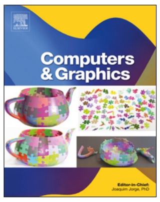
Computers & Graphics 115 (2023) 35–42

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag



Special Section on 3DOR 2023

SHREC 2023: Point cloud change detection for city scenes

Yang Gao [a], Honglin Yuan [a,*], Tao Ku [b], Remco C. Veltkamp [b], Georgios Zamanakos [c], Lazaros Tsochatzidis [c], Angelos Amanatiadis [c], Ioannis Pratikakis [c,e], Aliki Panou [d], Ioannis Romanelis [d], Vlassis Fotis [d], Gerasimos Arvanitis [d], Konstantinos Moustakas [d]

[a] Nanjing University of Information Science and Technology, China
[b] Utrecht University, Netherlands
[c] Democritus University of Thrace, Greece
[d] University of Patras, Greece
[e] Athena Research Center, Xanthi, Greece

A R T I C L E   I N F O

Article history:
Received 27 March 2023
Received in revised form 18 June 2023
Accepted 21 June 2023
Available online 25 June 2023

Keywords:
Point cloud change detection
Retrieval models and ranking
SHREC

A B S T R A C T

Localization and navigation are the two most important tasks for mobile robots, which require an up-to-date and accurate map. However, to detect map changes from crowdsourced data is a challenging task, especially from billions of points collected by 3D acquisition devices. Collecting 3D data often requires expensive data acquisition equipment and there are limited data sources to evaluate point cloud change detection. To address these issues, in this Shape Retrieval Challenge (SHREC) track, we provide a city-scene dataset with real and synthesized data to detect 3D point cloud change. The dataset consists of 866 pairs of object changes from 78 city-scene 3D point clouds collected by LiDAR and 845 pairs of object changes from 100 city-scene 3D point clouds generated by a high-fidelity simulator.

We compare three methods on this benchmark. Evaluation results show that data-driven methods are the current trend in 3D point cloud change detection. Besides, the siamese network architecture is helpful to detect changes in our dataset. We hope this benchmark and comparative evaluation results will further enrich and boost the research of point cloud change detection and its applications.

1. Introduction

Point cloud change detection is drawing more and more attention with the quick development of autonomous ground and unnamed aerial vehicles over the last few years. These autonomous vehicles are commonly used in a variety of tasks, such as forest monitoring [1], urban planning [2], logistics delivery [3] and remote sensing [4]. To make sure these tasks can be completed successfully, an accurate and up-to-date map is required, which makes point cloud change detection crucial.

The rapid development of 3D acquisition devices enables us to collect billions of points in a few hours. However, how to analyze quantitative collected data is a challenging task, especially for 3D point cloud change detection. The general objective of change detection from 3D point clouds is to detect spatial changes between two point clouds obtained from the same site at different times and then to identify the change types [5,6]. Classical approaches directly process raw crowdsourced data to detect changes [7,8]. However, these methods are time-consuming and are not able to distinguish the unknown and empty areas.

On the other hand, a different class of methods relies on classification-based comparison to achieve change detection [9–11]. Unlike direct comparison approaches, these methods first cluster and classify points into different classes and then a registration process is introduced to detect 3D changes. However, these methods impose a huge computational complexity. Thus, current methods have different strengths and weaknesses, depending on data processing and scene contents. More and more new algorithms, aiming for 3D change detection have been published. However, it is unclear whether these methods are suitable for complexity scene containing rich urban objects with varying light, shapes and textures, for the lack of available datasets.

Change detection plays an important role in robot navigation, remote sensing and logistics delivery, for such tasks require an accurate and up-to-date map with rich 3D information. Thus, collecting 3D data becomes crucial. However, current data equipment is expensive with costs in the thousands to millions of dollars. For example, expensive sensors including airborne laser scanners (ALS), mobile laser scanners (MLS) and terrestrial laser scanners (TLS) are often used to collect point clouds for change

* Corresponding author.
E-mail addresses: hlyuan@nuist.edu.cn (H. Yuan), gzamanak@ee.duth.gr (G. Zamanakos), arvanitis@ece.upatras.gr (G. Arvanitis).

https://doi.org/10.1016/j.cag.2023.06.025
0097-8493/© 2023 Elsevier Ltd. All rights reserved.

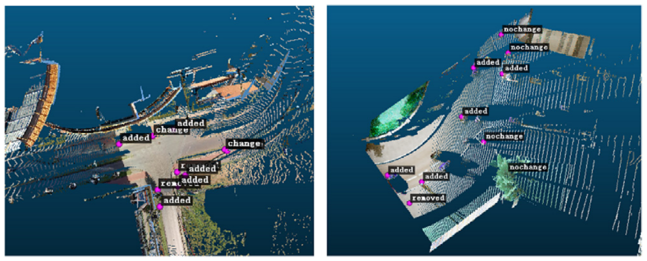


Fig. 1. Examples of the dataset showing city scenes with a variety of changed urban objects: left point cloud captured by LiDAR and the right point cloud generated by the simulator.

detection [12–15]. Alternative solutions rely on structure from motion (SfM) and multi-view stereo (MVS) to generate point clouds [16,17]. However, these methods are time-consuming and require several days to generate point clouds for a medium-sized urban environment [18].

The limited number of available datasets is also a barrier to Point cloud change detection [19]. SHREC 2021 point cloud change detection dataset [20] only provides point clouds for real urban scenes while datasets provided by [21] only focus on simulated scenes. To the best of our knowledge, there are no available datasets which take advantages of both real and synthesized data.

To address these issues, we propose the point cloud change detection for city scenes benchmark and compare different methods on the proposed benchmark. In this benchmark, we provide a 3D point cloud dataset consisting of real captured 3D data by a mobile laser scanner (MLS) and synthesized 3D data generated by the high-quality simulator AirSim built on Unreal Engine 4 [22].

The objects in our datasets cover a variety of shapes, sizes, and textures, such as cars, garbage bins and lights. Our main contributions are summarized as follows:

• A representative dataset providing high-quality annotated 3D point clouds, which contains rich urban objects with varying light, shapes and textures, and covers a wide range of change detection challenges. The dataset fills the vacancy in the point cloud change detection task.

• A comprehensive evaluation of point cloud change detection approaches. We organize the Shape Retrieval Challenge (SHREC) benchmark on point cloud change detection and use different evaluation metrics to compare the proposed methods based on our datasets. Evaluation results indicate that approaches that fully exploit the geometric features are more robust for point cloud change detection for minority classes.

2. Benchmark

Our point cloud change detection benchmark provides high-quality annotated 3D point clouds for complex city scenes containing urban objects of varying shapes and textures. To facilitate data-driven approaches, we also synthesize a large number of high-quality 3D point clouds using the simulator AirSim built on Unreal Engine 4.

2.1. Dataset

We use both real-world data and data generated from simulation for the point cloud change detection retrieval. As shown in Fig. 1, our dataset contains a variety of urban objects in complex city scenes.

The real-world dataset describes the street scenes in 2016 and 2020 for Schiedam, a city in the Netherlands. It is captured by LiDAR sensors which are mounted in a car. In this dataset, we focus on street furniture including road-sings, advertisements, garbage bins and so on. It contains 78 annotated street-scene 3D point cloud pairs in the year 2016 and 2020, respectively, and 866 labeled urban objects with different change types. Labeling is done through manual inspection.

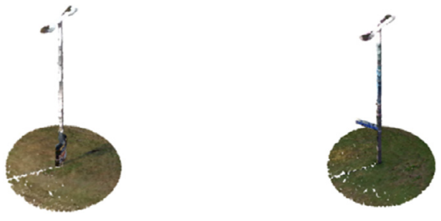


Fig. 2. An example of "*change*" type.

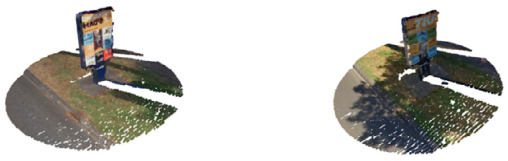


Fig. 3. An example of "*color_change*" type.

Our aim is to build a point cloud change detection dataset containing rich urban objects with varying viewpoints, shapes and textures which are the limitation for real captured datasets. Inspired by the low cost of producing very large-scale synthetic datasets with complete and accurate ground-truth information, as well as the recent successes of synthetic data for training change detection systems, we generate a synthesized dataset with a high-quality simulator.

We generate the synthesized dataset using simulated scenes in Unreal Engine 4 with the AirSim plugin. The synthesized dataset contains 100 annotated street-scene 3D point cloud pairs generated at two different time points and 845 labeled urban objects with different change types. The synthetic scenes we use are offered by Unreal Engine 4 which provides many cutting edge graphics features such as photometric lights, ray traced distance field shadows and so on. Thus, the synthetic scenes are able to mimic the real ones and model real-world complex objects such as electric poles, trees and buildings with shadows, reflections. Labeling is also done through manual inspection.

Compared with our real-world dataset, the urban objects in our synthesized dataset have richer shapes, textures and light conditions. We combine the synthesized and real-world data to build the final dataset, which covers a wide range of change detection challenges.

To define and distinguish the term "change" in the context of various scenarios, we follow the definition provided by SHREC 2021 [20]. The change types are defined as:

• *nochange* is used in the case where there is no significant change between the two scans of the same urban object.

• *added* is used in the case where an object which does not exist in the first scan appears in the second scan.

• *removed* is used in the case where object that exists in the first scan disappears in the second scan.

• *change* is used in the case where there exists some geometric change for the urban object. For example, a small handle is added to the urban object, as shown in Fig. 2.

• *color_change* is used in the case where the object has no geometric change but has a color change. For example, compared with the first scan the color of the advertisement is changed as shown in Fig. 3.

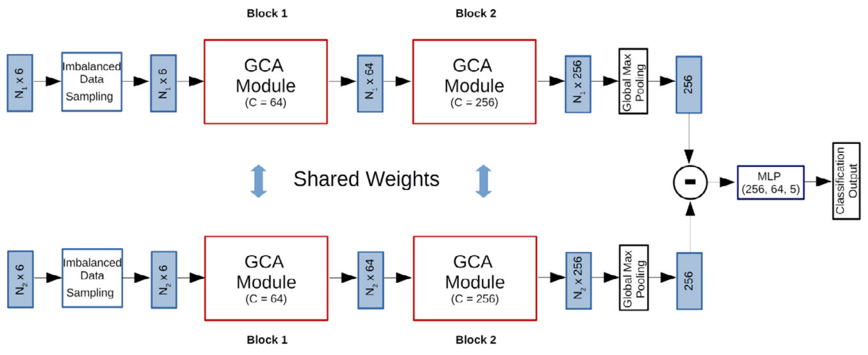


Fig. 4. The network architecture of the proposed SiamGCN-GCA method.

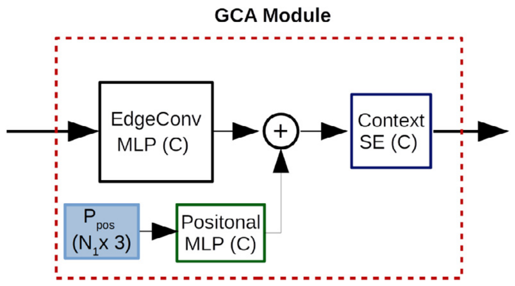


Fig. 5. The architecture of the proposed GCA module.

In our dataset, each 3D point cloud contains labeled urban objects with different change types. The labeled points have been placed on or at the base of the objects which are saved in csv files. Each point cloud and its corresponding csv file share the same name. So the same identifier can be used to find the point cloud and the corresponding change types of different objects in it. To make our dataset suitable for learning-based approaches, the dataset is split into training and test sets with ratios of 80 and 20.

2.2. Evaluation metrics

In this benchmark, we aim to find and classify changes for urban objects between two different states of complex city scenes. We provide scene-level 3D point clouds generated at two different time points and the corresponding center of urban objects.

As point cloud change detection can be considered as classification problem. Therefore, the standard classification metrics including Overall Accuracy (OA) [23] and mean Intersection over Union (mIoU) metrics [24] are introduced to evaluate the performance of different proposed methods in our benchmark.

The Overall Accuracy represents the proportion of correctly classified samples. The OA is calculated by dividing the number of correctly classified objects M by the total number of objects N with different change types.

$$\mathrm{OA} = \frac{M}{N} \tag{1}$$

The Intersection over Union (IoU) measures the overlap between predicted results and the ground truth

$$\mathrm{IoU} = \frac{TP}{PN + GN - TP}, \tag{2}$$

where TP, GT and PN are the number of correct predictions for a certain class, the number of objects predicted to the certain class and the ground truth number of objects to the certain class, respectively.

3. Methods

All the proposed methods are described in the following subsections.

3.1. SiamGCN-GCA: 3D point cloud change detection with siamese and geometry context aware (GCA) attention

This method is contributed by authors [Georgios Zamanakos, Lazaros Tsochatzidis, Angelos Amanatiadis and Ioannis Pratikakis]. In this work, a novel Geometry Context Aware (GCA) attention mechanism is proposed, that could be used to enhance the performance of Graph Convolutional Networks for the task of 3D point cloud change detection. SiamGCN [20] is used as a baseline model, which is modified with the proposed GCA mechanism. The resulting SiamGCN-GCA network follows a siamese architecture to process the two input point clouds from two different time periods. The code of SiamGCN-GCA approach is publicly available at [25].

3.1.1. Preprocessing

The provided dataset is in the form of large point cloud scenes with the center coordinates of areas of interest provided along with a classification label. It consists of two domains including a set of synthetic scenes and a set of real LiDAR data. For both synthetic and real sets, the preprocessing steps are the same.

As a preprocessing step, two object pairs are extracted around each center coordinate. This is accomplished by extracting a cylindrical area around each center coordinate with a distance of 3 m for all samples. The choice of distance is made after experimental work.

The resulting object pairs are centered, but the coordinates of the points are not normalized. As a result, the true geometry of the scene is kept intact.

Since data are class-imbalanced, SiamGCN-GCA adopts the same strategy as in SiamGCN [20]. SiamGCN-GCA uses an imbalanced data sampler to duplicate samples in the minority class. By doing so, it is ensured that the samples of each class have the same probability of being used during training.

3.1.2. Network architecture

As shown in Fig. 4, the proposed SiamGCN-GCA consists of two branches, one for each object scene. The two branches share the same weights. Each branch is designed to extract a one dimensional feature vector. The resulting two one dimensional features vectors are subtracted and their difference is used as input to a MLP, to obtain the final classification output.

As in SiamGCN, the Dynamic Graph CNN (Edge convolution [26]) is used as feature extractor.

The GCA attention mechanism consists of two distinct modules as shown in Fig. 5 including the positional sub-network and the contextual sub-network.

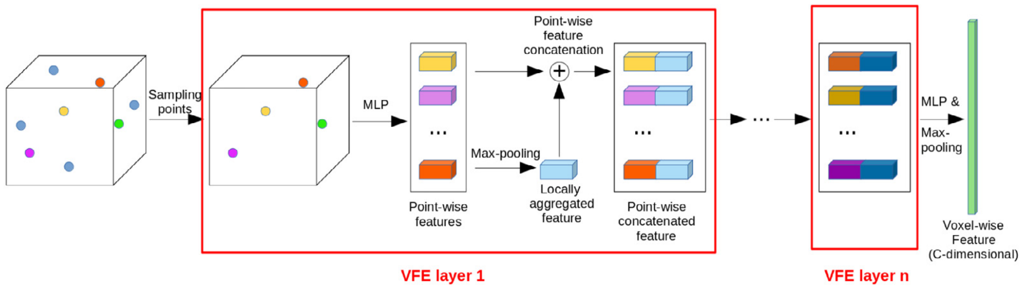

Sampling points
MLP
Point-wise features
Max-pooling
Locally aggregated feature
Point-wise feature concatenation
Point-wise concatenated feature
VFE layer 1
VFE layer n
MLP & Max-pooling
Voxel-wise Feature (C-dimensional)

Fig. 6. Block diagram of the Voxel Feature Encoding (VFE).

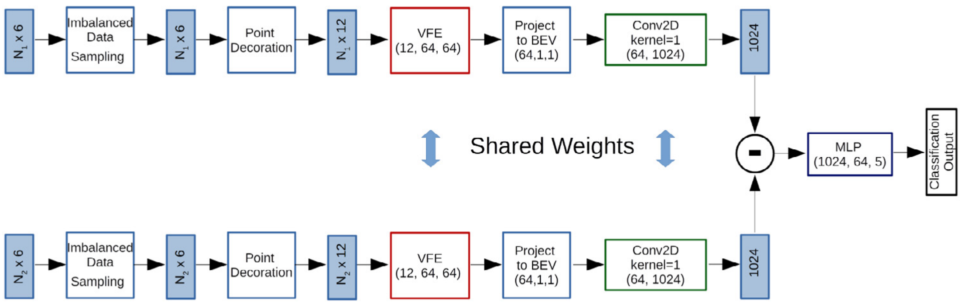


Fig. 7. The network architecture of the proposed SiamVFE method.

The Positional sub-network aims in encoding the geometry of the scene. Given a point cloud scene P consisting of N points, let $P_{pos} \in R^{N\times 3}$ be the $[x, y, z]$ 3D coordinates of the N points. The positional sub-network receives as an input the P_{pos} and projects it into a higher dimension. Mathematically,

$$PE_i = MLP_i(P_{pos}), \tag{3}$$

where PE is the positional encoding, MLP is a multilayer perceptron architecture and i is the block number. The resulting high dimensional PE is then added to the feature vector from the Edge convolution sub-network via a summation operation.

The Contextual sub-network aims in extracting a global context of each scene by re-weighting the already learned features of each point. Let $P_i^f \in R^{B\times N\times C}$ be the resulting feature vector of each scene, where B is the batch number, N is the number of points, C is the number of feature channels and i is the block number. The contextual sub-network first performs a global average pooling across all N points to extract the global average feature vector of each scene. Then, via a Squeeze and Excitation (SE) architecture [27], the global average feature vector is transformed into an attention vector $A_i \in R^{B\times C}$. Mathematically,

$$A_i = \sigma(W_2\delta(W_1(GAP(P_i^f)))), \tag{4}$$

where σ, δ are the sigmoid and ReLU activation functions respectively, W_1, W_2 are the linear projections of the squeeze and the excitation operations respectively. GAP is the global average pooling operation.

The resulting A_i is multiplied by P_i^f to perform the feature re-weighting.

3.2. SiamVFE: Point cloud change detection with siamese and the voxel feature encoding architectures

This method is contributed by authors [Georgios Zamanakos, Lazaros Tsochatzidis, Angelos Amanatiadis and Ioannis Pratikakis]. In this work, a deep learning architecture that relies on the Voxel Feature Encoding (VFE) architecture is proposed, for the task of 3D point cloud change detection inspired by the context of LiDAR-based 3D object detection. VFE is used by 3D object detectors, such as VoxelNet [28] and PointPillars [29], for extracting a discriminative feature vector per voxel or pillar, respectively.

VFE is built upon a mini-PointNet structure consisting of linear layers and max pooling, as shown in Fig. 6. Furthermore, due to its computational efficiency, VFE enables the processing of multiple points at the same time. The symmetric function of max pooling is applied on a local neighborhood of points, given the voxel or pillar that they belong to. The code of our approach is publicly available at [25].

3.2.1. Preprocessing

Same as in method SiamGCN-GCA (Section 3.1.1, Page 3).

3.2.2. Network architecture

As shown in Fig. 7, the proposed SiamVFE consists of two branches, one for each object scene. The two branches share the same weights. Each branch is designed to extract a one dimensional feature vector for each scene. The resulting two one dimensional feature vectors are subtracted and their difference is used as an input to a MLP to obtain the final classification output. In SiamVFE, the whole scene was treated as a single pillar that takes all points in a certain radius.

Given a point cloud scene P consisting of N points, let $P = \{p_i = [x_i, y_i, z_i, r_i, g_i, b_i]\} \in R^{N\times 6}$, where $[x_i, y_i, z_i]$ refers to the 3D coordinates and $[r_i, g_i, b_i]$ to the RGB values for each ith point, respectively. The mean coordinates of all points N define the centroid of the point cloud and are denoted as $V = [u_x, u_y, u_z]$. Let $K = [k_x, k_y, k_z]$ be the center coordinates of the scene. Each point is defined by its relative offset from both the centroid and center of the point cloud scene. As a result, each point p_i is now encoded as: $P' = \{p_i' = [x_i, y_i, z_i, x_i-u_x, y_i-u_y, z_i-u_z, x_i-k_x, y_i-k_y, z_i-k_z, r_i, g_i, b_i]\} \in R^{N\times 12}$.

Following steps in SiamVFE pipeline, P' is fed as an input to a VFE for feature extraction. The resulting pillar is projected into a Bird's Eye View (BEV) pseudo-image. This BEV pseudo-image is further processed by a 2D CNN with kernel equal to one.

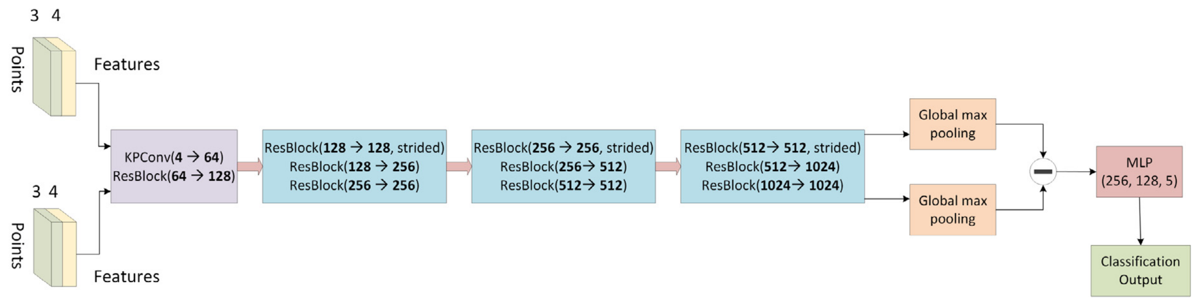


Fig. 8. The network architecture of the proposed SiamKPConv.

3.3. SiamKPConv: 3D point cloud change detection for street scenes

This method is proposed by [Aliki Panou, Ioannis Romanelis, Vlassis Fotis, Gerasimos Arvanitis, and Konstantinos Moustakas]. In this work, a siamese KPConv network to detect changes between two point clouds at different chronological times is proposed. The network architecture is briefly illustrated in Fig. 8. In a nutshell, SiamKPConv leverages Kernel Point Convolution (KPConv) [30], which is a type of convolution that operates directly on raw point clouds, to extract features at different scales. More details about the method will be presented in the following sections. A code implementation of the project is available on GitHub [31].

3.3.1. Data pre-processing

To prepare the data for our network, a fixed-size 3D bounding box with the point of interest at its center is defined and the point clouds that fall within that box are extracted. Specifically, a bounding box with a fixed size of 2.4 m in length and width and a height of 11 m is defined. The dimensions of the bounding box are chosen empirically by observing the datasets and are found to be satisfactory for both datasets. If no points are found within the bounding box, they are replaced with dummy points which are randomly generated points to ensure that the input to SiamKPConv network is always of a consistent size. Additionally, to limit the number of points, 2048 points are selected randomly in each point cloud.

3.3.2. Network architecture

A 4-stage KPConv feature extractor is employed in the siamese architecture. This network uniformly downsamples the input point clouds and learns hierarchical features, enabling the capture of information at different scales. We utilize code base from the open source implementation of [32].

KPConv: The building block in this architecture is KPConv. The point cloud is represented with two matrices: the points $P \in R^{N\times3}$ and the features $F \in R^{N\times D}$. The general point convolution of a matrix F by a kernel g at a point $x \in R^3$ is defined as:

$$(F * g) = \sum_{x_i \in N_x} g(x_i - x) f_i \tag{5}$$

where x_i is a point $P \in R^{N\times3}$ and f_i its corresponding features in $F \in R^{N\times D}$. N_x is defined in [30] as the radius neighborhood of point x. Thus, $N_x = \{\, x_i \in P \mid \|x_i - x\| \le r\}$, where $r \in R$ is the chosen radius. This neighborhood definition ensures robustness to varying densities.

The relative positions of the neighboring points with respect to the center x are defined by: $y_i = x_i - x$. The domain of the definition of g is the ball: $B_r^3 = \{y \in R^3 \mid \|y\| \le r\}$.

Let $\{\widetilde{x_k} \mid k < K\} \subset B_r^3$ be the K kernel points and $\{W_k \mid k < K\} \subset R^{D_{in}\times D_{out}}$ be the associated weight matrices that map features from dimension D_{in} to D_{out}. The kernel function g for any $y_i \in B_r^3$ is defined as:

$$g(y_i) = \sum_{k<K} h(y_i, \widetilde{x_k}) W_k, \tag{6}$$

where h is the correlation between $\widetilde{x_k}$ and y_i and is higher when $\widetilde{x_k}$ is closer to y_i. The correlation is defined as:

$$h(y_i, \widetilde{x_k}) = \max(0, 1 - \frac{\|y_i - \widetilde{x_k}\|}{\sigma}), \tag{7}$$

where σ is the influence distance of the kernel points.

Each stage in the encoder part of the architecture, except for the first one, consists of three convolutional blocks. In all stages except for the first one, the first block is always "strided". This "strided" KPConv operation simulates 2D strided convolutions by downsampling the point cloud, thus reducing the number of points to compute features at different scales. SiamKPConv uses grid subsampling as in [30] to downsample the point clouds. As described in [30], "ResBlocks" are designed like ResNet blocks [33].

The parameter in the KPConv network that matters the most is the initial voxel size for grid subsampling, which is determined based on the dataset density and the level of detail required to detect changes. In the case of the Change3D dataset, the initial voxel size was set to 0.015 m, while for the synthetic dataset, it was set to 0.02 m. The voxel size is doubled in each downsampling step along with the convolution radius, thereby increasing the receptive field of SiamKPConv.

The first KPConv layer is fed with four initial features: the RGB information of point clouds and a constant value of 1. Group normalization with 32 groups is applied after each KPConv layer. The same encoder is used for both point clouds.

The output of the final block is passed through a global max pooling layer to obtain a final feature vector for each point cloud. We then subtract these feature vectors to get a difference vector, which is fed to an MLP with 3 hidden layers for classification. A dropout of 0.5 is used in the first two layers of the MLP network.

To address the challenge of imbalance in both datasets, a weighted random sampler is used that draws more samples from the minority classes during training to ensure that the model was exposed to enough examples of each class.

3.3.3. Implementation details

The network is implemented using PyTorch framework and trained on a NVIDIA GeForce RTX 2060 GPU. Adam optimizer with a weight decay of 10^{-6} is used and minimizes a cross-entropy loss function. The batch size was set to 1. The initial learning rate was set to 10^{-2} and was multiplied by 0.95 every 4 epochs to decrease the learning rate gradually. SiamKPConv uses an 80:20 train/validation split for all the experiments.

Table 1
Quantitative evaluation of the change detection accuracy in terms of classification accuracy and OA.

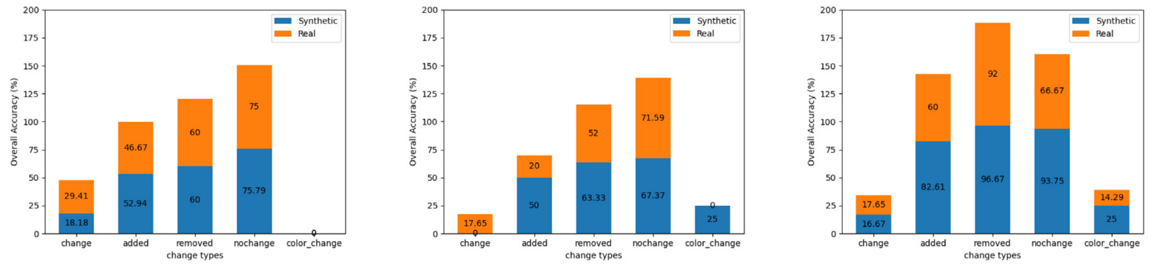
Methods	OA(%)		change		added		removed		nochange		color_change	
	Synthetic	Real	Synthetic	Real	Synthetic	Real	Synthetic	Real	Synthetic	Real	Synthetic	Real
SiamGCN-GCA	61.54	61.18	**18.18**	**29.41**	52.94	46.67	60.00	60.00	75.79	**75.00**	0	0
SiamVFE	56.73	53.95	0	17.65	50.00	20.00	63.33	52.00	67.37	71.59	**25.00**	0
SiamKPConv	**84.83**	**62.34**	16.67	17.65	**82.61**	**60.00**	**96.67**	**92.00**	**93.75**	66.67	**25.00**	**14.29**

Table 2
Quantitative evaluation of the change detection accuracy in terms of IoU and mIoU.

Methods	mIoU(%)		change		added		removed		nochange		color_change	
	Synthetic	Real	Synthetic	Real	Synthetic	Real	Synthetic	Real	Synthetic	Real	Synthetic	Real
SiamGCN-GCA	31.14	29.78	6.25	**18.52**	44.44	31.82	45.00	41.67	60.00	**56.90**	0	0
SiamVFE	27.86	23.16	0	11.11	40.96	14.29	39.58	35.14	51.61	55.26	7.14	0
SiamKPConv	**53.85**	**37.06**	**10.53**	8.82	**77.03**	**47.37**	**78.38**	**67.65**	**83.33**	55.56	**20.00**	**5.88**

Table 3
The number of annotated urban objects of different change types.

Dataset	nochange		removed		added		change		color_change	
	Synthetic	Real	Synthetic	Real	Synthetic	Real	Synthetic	Real	Synthetic	Real
Train set	257 (40.54%)	351 (59.80%)	113 (17.82%)	100 (17.04%)	212 (33.44%)	54 (9.20%)	41 (6.47%)	63 (10.73%)	11 (1.74%)	19 (3.24%)
Test set	96 (45.50%)	90 (58.44%)	30 (14.22%)	25 (16.23%)	69 (32.70%)	15 (9.74%)	12 (5.69%)	17 (11.04%)	4 (1.90%)	7 (4.55%)



(a) SiamGCN-GCA (b) SiamVFE (c) SiamKPConv

Fig. 9. The performance of different methods in terms of OA.

4. Experimental results

Overall performance. Tables 1 and 2 show the evaluation results for all the change classes. We evaluate three approaches including SiamGCN-GCA, SiamVFE and SiamKPConv. We use the Overall Accuracy (OA) and mean Intersection over Union (mIoU) metrics to evaluate these approaches for all classes. We also calculate the classification accuracy and IoU for each class.

From Tables 1 and 2 we can see that SiamKPConv achieves the best performance. For example, SiamKPConv outperforms SiamGCN-GCA and SiamVFE 23.29% and 28.10% in terms of OA, respectively on the synthesized dataset. SiamKPConv outperforms SiamGCN-GCA and SiamVFE 22.71% and 25.99% in terms of mIoU, respectively on the synthesized dataset.

Datasets. In our benchmark, we provide real and synthetic datasets to participants. Both datasets contain five change types and each change type consists of different numbers of urban objects, as shown in Table 3.

From Tables 1 and 2 we can see that compared with the real dataset, all the approaches achieve better performance on the synthetic dataset. This indicates that even though the synthetic dataset provides a variety of labeled urban objects which have richer shapes, textures and light conditions, the real dataset poses more challenges for the change detection. On the other hand, we can see that the evaluated methods achieve similar performance on the synthetic and real datasets, especially for SiamGCN-GCA and SiamVFE. It indicates that our synthetic dataset is able to mimic the complex real city scenes and cover most of point cloud change detection challenges.

Network architecture. The network architecture plays an important role in data-driven approaches. All the evaluated methods in our benchmark are based on Siamese architecture to design their networks, which indicates that Siamese architecture is popular for the change detection task.

Apart from the Siamese architecture, the evaluated methods use different sub-networks. For SiamGCN-GCA, it uses Edge convolution [26]) to extract features. Besides, SiamGCN-GCA introduces Context Aware (GCA) attention mechanism to enhance the performance of Graph Convolutional Networks. Fig. 9(a) shows its performance for each change type. Compared with the other two methods, SiamGCN-GCA has better performance to detect changes for urban objects with "nochange" type on the real dataset.

However, SiamGCN-GCA has poor performance to detect changes for urban objects with "color_change" type on both synthetic and real dataset.

Compared with SiamGCN-GCA and SiamVFE, SiamKPConv adds Kernel Point Convolution (KPConv) to its network. From Tables 1, 2 and Fig. 9 we can see that it achieves the best performance among the evaluated methods, and it outperforms the other approaches on the class of "removed" type in terms of OA. This indicates that KPConv is helpful for point cloud change detection.

Class imbalance. Class imbalance is a common problem in deep learning, especially in classification problems. It occurs when there are many more instances of some classes than others. In our synthetic and real datasets, 40.54% and 59.13% urban objects are classified as "nochange" while 1.74% and 3.65% urban objects are classified as "color_change".

In such cases, classifiers are more likely to be overwhelmed by the large classes and ignore the small ones. To address this problem, SiamGCN-GCA and SiamVFE uses an imbalanced data sampler to duplicate samples in the minority class. Compared with SiamKPConv which does not adopt the imbalanced data sampler, SiamGCN-GCA achieves better performance on the class of "change" type in term of OA as shown in Table 1.

Compared with duplicating samples in the minority class, adding samples is able to improve the performance significantly. From Tables 1 and 2 we can see that when the number of labeled urban objects of the same class is similar on both synthetic and real datasets, the evaluated methods have similar performance. However, when the number of labeled urban objects has big differences, the corresponding performances also change significantly.

For example, 33.44% of urban objects are labeled as "added" on the synthetic training dataset while only 9.55% of urban objects have the same labels on the real training dataset. SiamVFE trained on the synthetic dataset achieves 40.96% IoU while SiamVFE trained on the real dataset only achieves 14.29% IoU.

5. Conclusions

Point cloud change detection is a challenging but important research direction for remote sensing, robotics and visual navigation. With this benchmark, we provide a point cloud change detection dataset containing 178 city scenes and 1711 urban object pairs. We have captured some state-of-the-art approaches in this field and will be able to systematically measure its progress in the future.

Evaluation results show that data-driven methods are the current trend in 3D point cloud change detection. The siamese network architecture is helpful to detect the changes in our dataset and approaches which fully exploit the geometric features are more robust for point cloud change detection for minority classes. Additionally, comparison results show that the synthetic data are able to mimic the real data and narrow the reality gap between the virtual environment and the real scene. We also note some limitations of our dataset, which we hope to improve in the future. The dataset needs to be expanded by adding more urban objects for minority classes.

CRediT authorship contribution statement

Yang Gao: Software, Writing – original draft. **Honglin Yuan:** Data curation, Writing – review. **Tao Ku:** Data curation. **Remco C. Veltkamp:** Writing – review. **Georgios Zamanakos:** Methodology, Software, Writing – original draft, Writing – review. **Lazaros Tsochatzidis:** Methodology, Software, Writing – original draft, Writing – review. **Angelos Amanatiadis:** Methodology, Software, Writing – original draft, Writing – review. **Ioannis Pratikakis:** Methodology, Software, Writing – original draft, Writing – review. **Aliki Panou:** Methodology, Software, Writing – original draft, Writing – review. **Ioannis Romanelis:** Methodology, Software, Writing – original draft, Writing – review. **Vlassis Fotis:** Methodology, Software, Writing – original draft, Writing – review. **Gerasimos Arvanitis:** Methodology, Software, Writing – original draft, Writing – review. **Konstantinos Moustakas:** Methodology, Software, Writing – original draft, Writing – review.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The authors would like to thank the Startup Foundation for Introducing Talent of Nanjing University of Information Science and Technology, China (grant number: 2022r014) for financial support. This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH-CREATE-INNOVATE (project code:T2EDK-02743). This work has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 101092875 - DIDYMOS-XR: Digital DynaMic and responsible twinS for XR.

References

[1] Zhu Z. Change detection using landsat time series: A review of frequencies, preprocessing, algorithms, and applications. ISPRS J Photogramm Remote Sens 2017;130:370–84.
[2] Hebel M, Arens M, Stilla U. Change detection in urban areas by object-based analysis and on-the-fly comparison of multi-view ALS data. ISPRS J Photogramm Remote Sens 2013;86:52–64.
[3] Kiba-Janiak M, Marcinkowski J, Jagoda A, Skowrońska A. Sustainable last mile delivery on e-commerce market in cities from the perspective of various stakeholders. Literature review. Sustainable Cities Soc 2021;71:102984.
[4] Chlingaryan A, Sukkarieh S, Whelan B. Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review. Comput Electron Agric 2018;151:61–9.
[5] Sefrin O, Riese FM, Keller S. Deep learning for land cover change detection. Remote Sens 2020;13(1):78.
[6] Deng JS, Wang K, Deng YH, Qi GJ. PCA-based land-use change detection and analysis using multitemporal and multisensor satellite data. Int J Remote Sens 2008;29(16):4823–38.
[7] Singh A. Review article digital change detection techniques using remotely-sensed data. Int J Remote Sens 1989;10(6):989–1003.
[8] Shi W, Zhang M, Zhang R, Chen S, Zhan Z. Change detection based on artificial intelligence: State-of-the-art and challenges. Remote Sens 2020;12(10):1688.
[9] Park S, Ju S, Yoon S, Nguyen MH, Heo J. An efficient data structure approach for BIM-to-point-cloud change detection using modifiable nested octree. Autom Constr 2021;132:103922.
[10] 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(8):9682–704.
[11] Cheng L, Chen S, Liu X, Xu H, Wu Y, Li M, et al. Registration of laser scanning point clouds: A review. Sensors 2018;18(5):1641.
[12] Qin R, Gruen A. 3D change detection at street level using mobile laser scanning point clouds and terrestrial images. ISPRS J Photogramm Remote Sens 2014;90:23–35.
[13] 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(2–3):148–52.
[14] Jaboyedoff M, Oppikofer T, Abellán A, Derron MH, Loye A, Metzger R, et al. Use of LIDAR in landslide investigations: A review. Nat Hazards 2012;61:5–28.
[15] Abellán A, Vilaplana JM, Calvet J, García-Sellés D, Asensio E. Rockfall monitoring by terrestrial laser scanning–case study of the basaltic rock face at Castellfollit de la Roca (Catalonia, Spain). Nat Hazards Earth Syst Sci 2011;11(3):829–41.
[16] Schonberger JL, Frahm JM. Structure-from-motion revisited. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 4104–13.
[17] Seitz SM, Curless B, Diebel J, Scharstein D, Szeliski R. A comparison and evaluation of multi-view stereo reconstruction algorithms. In: 2006 IEEE computer society conference on computer vision and pattern recognition. vol. 1, IEEE; 2006, p. 519–28.
[18] Yew ZJ, Lee GH. City-scale scene change detection using point clouds. In: 2021 IEEE international conference on robotics and automation. IEEE; 2021, p. 13362–9.

[19] Rumsby B, Brasington J, Langham J, McLelland S, Middleton R, Rollinson G. Monitoring and modelling particle and reach-scale morphological change in gravel-bed rivers: Applications and challenges. Geomorphology 2008;93(1–2):40–54.
[20] Ku T, Galanakis S, Boom B, Veltkamp RC, Bangera D, Gangisetty S, et al. SHREC 2021: 3D point cloud change detection for street scenes. Comput Graph 2021;99:192–200.
[21] de Gélis I, Lefèvre S, Corpetti T. Change detection in urban point clouds: An experimental comparison with simulated 3D datasets. Remote Sens 2021;13(13):2629.
[22] Shah S, Dey D, Lovett C, Kapoor A. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Field and service robotics: Results of the 11th international conference. Springer; 2018, p. 621–35.
[23] Guo Y, Wang H, Hu Q, Liu H, Liu L, Bennamoun M. Deep learning for 3D point clouds: A survey. IEEE Trans Pattern Anal Mach Intell 2020;43(12):4338–64.
[24] Berman M, Triki AR, Blaschko MB. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 4413–21.
[25] SiamGCN-GCA. 2023, https://github.com/grgzam/SiamVFE_SiamGCN-GCA. [Accessed 25 May 2023].
[26] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic graph CNN for learning on point clouds. ACM Trans Graph (TOG) 2019;38(5):1–12.
[27] Hu J, Shen L, Sun G. Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 7132–41.
[28] Zhou Y, Tuzel O. Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 4490–9.
[29] Lang AH, Vora S, Caesar H, Zhou L, Yang J, Beijbom O. Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 12697–705.
[30] Thomas H, Qi CR, Deschaud JE, Marcotegui B, Goulette F, Guibas LJ. Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 6411–20.
[31] SiamKPConv. 2023, https://github.com/alikipanou/SHREC2023. [Accessed 25 May 2023].
[32] Qin Z, Yu H, Wang C, Guo Y, Peng Y, Xu K. Geometric transformer for fast and robust point cloud registration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022, p. 11143–52.
[33] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 770–8.