3D CNN Based Phantom Object Removing from Mobile Laser Scanning Data

Balázs Nagy[2,1] and Csaba Benedek[1,2]
[1]Machine Perception Research Laboratory, Institute for Computer Science and Control (MTA SZTAKI)
H-1111 Kende u. 13-17 Budapest, Hungary, Email: {balazs.nagy, csaba.benedek}@sztaki.mta.hu
[2]Pázmány Péter Catholic University, Faculty of Information Technology and Bionics
H-1083, Práter utca 50/A, Budapest, Hungary

***Abstract*—In this paper we introduce a new deep learning based approach to detect and remove phantom objects from point clouds produced by mobile laser scanning (MLS) systems. The phantoms are caused by the presence of scene objects moving concurrently with the MLS platform, and appear as long, sparse but irregular point cloud segments in the measurements. We propose a new 3D CNN framework working on a voxelized column-grid to identify the phantom regions. We quantitatively evaluate the proposed model on real MLS test data, and compare it to two different reference approaches.**

I. Introduction

Mobile mapping systems comprising integrated mobile laser scanning (MLS), optical imaging and georeferencing technologies are able to rapidly produce high density, very accurate and feature rich point clouds from large environments, even at a standard driving speed. Using a calibrated camera platform with 4-6 digital cameras, the collected 3D measurements can be completed with RGB color values, and relying on the inertial measurement unit (IMU) and Global Navigation Satellite System (GNSS) receivers, the point cloud frames are accurately registered into a geo-referenced global coordinate system. While the high speed of data acquisition is a clear advantage of MLS, due to the huge data size applying efficient automated data filtering and interpretation algorithms in the processing side is crucially needed, which steps still introduce a number a key challenges.

One of the most critical issues is the *phantom* effect caused independent object motions. Due to the sequential nature of the environment scanning process, scene objects moving concurrently with the MLS platform (such as passing vehicles and walking pedestrians) appear as phantom-like long-drawn, distorted structures in the resulting point clouds. On one hand, these phantom regions mislead the point cloud based automated object recognition and scene understanding algorithms. Road management and traffic control authorities rely on huge object databases where accurate classification and validity are indispensable, so identifying phantoms are very essential. Furthermore, the presence of phantom objects are very annoying in visual presentation for example in cultural heritage and augmented reality applications, so it is essential to remove phantom regions and clean the point cloud before further processing.

However, finding phantom regions is very challenging, since their geometric and structural properties are often very similar to real objects. Although it is widely supposed that phantom regions are sparser than the point cloud segments of static objects, this assumption is often invalid for several reasons. First, even the static background regions of the MLS point clouds have a considerably inhomogeneous point density due to occlusions, different distances of the surface points from the scanner's trajectory, varying speed of the scanner platform, and the different laser adsorption/reflection properties of the different surface materials. Moreover, moving objects may travel at varying speed, accelerating and breaking during the observation (for example stopping in front of a traffic light), so the densities and the characteristics of the phantoms are also significantly varying.

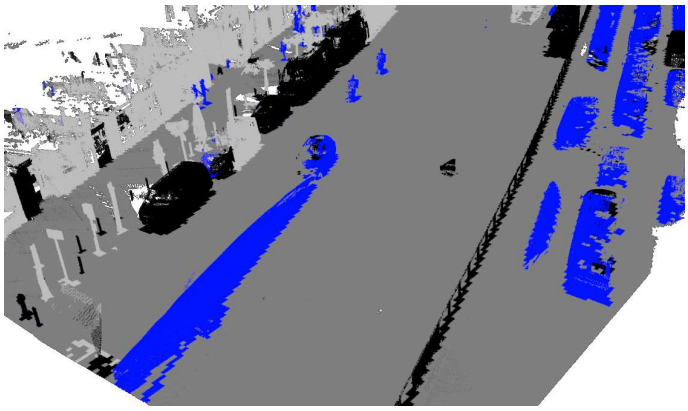


Fig. 1. Segmentation result of the 3D CNN based labeling. The color code of the predicted pillars is the following: dark grey denotes to ground, light grey marks high objects, black color is assigned to short objects and blue represents the phantom objects.

Considering the above detailed complexity of the phantom detection task, we propose a neural network (NN) based approach to tackle the problem. We construct a new 3D convolutional neural network (CNN) which is based on learning local region level 3D features from preliminary labeled training data.

Point cloud classification techniques follow different approaches in the literature. Object-based methods extract first point cloud blobs containing individual objects by a geometric segmentation, e.g. terrain removal followed by floodfill-like grouping of the remaining parts based on Euclidean point dis-

tances. In a second step, the categorization algorithm receives as input the preliminary detected object candidates: for example [1] derives range maps by projecting the object clouds into local vertical planes, and applies feature learning with various NN classifiers on the range images. In general, the drawback of object based approaches is that the object segmentation step may mean a bottleneck of the whole process: misdetected or erroneously merged object segments can mislead the subsequent categorization module. In addition, focusing on our particular task, phantoms can hardly be detected at object levels, as in the MLS scan they may form large connected regions containing several moving object ghosts (see Fig. 7).

Voxel-level methods fit a regular 3D voxel grid to the point cloud scene, and classify the different voxels into various semantic categories such as parts of roads, vehicles, pole-like objects, trees, etc. [2]. This approach allows to perform a detailed interpretation of the scene, however as the number of voxels increase the time and storage requirement of the process becomes higher. In addition, it is less straightforward to incorporate global contextual descriptors to a voxel-based decision process mainly based on local features.

In our solution, we construct a vertical column-based data structure for the CNN framework, which largely exploits the characteristics of urban MLS scenes. The columns - called henceforward as *pillars* - are defined on a regular 2D grid, and several neighboring pillars may correspond to the same object or phantom region, thus no restrictions are given for the horizontal extension of the observed object segments. Since the pillars are examined as unified regions, various local and contextual features can be simultaneously taken into consideration. As a result, we obtain a robust 3D CNN-based segmentation approach, which does not need preliminary object extraction and separation, meanwhile the training data can be prepared relatively easily by a 2D annotation process, which assigns appropriate labels to the adjacent columns of the input point cloud.

II. Related Work

Although phantom detection is a crucial practical problem in point cloud data processing, only a few related works have discussed its possible solutions in the literature. [3] and [4] examine the local density of a point neighborhood and compare it to the average density. Local coherence was introduced in [5], where the authors define a criteria system based on divergence, point distances and the number of nearest neighbors in a certain radius. However, the mentioned methods use only point-level and statistical features and do not take into account higher level, structural information. [6] noted that these approaches were not able to detect phantom regions effectively, they only removed outliers and the noise of the measurement.

Some existing methods [7] mentioning the problem of the occurring phantom objects deal with terrestrial laser scanning (TLS) measurements. TLS data is captured by tripod-mounted scanners from multiple positions, thus the final point cloud is composed of several scans recorded from different sensor positions with partially overlapping fields of view. In the overlapping areas of two neighboring segments we can identify the moving phantom objects, which only appear in one of the scans. Furthermore, [7] build a shadow map and they use ray tracing to detect the remaining temporal objects. Since in this paper we are dealing with mobile laser scanning (MLS) data, which is captured by a sensor in continuous motion, we cannot rely on such temporal features.

III. Proposed approach

In this section we introduce a new 3D CNN based segmentation method, which is able to robustly remove phantom regions from MLS data.

We assume that the moving objects causing the phantoms (such as vehicles and pedestrians) can be found on the ground, and have a maximum height of 3 meters, therefore we exclude all points of the MLS cloud from the further processing steps, which have higher elevation values than 3 meters. In this way, high-altitude objects, such as electric power lines, tree crowns and overhanging balconies - which may occlude the street from a top viewpoint - are mostly eliminated, allowing us an efficient 2D grid based analysis of the scene. More specifically, we divide the point cloud into vertical columns - called *pillars* - of 3m height, and a narrow base ($0.2m \times 0.2m$), which are arranged into a dense regular 2D grid on the horizontal plane. To obtain a complete scene segmentation, we distinguish four semantic classes in the cloud: *ground*, *high objects* (such as facades, tree trunks, lamp posts etc.), *static short objects* (parking vehicles, street furniture etc), and *phantom* regions (effects of moving vehicles and pedestrians). We segment the point cloud at cell level, i.e. each pillar should receive a unique class label from the above defined four-element label set. For local feature extraction, however, we voxelize the pillars, and also consider the voxels of the neighboring pillars to construct a high dimensional input vector for a 3D Convolutional Neural Network (CNN) classifier. Finally, the pillar labels obtained by the 3D CNN are refined in a post processing step considering the labels in the local neighborhoods of the pillars. At the end, each point of the point cloud is classified with the corresponding pillar label.

A. Training Data Generation

The first step of the process is to fit a regular 2D grid onto the horizontal plane according to [8], and to assign each point of the point cloud to the corresponding cell. Each cell defines a $s_c \times s_c \times 3m$ shaped 3D pillar, where the $s_c = 0.2m$ is chosen for the cell size.

For training data generation, we have developed a program with graphical user interface, which automatically fits and visualizes the pillar-grid structure for an input point cloud. Thereafter, the user may scroll through the adjacent pillars of the dense 2D grid, and assign to each pillar a unique training label from the set $L = \{ground = 0, highobject = 1, shortobject = 2, phantom = 3\}$. For speeding up the annotation process, for the high object and road regions an initial estimation has been provided using the cell-based method

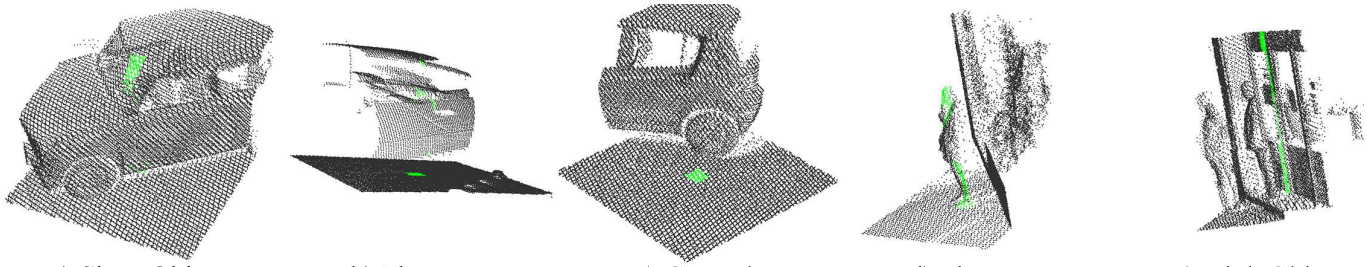


Fig. 2. Different training samples extracted from point cloud data. Each training sample consist of 11×11 pillars and they were labeled according to their central pillar marked with green color.

of [9], thus the operator only needs to approve the results of automatic pre-classification, and manually distinguish the short objects and phantom parts. Each annotated pillar acts henceforward as an independent training sample of the model.

Next we assign a feature vector to each pillar, which will be used both for training and recognition in the CNN framework. This process starts with a voxelization step: we divide the pillars into $0.1m \times 0.1m \times 0.1m$ cuboid segments, thus within each $0.2m \times 0.2m \times 3m$ column we obtain $2 \times 2 \times 30$ voxels. Thereafter we calculate for each voxel the number of the included points from the original point cloud, and store these cardinality values as parameters. For feature vector generation, we consider the 11×11 (horizontal) neighborhood of each pillar, which choice enables to exploit strong contextual information from the data. We do not use any handcrafted features, but the feature vector of the central pillar is obtained by reading the voxels' point cardinality values in the 11×11 pillar-neighborhood one after an other into a $22 \times 22 \times 30 = 14520$ dimensional descriptor vector.

Fig. 2 demonstrates five different samples labeled according to the central pillar of the training kernel (i.e. 11×11 pillar neighborhood), which is highlighted with green color. Due to the lack of object level segmentation, the cube-shaped training regions often contain different types of objects: for example, the segments shown in Fig. 2(d) and (e), contain simultaneously $ground$, $high\ object$ and $phantom$ regions. We emphasize that the training label is determined according to the central column, which corresponds to the $phantom$ class in Fig. 2(d) and to the $high\ object$ class in Fig. 2(e). Fig. 3 visualizes the point cardinality values in the voxelized training data, showing the difference between a static $object$ and a $phantom$. Red cubes indicate here voxels including larger numbers of points than the green ones.

By examining various point cloud scenes, we have observed that in different locations the orientation of the scene objects (such as cars, facades) may follow particular statistical distribution, which may not be valid for other scenes. To avoid overfitting during the CNN training, before the feature extraction process we have rotated the input point cloud segments around the vertical (z) axis with randomly selected angles.

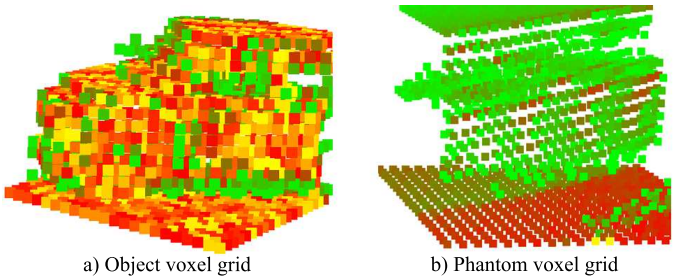


Fig. 3. Voxelized training samples. The red voxels cover dense point cloud segments, while green voxels contain fewer points

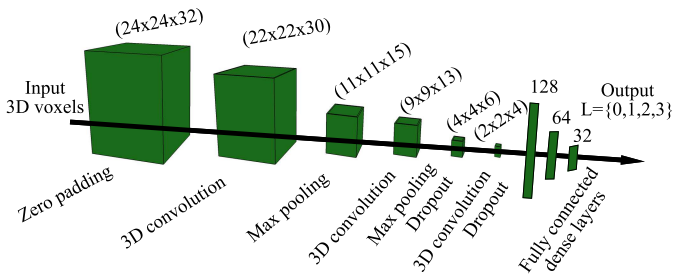


Fig. 4. Structure of the proposed 3D convolutional neural network, composed of three 3D convolutional layers, two max-pooling and two dropout layers. It takes a 3D voxel grid segment as input, and the result of the prediction is an integer encoding the class label $L = \{0, 1, 2, 3\}$.

B. 3D CNN architecture and the training process

We follow the LeNet [10] concept for building the CNN classifier. LeNet-style networks are often applied in various pattern recognition tasks, since in general, their performance can gain on or exceed the accuracy of human perception. Since we are dealing with a 3D structure recognition problem, we had to slightly modify the traditional 2D-CNN structures used in image processing tasks. We train our network with 3D voxelized data, thus we replace the 2D layers of the 2D-CNN models to the corresponding 3D ones.

Basically, the first part of our network is a feature extractor part, which uses a combination of several 3D convolution, pooling and dropout layers. During the offline training session,

we optimize the network parameters on the validation set, and after each iteration we always save best model found so far. The second (association) part of the network is formed by fully connected dense layers and the output is an integer value from the label set L. Fig. 4 demonstrates the architecture and the parameters of the trained network. Our proposed method implements an end-to-end pipeline: the first steps automatically extract and optimize the best feature combination, while the fully connected layers learn the different class models.

As Fig. 4 illustrates, by applying the convolution or pooling operators the size of the output layer is reduced as a function of the size of the operator kernel. Since the dimension of the training data $(22 \times 22 \times 30)$ is quite small, without the input zero padding layer the convolution and pooling operators could significantly reduce the size of the relevant training data. Therefore in the first layer, we padded the borders of the training data with zeros, yielding that after the first convolution the size of the training data still remains $22 \times 22 \times 30$. We added three 3D convolution layers to the network with $3\times3\times3$ kernels, and as shown in Fig. 4, we applied max-pooling ($2\times2\times2$ kernels) and dropout layers between the convolution layers.

To avoid overfitting, during the first dropout we drop the half of the connections, thereafter the second dropout layer eliminates the 25% of the edges. In each training cycle we choose the eliminated connections randomly, so we delete different edges among the training iterations. To optimize and update the network parameters we use a Stochastic Gradient Descent (SGD) algorithm, and we also decrease the learning rate from training epoch to epoch according to the following strategy: $learning\_rate = learning\_rate/number\_of\_epochs$.

After each convolution layer we use ReLu activation function and the output of the network is activated with Softmax function.

The CNN training step uses a k-fold cross validation based on the training data. We define $k = 5$ so in every training session we use 20% of our overall annotated training data for validation and the remaining 80% for the actual iteration of the network's training

C. Neighborhood based segmentation refinement

We perform the classification at pillar level so after the prediction, each pillar gets a label from set L. To refine the quality of the segmentation, and increase the accuracy of phantom removal, we perform a neighborhood based voting, and update the predicted labels accordingly. The process - presented by Algorithm 1 - loops through the entire grid and examines the local cell neighborhoods.

The refinement algorithm calculates the most frequent label in the neighborhood of each cell c excluding the $ground$ class, thereafter it updates label of c according to it. The refinement can smooth the resulting label field, and it also corrects several false predictions originated from strong irregularities in some notable noisy point cloud regions.

Algorithm 1 Segmentation refinement algorithm. Takes the grid (G) and a search kernel size (N).

```
procedure REFINEMENT(G, N)
    for all g ∈ G do
        Initialize neighborhood voting accumulator A
        for ı ∈ [−2, 2] do
            for ȷ ∈ [−2, 2] do
                Reach the given neighbor n = g[i, j]
                A ← Average(n)
            end for
        end for
        g ← A
    end for
end procedure
```

IV. Experiments and Results

In the experiments, we used real MLS test data from dense urban areas of Budapest, Hungary, which was produced by a Riegl VMX-450 scanner. We trained the network on Intel Core i7-4710HQ in 8 threads with 16GB memory. Since the size of the voxelized training data $(22 \times 22 \times 30)$ is quite small, the training process took around 24 hours using only CPU power. The network and learning pipeline were implemented in Python using the Theano framework, while the data management and the visualization of the voxelization parts were implemented in C++ and OpenGL.

A. Training

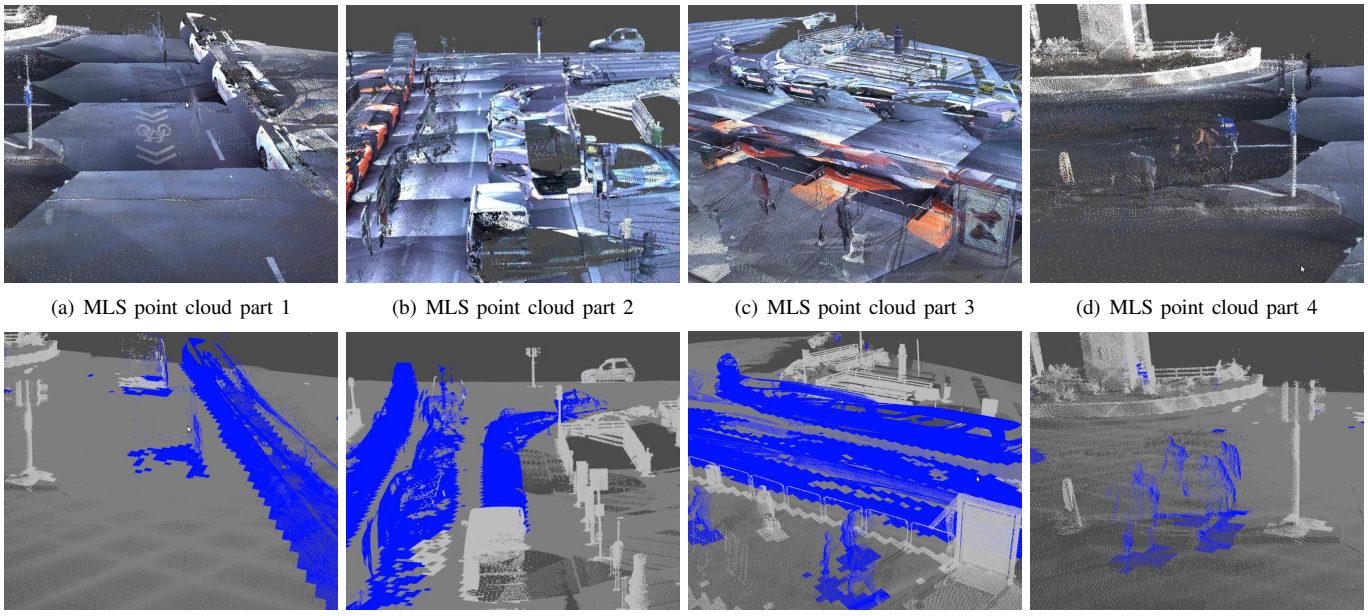
To train our network we annotated several scenes using *phantom*, *ground*, *short object* and *high object* labels. Parking cars, small street furniture elements were classified as *short objects*, while the regions of the *high object* category covered trams, buses, facades, traffic signs, poles and trees etc. Annotated *phantom* regions included moving object such as moving vehicles and pedestrians. We labeled 10000 training samples for each category (a sample is equivalent here to a labeled pillar), thus in aggregate we trained our network with 40000 pillar examples.

B. Testing the 3D CNN framework

We tested the proposed 3D CNN framework in various MLS point cloud segments, without any overlap with the training areas. Evaluation has been conducted in both qualitative and quantitative manners. For the quantitative tests, we manually labeled 39077 pillars from the test set, which labels were used as Ground Truth during the evaluation.

Fig. 5 shows the result of the proposed 3D CNN based segmentation. For convenient visual representation we marked here both high and short objects with light grey color and the detected phantom regions were colored with blue. (Note that Fig. 1 demonstrated the original four-class based segmentation where short objects are colored with black.)

Quantitative detection results for the whole test set are provided in Table I. Using the annotated test data, we calculated the precision, recall and F-measure rates for each class



(a) MLS point cloud part 1 (b) MLS point cloud part 2 (c) MLS point cloud part 3 (d) MLS point cloud part 4

(e) Proposed 3D CNN labeling part 1 (f) Proposed 3D CNN labeling part 2 (g) Proposed 3D CNN labeling part 3 (h) Proposed 3D CNN labeling part 4

Fig. 5. Qualitive result of the proposed pipeline. Dark grey marks the ground, light grey is represents the real objects and blue color is assigned to phantom regions. MLS point clouds were captured by a Riegl VMX-450 scanner.

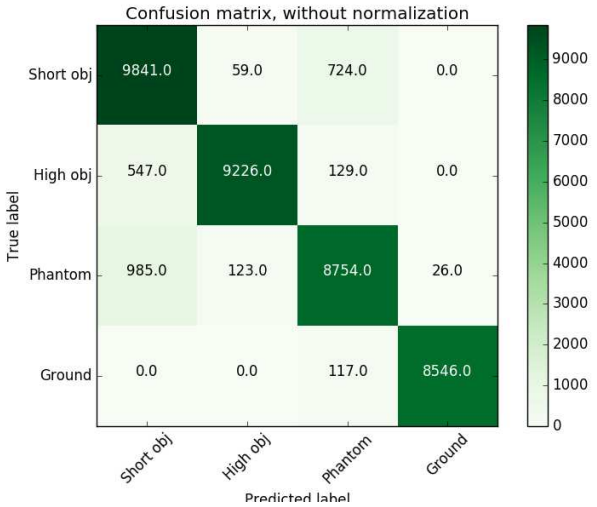


Fig. 6. Confusion matrix of the 3D CNN prediction, with respect to the four classes considered by the proposed method.

separately, and we provided the overall performance metrics considering all classes as well. Each class was detected with an F-rate above 89%, while the overall result surpassed 91%. To obtain a more comprehensive analysis of our method's performance, we also show in Fig. 6 the confusion matrix between the different classes. This figure shows that the confusion rate between the short objects and phantoms - which was the most critical source of errors - remained below 10%.

We should note that despite the applied sophisticated deep learning approach, various occlusion effects may yield partially invisible objects and facade segments, which facts can lead to misclassification. Furthermore, slowly moving objects, such as loitering pedestrians, may be occasionally clustered either as *phantoms* or as *static objects*. For example, in Fig. 5(g) we can see a pedestrian classified as short object, who presumptively did not move during the scanning.

TABLE I
EVALUATION RESULTS OF THE PROPOSED 3D CNN METHOD, FEATURING THE PRECISION, RECALL AND F-MEASURE RATES W.R.T. THE DIFFERENT CLASSES

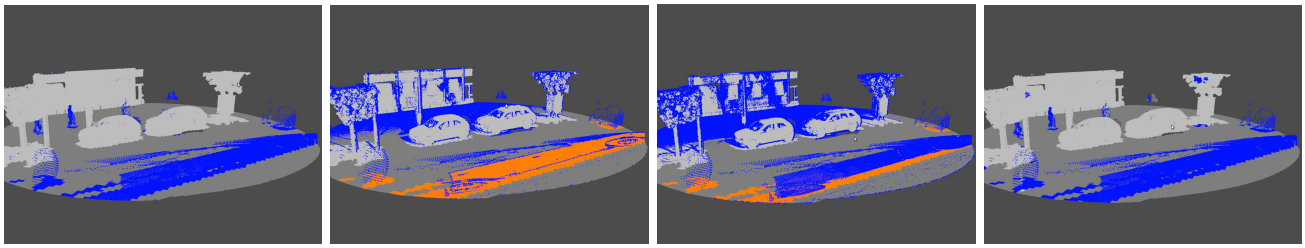
Class	Precision [%]	Recall [%]	F-measure [%]
Short obj.	86.5	92.6	89.5
High obj.	98.1	93.2	95.6
Phantom	90.0	88.5	89.3
Ground	99.7	98.6	99.2
Overall	91.5	91.4	91.5

The proposed method can be used for automatic point cloud cleaning, as demonstrated in Fig. 8: by removing phantom regions detected by the 3D CNN framework the quality of visualized MLS scene can be significantly enhanced.

C. Comparison to reference techniques

We compared our results to two traditional density based filtering methods on a relevant part of the test data set (around 10000 test pillars).

The first reference, called the *Num of neighbors* technique, adopts radius search based phantom removal in the point cloud, so that each point is classified as phantom point, which has less than $t = 100$ neighbors within a $r = 0.2m$



(a) Annotated Ground Truth (b) *Num. of neighbors* (radius: 0.2m, thresh: 100) (c) *Avg distances* [11] (d) Proposed 3D CNN based labeling

Fig. 7. Qualitative comparison of the *Num. of neighbors*, the *Avg distances* and the proposed approach. Dark grey marks the ground, light grey represents the real short/high objects and blue color is assigned to phantom regions. We visualize by orange the regions of false negative phantom detections.

TABLE II
QUANTITATIVE COMPARISON OF THE OVERALL PERFORMANCE OF THE PROPOSED 3D CNN MODEL TO TWO REFERENCE METHODS (EVALUATION PERFORMED FOR AROUND 10000 SAMPLES)

Method	Precision [%]	Recall [%]	F-rate [%]
Num of neighbors	10.43	24.12	14.57
Avg distances	12.34	51.54	19.91
Prop. 3D CNN	97.59	88.90	93.04

search radius. The second reference, called the *Avg distances* method uses a statistical outlier filter implemented in [11]. This algorithm iterates through the entire input twice: During the first iteration it computes the average distance that each point has to its nearest $k = 100$ neighbors. Then, the mean and standard deviation of all these distances are computed in order to determine a distance threshold. During the second iteration the points will be classified as inlier (part of the static scene) or outlier (phantom) if their average neighbor distance is below or above this threshold respectively.

Fig. 7 shows the qualitative, and Table II the quantitative comparison of the reference techniques and the proposed 3D CNN approach. We can see that with the above defined ($t = 100$, $r = 0.2m$) and ($k = 100$) parametrization, the *Num of neighbors* technique (Fig. 7(b)) could only find one fourth, and the *Avg distances* (Fig. 7(c)) method only the half of the real phantom areas (see large false negative regions in the figures marked with orange color). Meantime, they produced very high false alarm rates, erroneously classifying several ground and facade areas as phantoms. By changing the t, r and k parameters, we could increase the strength of the noise filtering, so that nearly all phantom regions became removed. However several additional objects and ground regions had been eliminated at the same time, eroding further the overall F-rate of these approaches.

The above tests confirmed our initial hypothesis from Sec. II: phantom detection cannot be handled by the density based filters which do not take into account any deeper structural information. On the contrary, the proposed 3D CNN framework can efficiently deal with the problem in this segment as well (93% F-rate).

V. CONCLUSION

Traditional statistical methods fail to remove phantom regions caused by moving objects in mobile laser scanning measurements. Therefore we proposed a new 3D CNN based approach to label the scene and find phantom regions. To achieve more accurate results, we refined the segmentation with a neighborhood based voting. We showed that our approach is able to robustly detect phantom regions, meanwhile it segments the scene into four different semantic classes.

ACKNOWLEDGMENT

The authors would like to thank Budapest Közút Zrt (Road Management Department) for the provision of the Riegl VMX-450 MLS test data. This work was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences and by the National Research, Development and Innovation Fund (NKFIA).

REFERENCES

[1] M. De Deuge, A. Quadros, C. Hung, and B. Douillard, "Unsupervised feature learning for outdoor 3D scans," *Proceedings of Australasian Conference on Robotics and Automation*, 2013.

[2] B. Wu, B. Yu, W. Yue, S. Shu, W. Tan, C. Hu, Y. Huang, J. Wu, and H. Liu, "A voxel-based method for automated identification and morphological parameters estimation of individual street trees from mobile laser scanning data," *Remote Sensing*, vol. 5, no. 2, p. 584, 2013. [Online]. Available: http://www.mdpi.com/2072-4292/5/2/584

[3] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. F. Loci, "Fast outlier detection using the local correlation integral." in *In Data Engineering, 19th International Conference on*, Los Alamitos, CA, USA, 2003, p. 315326.

[4] S. Sotoodeh., "Outlier detection in laser scanner point clouds." in *In International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVI-5*, 2006, p. 297302.

[5] T. Weyrich, M. Pauly, R. Keiser, S. Heinzle, S. Scandella, and M. Gross, "Post-processing of scanned 3d surface data." in *In Gross et al.*, p. 8594.

[6] J. Köhler, T. Nöll, G. Reis, and D. Stricker., "Robust outlier removal from point clouds acquired with structured light." in *In Eurographics 2012-Short Papers*, 2012, p. 2124.

[7] T. Kanzok, F. Süß, L. Linsen, and R. Rosenthal, "Efficient removal of inconsistencies in large multi-scan point clouds." in *21st International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision in co-operation with EUROGRAPHICS*, 2013.

[8] A. Börcs, B. Nagy, and C. Benedek, "Fast 3-D urban object detection on streaming point clouds," in *Workshop on Computer Vision for Road Scene Understanding and Autonomous Driving at ECCV'14*, ser. LNCS. Zürich, Switzerland: Springer, 2015, vol. 8926, pp. 628–639.

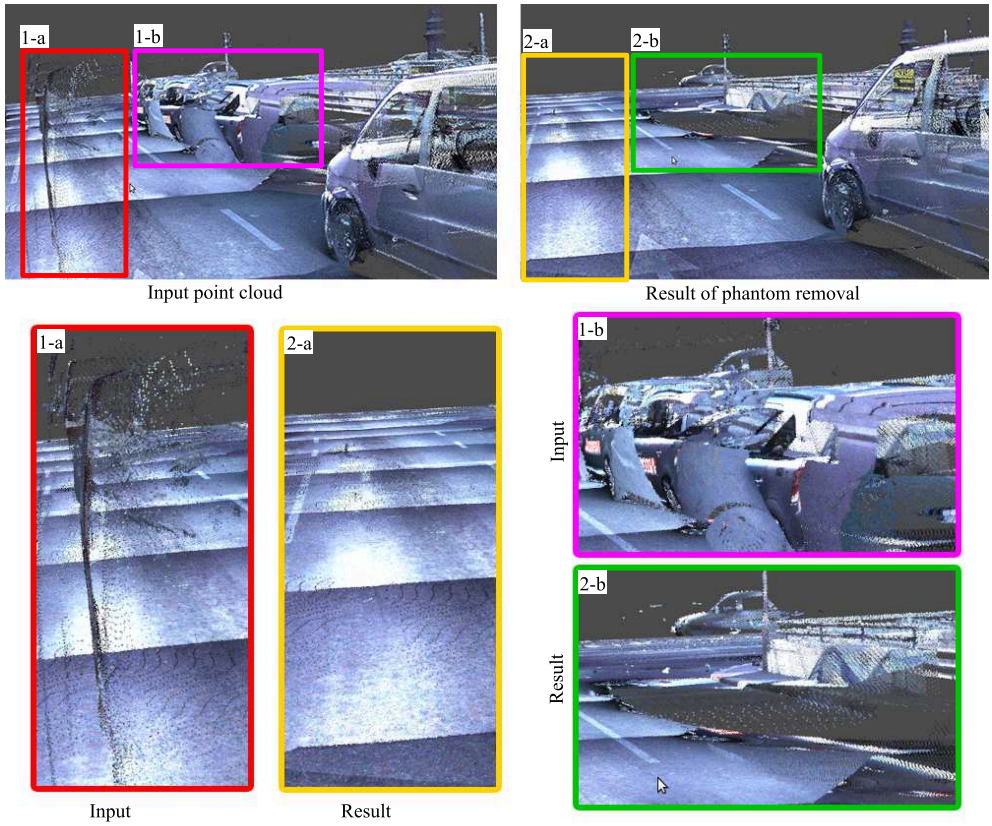


Fig. 8. Demonstration of phantom removal by the proposed 3D CNN method

[9] O. Józsa, A. Börcs, and C. Benedek, "Towards 4D virtual city reconstruction from Lidar point cloud sequences," in *ISPRS Workshop on 3D Virtual City Modeling*, ser. ISPRS Annals Photogram. Rem. Sens. and Spat. Inf. Sci., Regina, Canada, 2013, vol. II-3/W1, pp. 15–20.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[11] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1–4.