3D URBAN CHANGE DETECTION WITH POINT CLOUD SIAMESE NETWORKS

I. de Gélis[1,2]*, S. Lefèvre[2], T. Corpetti[3]

[1] Magellium, F-31000 Toulouse, France
[2] Université Bretagne Sud, IRISA UMR 6074, F-56000 Vannes, France
[3] CNRS, LETG UMR 6554, F-35000 Rennes, France

KEY WORDS: 3D Change Detection, Point Clouds, Deep Learning, Siamese Network, Kernel Point Convolution, Urban Monitoring

ABSTRACT:

As the majority of the earth population is living in urban environments, cities are continuously evolving and efficient monitoring tools are needed to retrieve and classify their evolution. In this context, analysing changes between two dates is a crucial point. In urban environments, most changes occur along the vertical axis (with new construction or demolition of buildings) and the use of 3D data is therefore mandatory. Among them, LiDAR constitutes a valuable source of information. However, With the difficulty of processing sparse and unordered 3D point clouds, most of existing methods start by rasterizing point clouds (for example to Digital Surface Models) before using more conventional image processing tools. This implies a significant loss of information. Among existing studies dealing directly with point clouds, and to the best of our knowledge, no deep neural network-based method has been explored yet. Thus, in order to fill this gap and to test the ability of deep methods to deal with change detection and characterization of 3D point clouds, we propose a Siamese network with Kernel Point Convolution inspired by Siamese architectures that have already shown their performances on change detection in 2D images and on KPConv network which achieves high-quality results for semantic segmentation of raw 3D point clouds. We show quantitatively and qualitatively that our method outperforms by more than 25% (in terms of average Intersection over Union for classes of change) existing machine learning methods based on hand-crafted features.

1. INTRODUCTION

Due to anthropogenic activities and natural disasters, cities are continuously evolving, yielding critical environmental problems (e.g. air pollution and heat waves). United Nations report that more than 50% of the earth population is currently living in urban areas. Monitoring their evolution is therefore critical and can be achieved with change detection from remote sensing data.

3D Point Clouds (PCs) constitute a relevant source of remote sensing data. Indeed, unlike 2D images, they provide supplementary information related to the height which is especially important in urban areas where most of permanent changes occur on vertical axis. Moreover, urban environments are mostly composed of 3D objects involving geometrical changes (e.g. new building construction), unlike for example land cover changes that are visible through radiometric measures at meter scale. Whether coming from LiDAR sensor or a photogrammetric process, 3D PCs are particularly interesting in urban environments as they provide 3D geometric information on objects.

However, change detection in 3D PCs remains a difficult task because of the unstructured nature of data that prevents standard tools for 2D images from being straightforwardly applied. While rasterization of 3D data into 2D matrices of elevation, called Digital Surface Models (DSM) can be seen as a valuable solution, this rasterization process implies some loss of information since only highest points are taken into account (thus leading to removing information on the facades). Rasterization of the 3D PCs into 3D voxels is somehow better, such a strategy is constrained by the resolution and faces both loss of information and management of sparse voluminous data. This calls for methodologies able to cope with 3D PCs directly, and to distinguish between real changes from those induced by 3D acquisition. Indeed, multiple 3D scans of the same scene lead to different point distributions.

Existing studies dealing with this problem relied on hand-crafted features or distance computation. To the best of our knowledge, deep learning has not been used to address the problem of change detection and characterization directly on raw PCs yet. Nevertheless, a recent study performed change detection with deep neural networks on 3D data (Zhang et al., 2019). In practice, the authors first rasterized their PCs into DSMs and then applied Feed Forward or Siamese networks on those 2D matrices of elevation. A binary result is obtained for each patch of the DSM. As we believe that 3D data could bring much more information to retrieve and classify changes, our goal is to design a deep network able to directly process PCs.

Thus, we build in this paper upon deep learning developments in change detection and PCs semantic segmentation to propose a 3D point cloud change detection technique for monitoring urban areas at the point level. After a presentation of related works on 3D point cloud change detection in urban environments in Sec.2, we detail our proposed framework in Sec. 3. Then, experimental settings and results are reported in Sec. 4. Finally we provide discussion and conclusion in Sec. 5 and 6, respectively.

* I. de Gélis (corresponding author) work is funded by Magellium & supported by a CNES grant. iris.de-gelis@irisa.fr

2. RELATED WORKS

In this section, we briefly review existing change detection and characterization methods in urban environment which rely on 3D PCs. Despite the availability of numerous methods that convert PCs into DSMs, we will not focus on these studies since they are not directly related to the scope of our paper.

Several techniques have already been proposed to highlight and characterize changes in 3D PCs. One can roughly classify them into three different families. The first one consists in segmenting independently the scenes acquired at the two dates and then characterizing changes based on these two classified PCs. There are few studies using this so-called post-classification method especially for urban environments. For example, after generating a building mask from LiDAR data, (Awrangjeb et al., 2015) manually extract building boundaries from aerial images. Then, resulted footprints are compared in order to highlight changes in a 2D map. Following this study, the approach in (Siddiqui and Awrangjeb, 2017) relies on the same idea: after retrieving 3D buildings roof planes at each date, 3D building models are cross-correlated using size and height information of building planes in order to classify them into categories of change. Among methods that use only PCs information, (Roynard et al., 2016) extracts ground points and applies a region growing algorithm to retrieve each separated object. A classification of each remaining object is then made through using a Random Forest (RF) algorithm with several geometric and histogram-based features. The authors suggested comparing small segmented sub-clouds at each date in order to identify and classify changes. The study in (Xu et al., 2015b) also suggests to segment each PC in order to extract buildings. Then, a 3D surface difference map is created by computing a point-to-plane distance between a point in the first set and the nearest plane in the second set. A classification is finally performed to identify various kind of changes (e.g. new dormer, addition of a floor, . . .). This last study shows that errors in the classification part are propagated in the change detection part. As a consequence, the final results highly depend on the scene classification accuracy.

Conversely, pre-classification methods consist in first highlighting changes before characterizing them. As an example in urban environment, (Xu et al., 2015a) first establish an octree from one of the two PCs, and then directly extract changes in the other PC by identifying corresponding missing leaf nodes. A clustering of changed points is made to remove noise and separate the various changes. Finally, remaining clusters are classified according to fixed rules concerning the area, height or roughness. Here again, pre-classification methods embed errors coming from the change detection step in the characterization step. More generally, the more steps there are in the method, the more errors can be propagated to the final results.

This leads us to the last category of methods aiming to perform change detection and characterisation in one single step. For example in (Tran et al., 2018), authors trained a RF algorithm with hand-crafted features to directly obtain a classification of their PCs according to the changes. To do so, the authors extract features related to points distribution, terrain elevation, multi-target capability of LiDAR at one date and between dates. The stability is computed for each point based on the distribution of the neighboring points in both PCs. More precisely, it is defined as the ratio of the number of points in a spherical neighborhood in the other PC and a vertical cylindrical z-oriented neighborhood of the point in the current PC. Then, a RF algorithm is applied to obtain a supervised classification of changes.

To the best of our knowledge, there is no deep learning method able to directly take as input two PCs and classify changes at point level, despite the important progress made recently in the processing of spatial data.

3. METHOD

3.1 Background

To tackle change detection and characterization in 2D images, recent studies proposed to use a deep Siamese Fully Convolutionnal Network (FCN). It consists in a usual encoder-decoder network with skip connections. To extract features, both images will pass through the encoder part which is made of two branches for both images. Each branch is a succession of traditional convolution and pooling layers in order to extract information on data at several scales. The particularity of Siamese network is that, at each step of pooling, the difference of extracted features of the two branches is kept and concatenated in the corresponding scale in the decoder part (Daudt et al., 2018). The two branches of the encoder part may have shared weights if data are quite similar in order to extract features in the same way. When data are significantly different, for example if images are coming from two types of sensors (e.g. optical and radar sensors) weights could be independent, leading to the so-called pseudo-Siamese network (Zhan et al., 2017).

In order to address the 3D part of the problem, we suggest to rely on deep networks able to perform semantic segmentation directly on PCs. To this end, we consider the recent Kernel Point Convolution (KPConv) (Thomas et al., 2019) network that achieved very good results on segmentation and classification tasks even on large urban Aerial LiDAR Survey (ALS) dataset (Varney et al., 2020). In a spirit similar to 2D image encoders, its principle is to apply successive convolutions at various scales. However, unlike images where the selection of pixels involved in a kernel convolution is trivial, KPConv adapts this operation to 3D PCs by selecting *kernel-points*, i.e. points embedded in the specific neighborhood of each convolution operation. KPConv authors implemented different kinds of networks inspired by traditional ones in 2D images, replacing the convolution by the Kernel Point Convolution and the max-pooling operation by down-sampling of PCs using strided Kernel Point convolution. Thus they were able to implement a convolutional network KP-CNN for PCs classification and FCN with skip links (KP-FCNN) for PC segmentation.

We build upon these two principles to build our novel framework that we are detailing now.

3.2 Our framework

To extend the Siamese principle to 3D PCs, we propose here to embed the KPConv architecture in a deep Siamese network where both PCs will pass through the same encoder with shared weights. Similarly to an usual encoder-decoder with skip connections, at each scale of the decoding part, we concatenate the difference of extracted features associated with the corresponding encoding scale (see Figure 1). In practice, the computation of this feature difference is not obvious since PCs do not contain the same number of points and are not defined at the same positions, even in non-changed areas. We suggest to compute this difference in each point of the second PC by retrieving features of the corresponding nearest spatial point in the first PC.

In our study case, both data are PCs acquired by quite similar sensors, thus we chose to use a real Siamese network with shared weight between both encoder branches instead of a pseudo-Siamese architecture.

4. EXPERIMENTS

4.1 Urban change detection dataset

In order to train and test our method, we have developed a PC simulator for urban datasets. Given a 3D model of a city, one can introduce random changes and the simulator generates a synthetic ALS above the city. Then, by adding or removing buildings in the model, we can simulate their construction or demolition. Our simulator allows us to generate unlimited sets of 3D PCs annotated according to these two change classes, and to conduct experimental evaluation in a well-controlled scenario. Besides, this is particularly interesting for training deep networks that usually require a large amount of labelled data. Besides, the simulator makes possible to tune PC acquisition condition, a flight plan being defined according to predefined parameters such as resolution, overlapping between swaths and scan angle. Finally, Gaussian noise can be added to simulate errors and lack of precision in LiDAR range measuring and scan direction. Our resulted PCs have been checked by LiDAR experts and seem similar to real acquisition.

We report in Table 1 all parameters used in our experiment for the acquisition of PCs. We ran a simulation over the city of Lyon, France at a challenging low spatial resolution of 0.5 points/m^2. To mimic real conditions, we added a Gaussian noise of 5 cm in range measurement and 0.01° in scan direction across track.

Parameter	Value
Resolution	0.5 points/m^2
Noise (range)	5 cm
Noise (scan direction)	0.01°
Scan angle	-20° to 20°
Overlapping	10%
Height of flight	700 m

Table 1. Configuration of acquisition simulation.

Figure 2 presents a vertical view of the 3D model of Lyon where all available buildings are shown (elevation is shown in color with a color range relative to each subset). We used several PC pairs for each area, and randomly simulated building construction and destruction between the two dates by either selecting or filtering out each building from the reference 3D model. It could be seen as a data augmentation process especially useful for supervised learning. In particular, we generated 10 different pairs of PCs over the training part, 1 for the validation part, and 3 for the testing part (see Table 2). The flight plan is set randomly, so the swath will not be the same between pairs of PCs, and each acquisition may not have exactly the same visible or invisible parts. Notice that no registration errors were considered here. Figure 3 shows three PCs acquired by the simulator over exactly the same district, and extracted from the three simulations composing the testing set. Finally, we generate for each simulation a pair of PCs with different initial and final states (i.e. different buildings are put into the city model from one pair to another thanks to the simulator), and a different flight plan is set implying various positions of points even on a same building. Indeed, we can see in the three simulations of Figure 3 that LiDAR scanning is not achieved in the same way, so scan lines are not oriented similarly and various facades are visible.

Set	# simulated pairs	# points per date
Training	10	$\approx$ 1,600,000
Validation	1	$\approx$ 335,000
Test	3	$\approx$ 912,000

Table 2. Dataset description.

In order to test the ability of detecting and classifying changes from 3D PCs only, our method does not require any additional input beyond the 3D coordinates.

4.2 Experimental settings

Similarly to the segmentation task in KPConv experiments, we do not feed the network on entire PCs. Indeed, the PCs are too large to be segmented as a whole. Thus, KPConv authors divided their dataset into small spherical sub-clouds (Thomas et al., 2019). In our context of urban ALS PCs change detection, we prefer to use cylinders aligned to the vertical axis over spheres to cope with the privileged vertical direction of ALS data. By doing so, we avoid empty sub-clouds and therefore dealing with complicated change pairs in the training set (note that centers of cylinders are the same for both dates). Indeed, if a sphere is centered at the top of a building that does not include any ground point, and if this building is demolished, then the sphere obtained in the second point cloud will be free of points and will disturb the training of the network. To illustrate, examples of two input cylinders are given in Figure 4. Let us consider a point in the center of the building roof (center of Figure 4b). In this case, the corresponding sphere at the other date could have been empty. Indeed, depending on the radius no ground points would have been visible. With taking cylinders we ensure that each of the sub-clouds contain ground. At testing, cylinders are chosen regularly with an overlapping to be sure that all points are seen at least once by the network. For points seen several times, predicted probabilities are averaged to decide the final label, similarly to voting schemes. It should be outlined that classes are largely unbalanced. Thus during training, centers of cylinders are chosen thanks to a weighted random drawing. The weight is set as a function of dataset balance, in order to set the probability higher for smaller classes. This allows our network to often see classes of change during the training. Moreover, we perform data augmentation through both random rotation around vertical axis for each selected cylinder and random Gaussian noise at point scale.

At each layer of the network, PCs are subsampled via strided KPConv to mimic strided convolution operations with 2D matrices. The cell size of each subsampling depends on the initial cell size dl_0 which is fixed according to the dataset. With our PCs resolution of 0.5 points/m^2, we have empirically observed that dl_0 set to 1 m allowed us to take all available points into account at the first layer. Then, we set $dl_j = 2 \times dl_{j-1}$ for the following layers j. Experiments were conducted with rigid kernels of 15 points. We have empirically set the radius of cylinders to 50 m, following the recommendation of KPConv authors who fixed the radius to $50 \times dl_0$.

Parameters settings have been largely influenced by original KPConv parameters. We thus use a Stochastic Gradient Descent with momentum to minimize a point-wise negative log likelihood loss, with a batch size of 10, a momentum of 0.98 and an initial learning rate of 10^{-2}. Our learning rate is scheduled to decrease exponentially. Unlike KP-FCNN, we included a

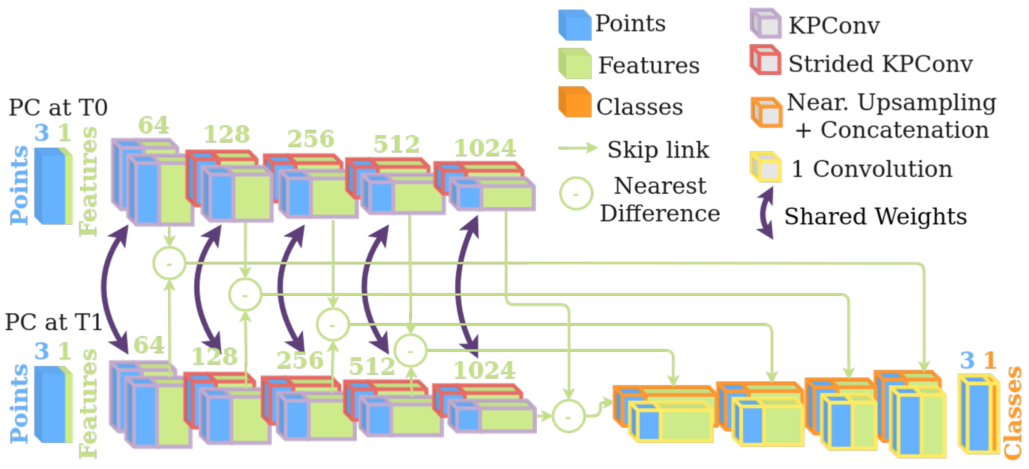

Figure 1. Our Siamese KPConv network architecture.

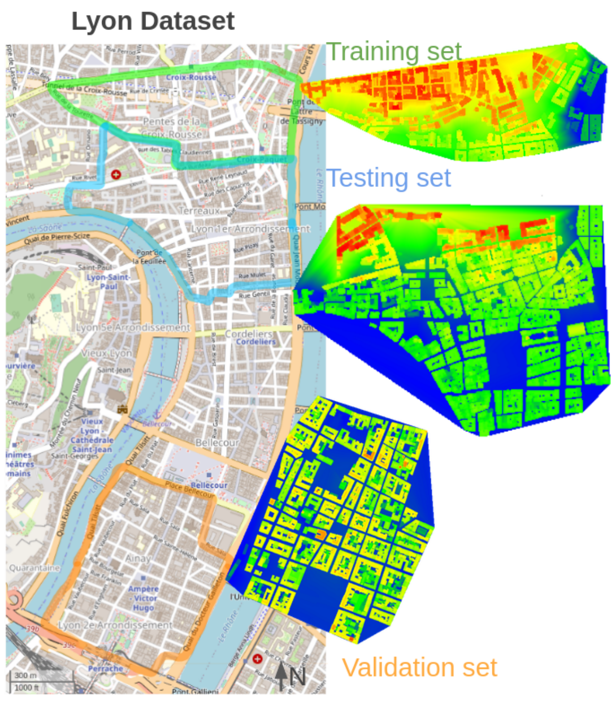

Figure 2. Lyon dataset split into 3 distinct parts: training, validation and test sets. Elevation is shown in color and is relative to each subset.

probability dropout of 0.5 in the last classification layers. Also, in order to prevent from over-fitting, we set a L2 loss regularisation with a factor of 10^{-6}. For this experiment, 325 epochs were required to train the network. Finally, and as already indicated, input cylinders are randomly chosen in training. Thus, the number of input cylinders is another hyper-parameter to set. After experimenting with a few configurations, best results where obtained when 6000 pairs of cylinders were seen by the network by epoch, which corresponds to 600 optimizing steps with a batch size of 10.

The whole development is made with PyTorch and rely on KPConv implementation available in Torch-Points3D (Chaton et al., 2020).

4.3 Results

We recall that, to the best of our knowledge, no deep learning method has been proposed for change detection and characterisation at point scale yet. So we decided to compare our results with those provided by (Tran et al., 2018) where changes are retrieved and classified in one single step also, but using supervised classification of hand-crafted features. A total of ten features have been re-implemented in python, including the stability feature that greatly helps to detection changes. We kept all features presented by the authors except those concerning multi-target capability of LiDAR because our dataset does not contain such information. While in the original work, authors have used a neighborhood radius of 1 m, we have observed that best results were obtained with a value of 5 m (this is due to a different resolution between the datasets). Finally, we use the same training set as for our Siamese KPConv network.

Let us remark that change detection problems are usually strongly unbalanced. Indeed, the huge majority of points or pixels are labeled as unchanged. Thus, precision or accuracy scores do not seem to be fair measures to evaluate performances. We thus report global results in Table 3 using the balanced mean accuracy (mAcc) and the mean of Intersection over Union (mIoU). Besides, since most difficult classes to classify are classes of change, we also averaged the IoU over the change classes (mIoU change). As both methods give pointwise results, metrics are calculated at each point of the testing dataset. Our method overpasses the ML method for all metrics. Especially, if we focus on change classes, we get an improvement of about 15 points of IoU over classes of change in the same training conditions and over the same testing dataset.

Metric (%)	**Ours**	**(Tran et al., 2018)**
mAcc	**96.24**	91.14
mIoU	**93.27**	74.53
mIoU Change	**90.22**	63.41

Table 3. Quantitative results (best in bold) of our Siamese KPConv network compared to RF + hand-crafted features.

We also measured the IoU for each class. As explained in Sec. 4.1, three different pairs have been used for generating the testing set. Table 4 reports the results of both methods over the three pairs of PCs and the overall results for the testing set. Notice that most interesting results are visible in IoU of new building or destruction classes. As expected, results on the unchanged class are excellent. As for changed classes, the net-

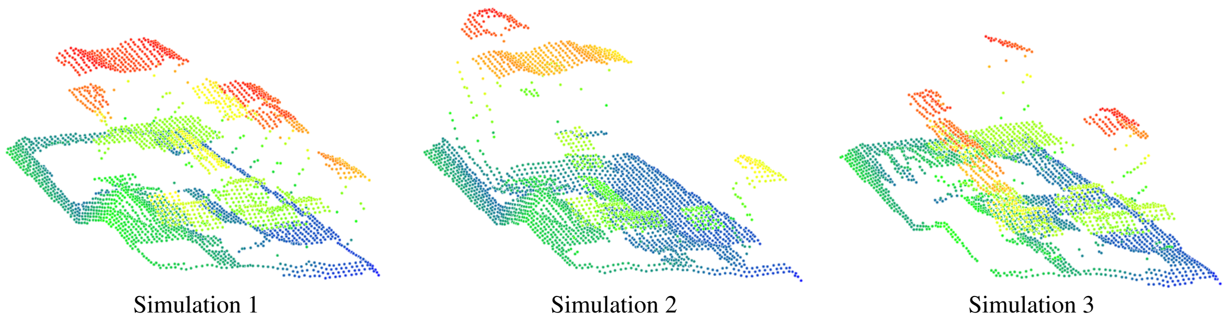

Figure 3. Variability of PCs generated by the simulator over the same district of Lyon. Elevation is shown in a relative color range.

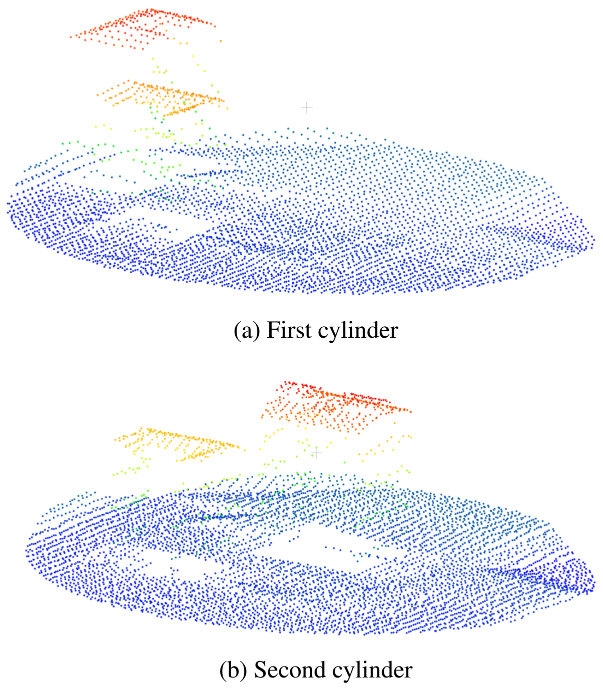

Figure 4. Example of input cylinders: the two input point clouds (a-b) are colorized based on relative elevation.

work is able to retrieve and correctly classify changes in both classes even if a lower score is obtained for the destruction class. We can observe both that (Tran et al., 2018) leads to results of lower quality, but also of high variability among the sets. Conversely, our Siamese KPConv provides stable results, no matter the configuration of acquisition of the two PCs, highlighting the capacity of generalizing results to different pairs of PCs.

Area	Method	Per class IoU (%)		
		Unchanged	New building	Destruction
1	Ours	**99.32**	**96.21**	**82.97**
	RF	98.29	80.08	70.81
2	Ours	**99.47**	**96.37**	**85.57**
	RF	95.54	52.99	61.35
3	Ours	**99.31**	**95.04**	**85.18**
	RF	96.46	64.57	58.37
Total	Ours	**99.37**	**95.87**	**84.57**
	RF	96.76	65.71	63.51

Table 4. Per area IoU scores (best in bold) of our Siamese KPConv network compared to RF + hand-crafted features.

As shown in Figure 5 and Figure 6, applying our method on two PCs (a) and (b) leads to changes (d) in accordance with the ground truth (c), e.g. the main new building and the destruction are well identified by the network. While the RF method shown in (e) leads to numerous misclassifications, our results seem more in accordance with the ground truth, as confirmed by numerical measures. We also remark that there is almost no confusion between changed class, thus when a change is detected it is most of the time well classified. One can see that the RF method with hand-crafted features has difficulties to predict changes on facade: on Figure 5, most of visible facades are identified as new buildings; the same remark holds for the background facade in Figure 6. Looking at PCs (a) and (b) of Figure 5, we observe that underlying LiDAR flights were different between the two acquisitions, which induces hidden parts in the first PCs. Thus, on the second PC, some points appeared on facade of unchanged building because of the different viewpoints of the LiDAR during the acquisition. This specificity of LiDAR acquisition constitutes a challenge for change detection methods since the building has to be seen as a whole. Based on our experiments, our method is more robust and able to understand more globally changes whatever the scale. This is especially true in hidden facades or at the ground with building shadows that generate wrong classification as shown in Figure 5 (e). Table 4 confirms this trend with per area results. Indeed, higher results are obtained in the first pair of PCs where the flight plan is almost the same between the two acquisitions. The RF method faces a much stronger gap of performance than our method.

Main differences with ground truth remain at boundaries of change objects or in some difficult situations as shown in the close-up view of Figure 6. Indeed, in our dataset, ground contains small hills. When a building at the bottom of the hill is low, its roof can be near the ground level of higher ground points (see Figure 6) which implies confusion between ground and building roof. Our method considers there is a new building but the dividing line between new construction and unchanged part is not totally exact. Moreover there is a deconstruction just near this place which makes the task even more difficult. Unsurprisingly, the labels returned by the RF method are all mixed up.

5. DISCUSSION

Looking at figures in Table 4, one can observe that the class of deconstruction has worse results than the new building class. This might be due to the dataset itself. Indeed, deconstruction parts are labeled according to the convex hull of the demolished building at the ground level. Thus, when building ground footprint is not convex, some unchanged ground points are possibly labeled as changed in concavity of buildings. This could leads to some difficulties during the training to understand well what is deconstruction. By exploring visual results of destructed buildings, we notice that prediction were sometimes closer to building boundaries than the annotations. Despite this, most of points are well annotated.

The dataset challenges the methods with the chosen low resolution for the LiDAR (0.5 points/m^2). Notice that in their publication, (Tran et al., 2018) tested their method on LiDAR with a high resolution of 12 to 16 points/m^2. This could explain the lower results obtained with their method. In such acquisition conditions, our method seem to be a better solution.

However the dataset has a challenging low resolution. It is still a bit simplistic as only ground and buildings are present in the area. Even if buildings may have a complex form (e.g. churches), there is no vegetation or mobile objects in our dataset. Similarly, there is only two classes of change (construction or destruction). Conversely (Tran et al., 2018) consider about eight classes of changed or unchanged labels.

Finally the better results achieved with our method come with a more complex training process. In our case, it took about approximately 30 hours to train the network from scratch on a Titan RTX Graphical Processing Unit (GPU). Whereas training the RF algorithm takes only 19 min on a CPU. These results are not specific to our network and are systematic when considering using deep methods. Since the dataset is quite large, the whole test set (900,000 points per date) cannot be given to the network. Indeed, we need to divide it into 5567 cylinders, leading to an inference time of 20 minutes. To compare, the inference time for the RF algorithm is about 4 minutes.

6. CONCLUSION

In this paper, we tackle change detection and characterisation in urban environment. To do so, we proposed a novel deep learning method which takes as input bi-temporal raw PCs and gives final results at the 3D point level. Our method is inspired by 2D change detection deep networks using Siamese architecture and deep network used for semantic segmentation in 3D PCs, in particular Kernel Point Convolutions. To the best of our knowledge, this is the first deep network able to cope with change detection and characterisation in 3D PCs. We have compared our Siamese KPConv network to a more traditional existing method relying on RF algorithm with hand-crafted features. An improvement of about 27 points of IoU over classes of change has been observed. The main advantage of our network seems to lie in its ability to understand structured objects at a global scale, thus leading to a correct classification of hidden part and shadows. Indeed, such a problem remain challenging when dealing with 3D PCs data because the point distribution could highly change even on unchanged parts of a scene.

Our simulated dataset remains quite simplistic. So it would be interesting to test our method on more complex data and especially on real data. However, only a few multi-temporal 3D PCs dataset are publicly available and none of them contain annotations related to the changes. Still, we plan to consider the Actueel Hoogtebestand Nederland (AHN) dataset that provides a semantic labeling per date at the point level and comes with three dates. We thus plan to update the labels according to the changes and to further assess our method on this updated dataset. Besides, it would be interesting to vary acquisition conditions thanks to our simulator in order to test our method on even more challenging conditions such as noisy PCs.

REFERENCES

Awrangjeb, M., Fraser, C., Lu, G., 2015. Building change detection from LiDAR point cloud data based on connected component analysis. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, 2, 393.

Chaton, T., Chaulet, N., Horache, S., Landrieu, L., 2020. Torch-Points3D: A Modular Multi-Task Frameworkfor Reproducible Deep Learning on 3D Point Clouds. *arXiv preprint arXiv:2010.04642*.

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

Roynard, X., Deschaud, J.-E., Goulette, F., 2016. Fast and robust segmentation and classification for change detection in urban point clouds.

Siddiqui, F. U., Awrangjeb, M., 2017. A novel building change detection method using 3d building models. *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, IEEE, 1–8.

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

Tran, T., Ressl, C., Pfeifer, N., 2018. Integrated change detection and classification in urban areas based on airborne laser scanning point clouds. *Sensors*, 18(2), 448.

Varney, N., Asari, V. K., Graehling, Q., 2020. Dales: a large-scale aerial lidar data set for semantic segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 186–187.

Xu, H., Cheng, L., Li, M., Chen, Y., Zhong, L., 2015a. Using octrees to detect changes to buildings and trees in the urban environment from airborne LiDAR data. *Remote Sensing*, 7(8), 9682–9704.

Xu, S., Vosselman, G., Oude Elberink, S., 2015b. Detection and classification of changes in buildings from airborne laser scanning data. *Remote sensing*, 7(12), 17051–17076.

Zhan, Y., Fu, K., Yan, M., Sun, X., Wang, H., Qiu, X., 2017. Change detection based on deep siamese convolutional network for optical aerial images. *IEEE Geoscience and Remote Sensing Letters*, 14(10), 1845–1849.

Zhang, Z., Vosselman, G., Gerke, M., Persello, C., Tuia, D., Yang, M., 2019. Detecting building changes between airborne laser scanning and photogrammetric data. *Remote sensing*, 11(20), 2417.

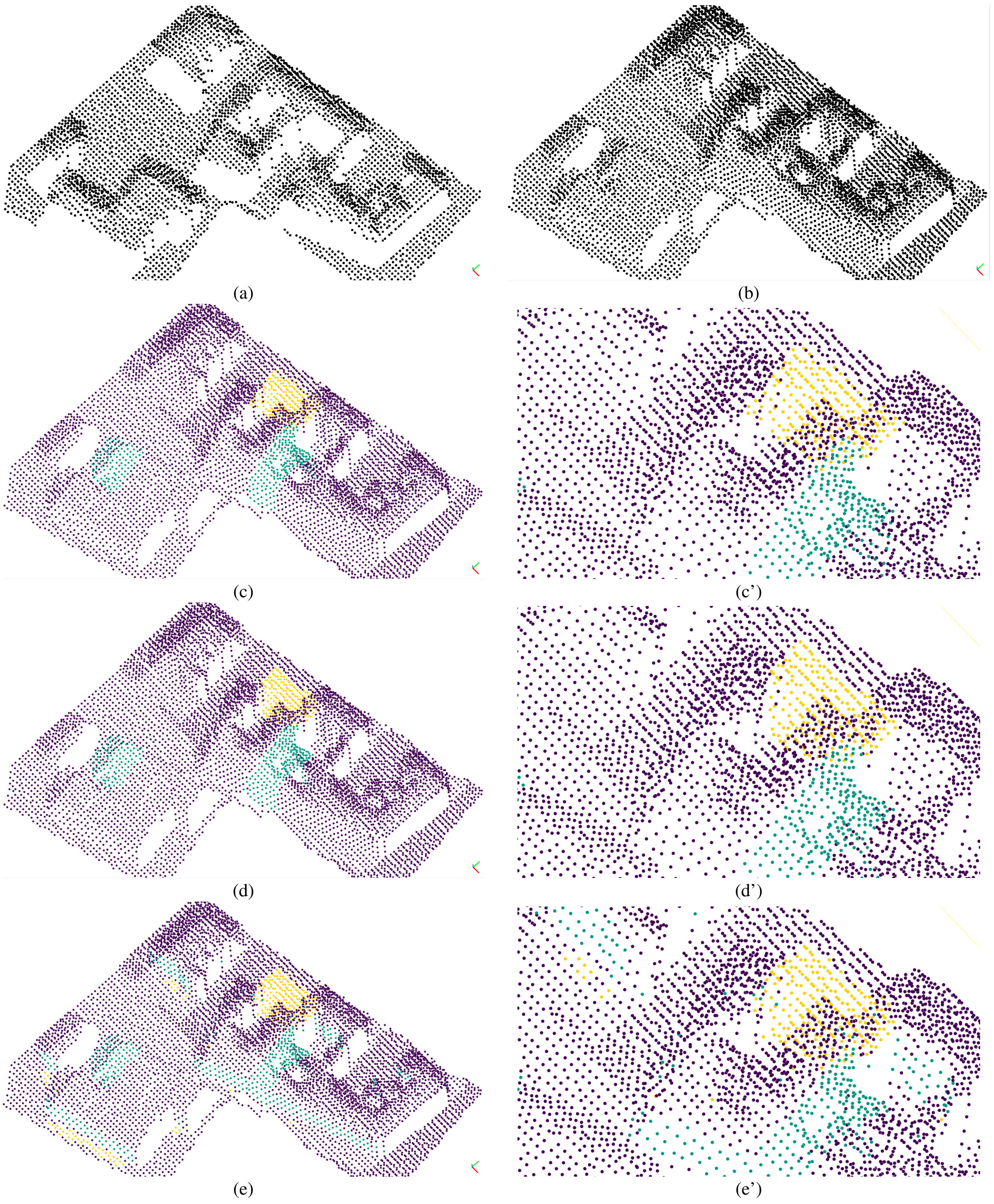

Figure 5. Visual change detection results. (a-b) the two input point clouds ; (c) simulated changes (purple: no change, blue: new construction, yellow: destruction) ; (d) our results; (e) results with (Tran et al., 2018) method; (') denote close-up views.

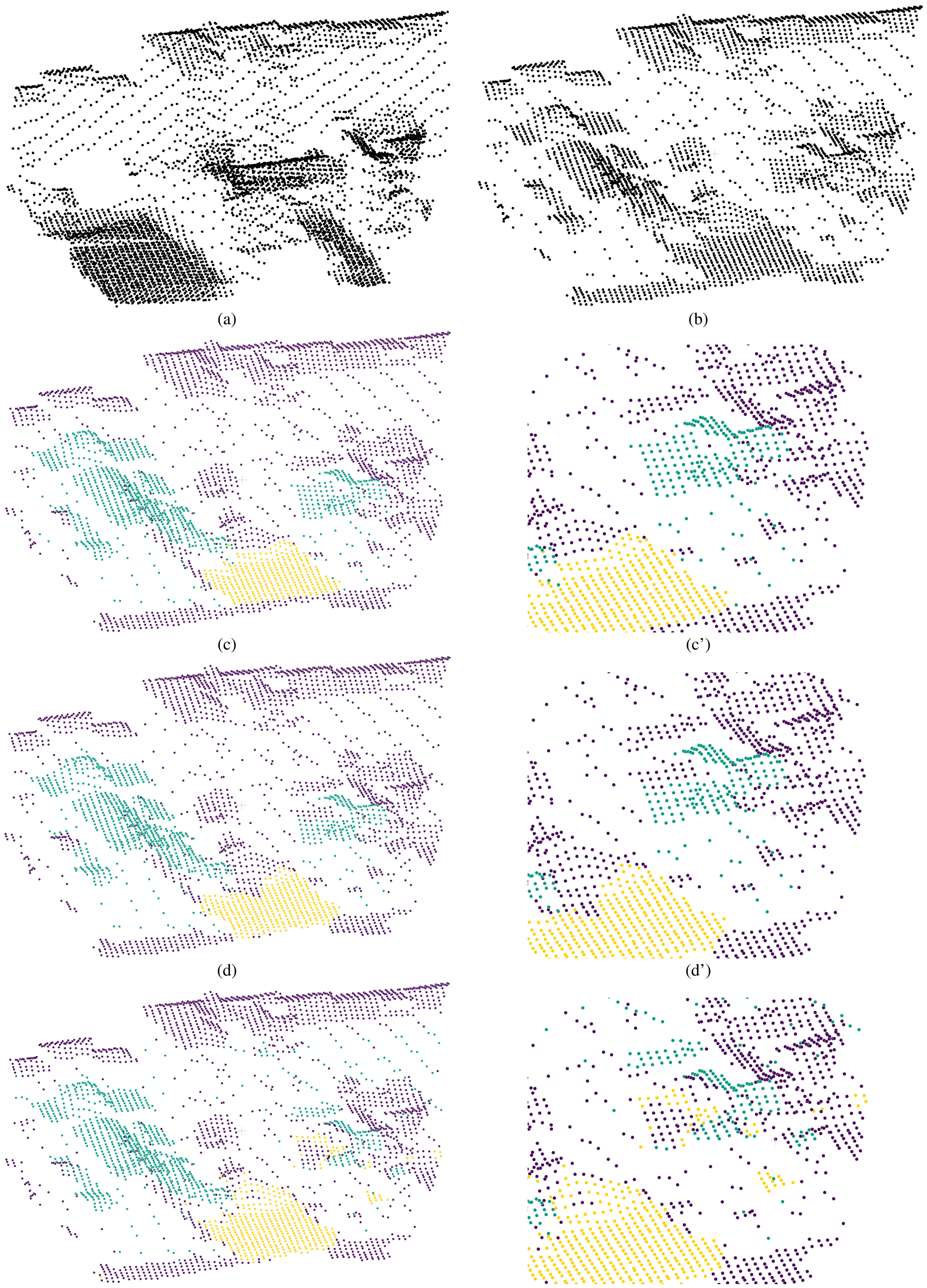

Figure 6. Visual change detection results. (a-b) the two input point clouds ; (c) simulated changes (purple: no change, blue: new construction, yellow: destruction) ; (d) our results; (e) results with (Tran et al., 2018) method; (') denote close-up views.