo, Y.; Wang, H.; Hu, Q.; Liu, H.; Liu, L.; Bennamoun, M. Deep Learning for 3D Point Clouds: A Survey. *IEEE Trans. Patt. Anal. Mach. Inte.* **2021**, *43*, 43384364. [CrossRef] [PubMed]
16. Qi, C.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 2017.

17. Yu, X.; Tang, L.; Rao, Y.; Huang, T.; Zhou, J.; Lu, J. Point-BERT: Pre-Training 3D Point Cloud Transformers With Masked Point Modeling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022.
18. Qi, C.; Yi, L.; Su, H.; Guibas, L. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.
19. Shao, Y.; Tong, G.; Peng, H. Mining local geometric structure for large-scale 3D point clouds semantic segmentation. *Neurocomputing* **2022**, *500*, 191–202. [CrossRef]
20. Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020.
21. Yang, Z.; Sun, Y.; Liu, S.; Jia, J. 3DSSD: Point-Based 3D Single Stage Object Detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020.
22. Giancola, S.; Zarzar, J.; Ghanem, B. Leveraging Shape Completion for 3D Siamese Tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
23. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic Graph CNN for Learning on Point Clouds. *ACM ToG* **2019**, *38*, 1–12. [CrossRef]
24. Gojcic, Z.; Zhou, C.; Wieser, A. F2S3: Robustified determination of 3D displacement vector fields using deep learning. *J. Appl. Geod.* **2020**, *14*, 177–189. [CrossRef]
25. Gojcic, Z.; Schmid, L.; Wieser, A. Dense 3D displacement vector fields for point cloud-based landslide monitoring. *Landslides* **2021**, *18*, 3821–3832. [CrossRef]
26. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.
27. Lin, T.; Wang, Y.; Liu, X.; Qiu, X. A survey of transformers. *AI Open* **2022**, *3*, 111–132. [CrossRef]
28. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Peep Pidirectional Pransformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
29. Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.V.; Salakhutdinov, R. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. *arXiv* **2019**, arXiv:1901.02860.
30. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16 $\times$ 16 words: Transformer for image recognition at scale. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 4 May 2021.
31. Wang, W.; Yao, L.; Chen, L.; Lin, B.; Cai, D.; He, X.; Liu, W. CrossFormer: A Versatile Vision Transformer Hinging on Cross-scale Attention. *arXiv* **2021**, arXiv:2108.00154.
32. Chen, C.; Fan, Q.; Panda, R. CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QU, Canada, 10–17 October 2021.
33. He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; Girshick, R. Masked Autoencoders Are Scalable Vision Learners. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022.
34. De Gélis, I.; Lefèvre, S.; Corpetti, T. Change Detection in Urban Point Clouds: An Experimental Comparison with Simulated 3D Datasets. *Remote Sens.* **2021**, *13*, 2629. [CrossRef]
35. Thomas, H.; Qi, C.R.; Deschaud, J.; Marcotegui, B.; Goulette, F.; Guibas, L.J. KPConv: Flexible and Deformable Convolution for Point Clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
36. Xu, M.; Ding, R.; Zhao, H.; Qi, X. PAConv: Position Adaptive Convolution With Dynamic Kernel Assembling on Point. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Online, 19–25 June 2021.
37. Guo, M.; Cai, J.; Liu, Z.; Mu, T.; Martin, R.; Hu, S. PCT: Point cloud transformer. *Comp. Visual Media* **2021**, *7*, 187–199. [CrossRef]
38. Zhao, H.; Jiang, L.; Jia, J.; Torr, P.; Koltun, V. Point Transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QU, Canada, 10–17 October 2021.
39. Zhang, R.; Guo, Z.; Gao, P.; Fang, R.; Zhao, B.; Wang, D. Point-M2AE: Multi-scale Masked Autoencoders for Hierarchical Point Cloud Pre-training. *arXiv* **2022**, arXiv:2205.14401.
40. Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; Shah, R. Signature Verification using a "Siamese" Time Delay Neural Network. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Denver, CO, USA, 29 November–2 December 1993.
41. Hoffer, E.; Ailon, N. Deep Metric Learning Using Triplet Network. In Proceedings of the Similarity-Based Pattern Recognition (SIMBAD), Copenhagen, Denmark, 12–14 October 2015.
42. Zagoruyko, S.; Komodakis, N. Learning to Compare Image Patches via Convolutional Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
43. Chopra, S.; Hadsell, R.; LeCun, Y. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA, 20–26 June 2005.

44. Bertinetto, L.; Valmadre, J.; Henriques, J.F.; Vedaldi, A.; Torr, P.H.S. Fully-Convolutional Siamese Networks for Object Tracking. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016.
45. Held, D.; Thrun, S.; Savarese, S. Learning to Track at 100 FPS with Deep Regression Networks. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016.
46. Wang, Z.; Peng, C.; Zhang, Y.; Wang, N.; Luo, L. Fully convolutional siamese networks based change detection for optical aerial images with focal contrastive loss. *Neurocomputing* **2021**, *457*, 155–167. [CrossRef]
47. Zhang, Z.; Vosselman, G.; Gerke, M.; Tuia, D.; Yang, M. Change Detection between Multimodal Remote Sensing Data Using Siamese CNN. *arXiv* **2018**, arXiv:1807.09562.
48. Zhang, Z.; Vosselman, G.; Gerke, M.; Persello, C.; Tuia, D.; Yang, M. Detecting Building Changes between Airborne Laser Scanning and Photogrammetric Data. *Remote Sens.* **2019**, *11*, 2417. [CrossRef]
49. Li, Z.; Chen, Z.; Yang, F.; Li, W.; Zhu, Y.; Zhao, C.; Deng, R.; Wu, L.; Zhao, R.; Tang, M.; et al. MST: Masked Self-Supervised Transformer for Visual Representation. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Online, 6–14 December 2021.
50. Doersch, C.; Gupta, A.; Zisserman, A. CrossTransformers: Spatially-aware few-shot transfer. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Vancouver, BC, Canada, 6–12 December 2020.
51. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Vancouver, BC, Canada, 6–12 December 2020.
52. Guo, G.; Wang, H.; Bell, D.; Bi, Y.; Greer, K. KNN Model-Based Approach in Classification. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*; Springer: Berlin/Heidelberg, Germany, 2003.
53. Zhang, S.; Li, X.; Zong, M.; Zhu, X.; Cheng, D. Learning k for kNN Classification. *ACM Trans. Intell. Syst. Technol.* **2017**, *8*, 1–19. [CrossRef]
54. Rusu, R.B.; Cousins, S. 3D is here: Point Cloud Library (PCL). In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011.
55. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional Sequence to Sequence Learning. In Proceedings of the 34th International Conference on Machine Learning (PMLR), Sydney, Australia, 6–11 August 2017.
56. Murakami, H.; Nakagawa, K.; Hasegawa, H.; Shibata, T.; Iwanami, E. Change detection of buildings using an airborne laser scanner. *ISPRS J. Photogramm. Remote Sens.* **1999**, *54*, 148–152. [CrossRef]
57. Girardeau-Montaut, D.; Roux, M.; Marc, R.; Thibault, G. Change Detection on Points Cloud Data Acquired with A Ground Laser Scanner. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2005**, *36*, 30–35.
58. Lague, D.; Brodu, N.; Leroux, J. Accurate 3D comparison of complex topography with terrestrial laser scanner: Application to the Rangitikei canyon (N-Z). *ISPRS J. Photogramm. Remote Sens.* **2013**, *82*, 10–26. [CrossRef]
59. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man. Cyber.* **1979**, *9*, 62–66. [CrossRef]
60. Stal, C.; Tack, F.; de Maeyer, P.; de Wulf, A.; Goossens, R. Airborne photogrammetry and lidar for DSM extraction and 3D change detection over an urban area—A comparative study. *Int. J. Remote Sens.* **2012**, *34*, 1087–1110. [CrossRef]