www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

IDENTIFYING ROADSIDE OBJECTS IN MOBILE LASER SCANNING DATA USING IMAGE-BASED POINT CLOUD SEGMENTATION

SUBMITTED: December 2019
REVISED: November 2020
PUBLISHED: December 2020
GUEST EDITORS: Hung-Lin Chi & Francis Siu
DOI: 10.36680/j.itcon.2020.031

Gustaf Uggla,
KTH Royal Institute of Technology;
gustaf.uggla@abe.kth.se

Milan Horemuz, PhD,
KTH Royal Institute of Technology;
milan.horemuz@abe.kth.se

SUMMARY: *Capturing geographic information from a mobile platform, a method known as mobile mapping, is today one of the best methods for rapid and safe data acquisition along roads and railroads. The digitalization of society and the use of information technology in the construction industry is increasing the need for structured geometric and semantic information about the built environment. This puts an emphasis on automatic object identification in data such as point clouds. Most point clouds are accompanied by RGB images, and a recent literature review showed that these are possibly underutilized for object identification. This article presents a method (image-based point cloud segmentations – IBPCS) where semantic segmentation of images is used to filter point clouds, which drastically reduces the number of points that have to be considered in object identification and allows simpler algorithms to be used. An example implementation where IBPCS is used to identify roadside game fences along a country road is provided, and the accuracy and efficiency of the method is compared to the performance of PointNet, which is a neural network designed for end-to-end point cloud classification and segmentation. The results show that our implementation of IBPCS outperforms PointNet for the given task. The strengths of IBPCS are the ability to filter point clouds based on visual appearance and that it efficiently can process large data sets. This makes the method a suitable candidate for object identification along rural roads and railroads, where the objects of interest are scattered over long distances.*

KEYWORDS: *Object identification, Point clouds, Mobile mapping, Laser scanning, Deep learning*

REFERENCE: *Gustaf Uggla, Milan Horemuz (2020). Identifying roadside objects in mobile laser scanning data using image-based point cloud segmentation. Journal of Information Technology in Construction (ITcon), Special issue: 'CIB World Building Congress 2019: Constructing Smart Cities', Vol. 25, pg. 545-560, DOI: 10.36680/j.itcon.2020.031*

1. INTRODUCTION

The use of information technology is increasing in society and with this comes an increased demand for structured information about the built environment. In disciplines such as building information modeling (BIM), all data exists as objects, and in order to create models for existing buildings and assets, it is necessary to identify real-world objects in unstructured geodata. BIM is more commonly used in new construction projects than for existing assets, and it is also more common for buildings than for infrastructure. The challenges related to BIM for infrastructure are in some cases different from their building counterparts. Roads and railroads span larger areas, which has implications regarding choices of map projections and georeferencing methods (Uggla and Horemuz, 2018), and the methods used to acquire and process geodata are different as well. Mobile mapping[1], or more specifically mobile laser scanning (MLS), is today commonly used to survey roads and railroads as the method is both safe and efficient (Guan et al., 2016). In comparison to terrestrial laser scanning (TLS), MLS can cover much larger areas, and it eliminates the safety risks of having unprotected surveyors working close to roads and railroads. The output from MLS is typically a georeferenced point cloud together with RGB images that are used to colorize the point cloud. In TLS it is common practice to scan objects from several different directions, which gives a more complete view and a more uniform point density over a scene. On the contrary, a point cloud created by MLS will mostly consist of partially scanned objects, and the point density will decrease significantly as the distance to the vehicle increases.

A significant portion of all infrastructure exists in rural areas where manmade objects that are relevant for mapping and modeling are typically spread out over long distances. The number of points that represent such objects is very small compared to the large number of points representing forest and other surroundings, and it is therefore key to find methods that efficiently can locate regions of interest in the large data sets created by mobile laser scanning. Object identification in point clouds requires that patterns formed by several independent points can be recognized. Despite the innate human ability to perform this type of pattern recognition, it is difficult to formulate and program rules that allow a computer to see what humans can see in an image or a point cloud. For such pattern recognition, machine learning, and especially deep learning, have shown tremendous empirical results. These algorithms learn from data instead of being explicitly programmed, and they have the capacity to comprehend complex problem areas such as vision and natural languages.

Semantic segmentation of street view images is common practice in the field of autonomous driving (for examples see Yang et al., 2018; Kong and Fowlkes, 2018), and even though images typically are captured during MLS, they are rarely used for object identification in point clouds (Che et al., 2019). Due to their higher resolution, images can show details and nuances that are not visible in point clouds. Methods for object recognition and segmentation of images are also more researched than their point cloud counterparts and finding ways to utilize this information and technology for object identification in point clouds can be of great value.

1.1 Aim and contribution

The aim of this research is to explore the benefits of utilizing image information for object identification in point clouds and to develop a robust method for identifying roadside objects in large MLS data sets. Uggla (2019) showed that semantic segmentation of images and perspective projection can be used to identify noise barriers in MLS data, and this article is a continuation of that work. In this article, the method *image-based point cloud segmentation* is formalized and divided into two steps, and an example implementation where it is used to identify roadside game fences along a country road in Sweden is provided. To validate the performance of the method we compare it to PointNet (Qi et al., 2017), which is a neural network capable of end-to-end classification and segmentation of point clouds. PointNet has shown strong performance in semantic segmentation and can be applied to different types of point clouds without the need for adaptation, and this makes it suitable for this type of comparison.

The strength of the IBPCS method is that it can filter, or segment, point clouds based on inferred semantic information rather than point characteristics. It also has a close to linear complexity with regards to distance covered or the number of points in the point cloud and is therefore suitable for identifying infrequent roadside

[1] Mobile mapping is a process of acquiring geodata from a vehicle, typically equipped with laser scanners and/or cameras

objects spread out over long distances, which is typical for infrastructure such as roads and railroads in rural areas. Since the method primarily uses images to identify relevant regions in point clouds, it is somewhat robust to variations in driving speed, point density, and partially scanned objects.

The game fence object type is challenging in the sense that it is transparent to both cameras and laser scanners, and in this article, it is used as a proxy for all roadside objects. The purpose is not to present the most accurate method for identifying game fences, but rather to showcase the capabilities of IBPCS. If it is possible to successfully identify game fences using IBPCS, it follows that it should be possible to identify most other roadside objects as well, given that they can be recognized in images.

1.2 Background and related research

The basis of modern image recognition is the convolutional neural network (CNN), which was initially described by LeCun at el. (1990). Since the development of AlexNet in 2012 (Krizhevsky et al., 2012), the CNN architecture has dominated vision-related recognition fields (LeCun et al., 2015). A CNN is a neural network where the main operator is a convolutional filter that processes each location in the input independently. The semantic classification of an image rarely depends on where within the image certain objects appear, and the location invariant nature of the CNN therefore makes it suitable for visual recognition. The fully convolutional network (FCN) (Long et al., 2015) is an adaptation of a conventional CNN that performs semantic segmentation (pixel-wise classification) instead of classifying entire images. This is accomplished by replacing the last fully connected layer of the CNN, which maps the output from the last hidden layer to a vector representing the different classes, with yet another convolutional layer of size 1×1 and with a depth corresponding to the number of classes. Donahue et al. (2013) and Razavian et al. (2014) have shown that large portions of what a CNN learns from a data set such as ImageNet[2] can be transferred to other domains. This procedure is known as transfer learning. In practice, this means that it is possible to copy the architecture of a top-performing CNN, initialize it with the weights it has learned from ImageNet, and retrain the topmost layers on a much smaller data set without over-fitting to the small data set. This makes CNNs more viable in real-world scenarios, as labeled training data typically is hard to find and time consuming to produce.

Guan et al. (2016) conducted a literature review regarding the use of laser scanning and mobile mapping for road applications. The authors concluded that MLS efficiently and safely can capture large amounts of data that include information about the road surface, road markings, and trend-lines, as well as objects located in the proximity of the road, such as road signs, traffic lights, and barriers. As a future challenge, the authors identified the development of efficient post-processing routines where parametric representations and semantic objects are extracted from the raw point cloud. Che et al. (2019) conducted a literature review of object identification in data sets captured by MLS. The review showed that most existing work is conducted in urban areas and that the use of image data is very limited. Rasterization of point clouds is used for identification of objects on planar surfaces such as roads, as this allows the use of mature and high-performing image processing methods, and RGB information is used for object identification in point clouds, but very few studies utilize the source images for object identification. The authors recognize the potential in using CNNs and FCNs to classify MLS data, and they mention the use of rasterization, virtual cameras, and voxelization as possible methods to vectorize the point clouds so that they can be consumed by a neural network. The use of RGB images as input for a CNN or FCN is not mentioned in the review. Challenges identified by the authors include object identification in rural areas, managing complexity in large data sets, and developing methods invariant to the speed of the capturing vehicle.

Pu et al. (2011) recognized the potential in using road-borne mobile mapping systems to effectively acquire dense point clouds covering roads and roadside objects. The authors proposed a rule-based classification method where the point cloud was segmented into ground, on-ground, and off-ground, which was used as a basis to identify objects such as traffic signs, poles, barriers, and walls. Yu et al., (2015) presented a method for classifying road markings in point clouds. Road markings were extracted using a rule-based approach that considered the geometry and intensity of the point cloud. The road markings were transformed into 2D raster images that were classified using a two-layer Deep Boltzmann machine (Salakhutdinov and Hinton, 2009). Guan et al. (2015) proposed a method to extract and classify trees from point clouds in urban areas. The trees were extracted from ground points

[2] ImageNet is a data set consisting of well over a million images divided between 1000 classes that is used in the ImageNet Large Scale Visual Recognition Challenge (Russakovsky et al., 2015)

using a voxel-based upward-growing algorithm, and the extracted tree clusters were classified using a two-layer deep Boltzmann machine. Soilan et al. (2016) proposed a method that can extract road signs from point clouds and classify the road signs using image recognition. The road sign geometries were detected in the point cloud from their intensity and projected to the corresponding images captured by during MLS. The image was cropped according to the bounding box of the projected geometry, and the cropped image was classified using a support vector machine (SVM). Arcos-García et al. (2017) proposed a similar method but where the image classification was performed by a neural network instead of an SVM.

Image-based deep learning has been utilized to identify regions of interest and to measure efficiency in construction sites (Chen et al., 2019; Chen et al., 2020), but there are no examples of it being used as a primary data source in point cloud object identification in the existing literature.

2. IDENTIFYING GAME FENCES IN A POINT CLOUD

Game fences are roadside objects that occur sporadically along country roads and for which there is a demand for automatic mapping methods (Halvorsen, 2015). The appearance and geometry of a game fence is somewhat special as it is largely transparent for both cameras and laser scanners. Game fences are commonly occurring in close proximity to forest, which makes it challenging to distinguish them from trees and branches. The data set used was captured by a road-borne MLS system in May 2016 on a mostly overcast day, and it covers 7 km of country road in Västergötland, Sweden. The point cloud consists of approximately 160 million points and is accompanied by 4200 images that were captured at 700 locations and in 6 different directions relative to the trajectory of the vehicle. The images were captured by a Ladybug 5 camera (Point Grey, 2017) that was mounted in such a way that its lenses were facing 36°, 108°, 180°, 252°, and 324°, relative to the trajectory of the vehicle, as well as one lens facing straight up. The dimensions of the rectified images were 4096×4896 pixels. The distance between the laser scanner and the game fences was typically 7-10 meters.

2.1 Image-based point cloud segmentation

The IBPCS method consists of two stages. The first stage is to classify pixels in images and to transfer this classification to the point cloud. This creates a subset of the point cloud with limited spatial extents and with limited semantic content. In the second stage, this subset is processed geometrically in order to refine the selection and to divide the points into discrete objects, see Figure 1. The first stage is performed identically regardless of object type while the algorithms used in the second stage will depend on the characteristics of the sought-after objects. The algorithms used in the second stage can vary from simple noise filtering and point clustering (Uggla, 2019) to more advanced algorithms such as the one described in Section 2.1.2.

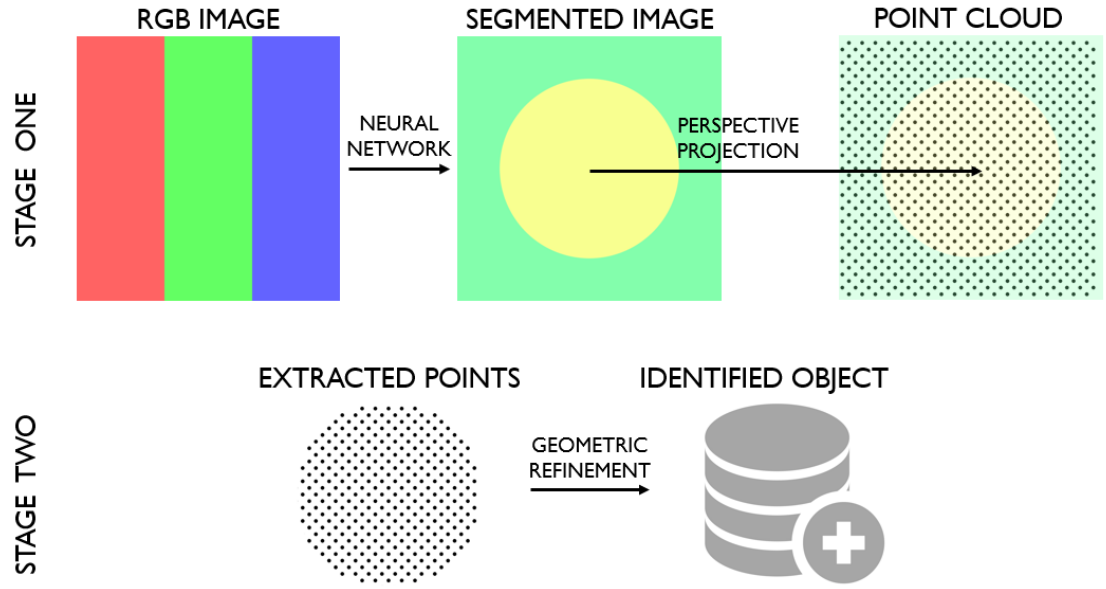

Figure 1. Schematic overview of the two stages of image-based point cloud segmentation (IBPCS).

The point cloud was stored in a relational database and indexed on timestamps. This made it possible to quickly extract points captured within a specific temporal interval. The most unobstructed views of the roadside features that were of interest in this article were given by the two lenses facing right, since the vehicle was driving on the right side of the road, and only images from those two lenses were therefore used. This means that 1400 images out of the total 4200 were used.

All data processing was performed using code written by the authors in Python 3.7 together with the libraries SciPy (Jones et al., 2001), scikit-learn (Pedregosa et al., 2011), TensorFlow (Abadi et al., 2015), and Keras (Chollet, 2015). All image processing was performed using a Nvidia GTX 970 GPU.

2.1.1 Stage one - semantic segmentation and classification transfer

The semantic segmentation was performed using an FCN with a filter size of 64×64 pixels and a stride of 32 pixels as described by Long et al. (2015). The FCN was based on the 4 convolutional layers of the CNN VGG16 (Simonyan and Zisserman, 2014). To utilize the benefits of transfer learning, the convolutional layers from VGG16 were initialized with the weights learned from the ImageNet data set (Russakovsky et al., 2015) and the first three of them were frozen during the training. It was therefore only the fourth convolutional layer from VGG16 and the new convolutional top layer that were affected by the training process.

Due to limitations of GPU memory, the images were cropped into square tiles with sides of 1000 pixels. The camera lenses had a wide viewing angle that caused all relevant pixels to be located in the center of the images, and two or three (depending on camera) tiles were extracted automatically from each image, see Figure 2. A training sample consisted of one 1000×1000×3-pixel RGB image together with a 1000×1000×1-pixel image that shows the class affiliation of all pixels in the RGB image. 526 positive and 531 negative samples[3] were created. The labels for the positive samples were created using the Matlab tool LIBLABEL (Geiger et al., 2014), which allows the user to annotate images with polygons corresponding to different object classes. Empty labels for the negative samples were generated automatically.

To determine how well the FCN can identify game fences in images, the images were divided into five pools containing equal proportions of positive and negative samples, and the FCN was trained using cross validation. After cross validation, a trained FCN was used to create predictions for all images in the data set. The information in the predictions that contained game fence pixels was transferred to the point cloud using perspective projection. The point cloud was transformed from the map projection to the coordinate frame of the vehicles inertial measurement unit (IMU), from the IMU-frame to the frame of the camera body, and from the camera-frame to the frame of the individual lens, by using a 6-parameter Helmert transformation:

$$X_B = T_B + R_A^B X_A$$

where X_A are the coordinates in A-frame, X_B are the coordinates in B-frame, T_B is the translation vector from A to B described in B-frame, and R_A^B is the rotation matrix from A to B. Once transformed, the point cloud was projected to a plane parallel to the image sensor using perspective projection:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \frac{1}{Z}$$

where X, Y, and Z are coordinates in the camera frame, and x and y are coordinates in the parallel plane. The coordinates in the parallel plane were transformed to pixel coordinates (u, v) in the image by using:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where f_u and f_v are focal lengths for the respective coordinate axes, and where c_u and c_v are the coordinates for the intersection between the sensor and the optical axis. The points, now in pixel coordinates, were given the same classification as the pixels they intersect.

[3] A positive sample is a sample that contains the sought-after object class, and a negative sample is a sample that does not contain the sought-after object class

This transformation and classification transfer were performed for all positive predictions, and the results were stored in a database. After the transformation from the IMU-frame to the camera-frame, all points with negative Z-values were discarded, as points behind the camera cannot appear in the image. In order to reduce the computational cost, only the points that were captured within a temporal interval of 0.8 seconds centered around the timestamp of the image were transformed. The images were captured roughly 0.72 seconds apart, and an interval length of 0.8 seconds proved to include all relevant points visible in the image. A schematic overview of the implementation of IBPCS stage one is shown in Figure 2.

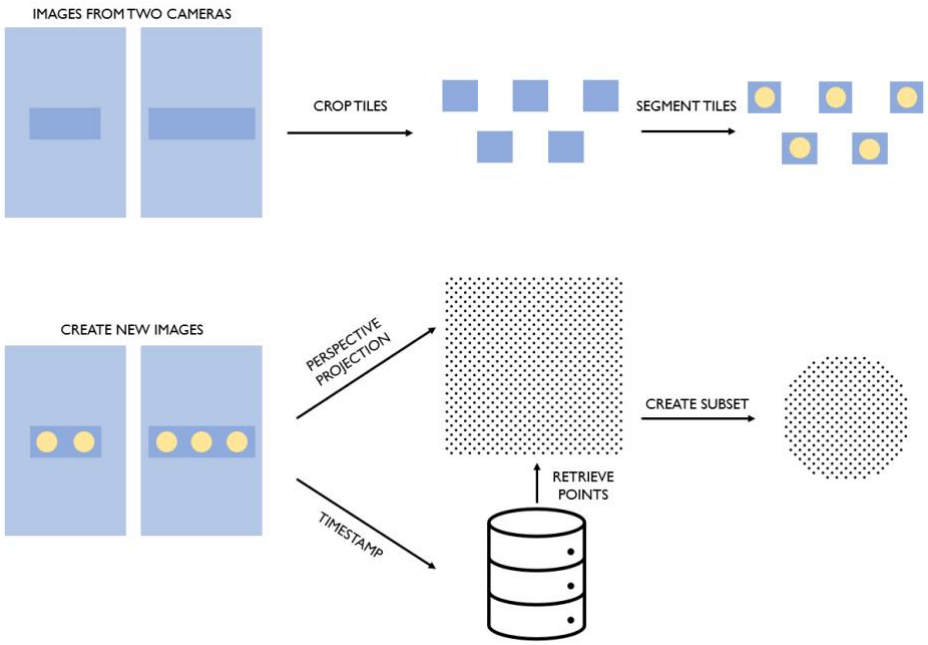

Figure 2. Flowchart showing the implementation of the first stage of IBPCS. Two cameras facing right were used, and from each image pair, 5 tiles were cropped. Each tile was segmented by the FCN, new images with identical dimensions to the original ones were created, and the tiles were placed in their original locations. The timestamp of each image capture was used to retrieve the corresponding points from the database, and the pixel information was transferred to the points via perspective projection. Finally, the points were filtered based on this newly transferred attribute, and a subset was created.

Since any given point in the point cloud could be visible in more than one image, most points were given several, possibly contradictory, classifications. In order to export a classified point cloud from the stored classifications, a single classification had to be chosen for each point. In this article, each match between a pixel and a point was stored together with the 2D distance from the image center to the pixel and the 3D distance from the lens's focal point to the point in the point cloud. The effects of radial distortion are greater in the edges of an image, and all points were therefore classified according to the match where the pixel was closest to the image center. One could consider using more sophisticated decision making, for example weighted voting, but the chosen approach was deemed sufficient. The subset exported from the database contained all points classified as game fence that were within 15 meters from the lens's focal point for at least one of its matches. Points farther away were discarded because the game fences always appear in closer proximity to the road.

Generally, it could be beneficial to use some sort of visibility analysis, for example the method described by Vechersky et al. (2018), when choosing the correct classification for each point. Even though the camera and the scanner were mounted close to each other, roughly 0.5 meter, and therefore shared approximately the same field of view at all times, there will always be points that are intersected with the pixels in an image without actually being visible in the image. However, due to the transparent nature of game fences, visibility analysis would not solve the problem of objects behind the game fence being intersected with the image.

2.1.2 Stage two – geometric processing

The result of the first stage is a subset of the point cloud that contains all likely game fence points as well as any points appearing behind them or in front of them. Unlike the ground and trees that surround them, game fences form a polyline in the horizontal plane. This characteristic, together with the fact that game fences are vertical structures, was used to separate the game fences points from the rest of the subset.

In order to identify ground points, a voxel-grid that covers the point cluster was created. The voxel size was 10 centimeters, and a voxel was seen as populated if it contained at least one point. All vertical columns where more than one voxel was populated were kept, and all other points were discarded. This resulted in a point cloud where all points were part of some sort of vertical structure. The linear segments constituting the game fences were identified using Hough transform (Ballard, 1981), which is an algorithm that is able to recognize weak geometric shapes in strong noise. A down-sampled version of the point cloud was created in order to achieve a more uniform point density, and the two versions are from now on referred to as the *dense point cloud* and the *sparse point cloud*. All unique coordinate combinations in the horizontal plane were extracted from the sparse point cloud, effectively creating a binary 2D image. The vertical axis was in this step ignored, and two points with identical coordinates in the horizontal plane but with different heights were therefore considered as one point. For each point in this image, lines were created with attitudes ranging from 0° to 180° and with 1° intervals. Each line was stored as a tuple consisting of the attitude angle and the orthogonal distance between the line and the origin of the 2D coordinate system, which created a list of all possible linear features in the image.

The most frequently occurring line was chosen, and within this line, the largest cohesive cluster of points was chosen using density-based clustering (scikit-learn, 2018). Even though the most frequent line represents a section of the game fence, there will in many cases be points outside of the game fence that are located on the same line. The clustering is therefore necessary to separate the game fence section from e.g., tree points that happen to be part of the line. All points along the extents of this cluster, including the cluster itself, were removed from the sparse point cloud, see Figure 3. This procedure was repeated until it was no longer possible to find a cohesive linear cluster containing at least 300 points. The limit of 300 points was determined empirically.

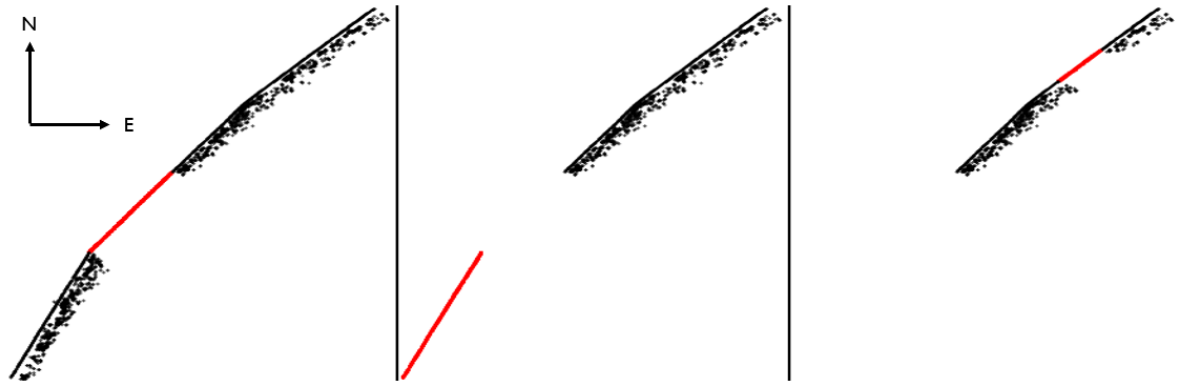

Figure 3. Three images showing the first three steps (in order left to right) of the sequential application of Hough transform. The images show the point cloud viewed from above. The red points are the identified linear segment, and the black points are the remaining points. The identified segment and all points along its extent were removed in each step.

The topology of the lines was determined by the points' timestamps, and intersections were created between neighboring lines if their angular difference was greater than an empirically determined threshold. In other cases, a new line was created between the end points of the neighboring lines. The resulting polyline was matched against the dense point cloud, and all points within a horizontal distance of 0.2 meters from the polyline were selected. Since the ground points had already been removed from the dense point cloud, in cases where there was an actual gap in the game fence, no points would likely be retrieved from the matching and the gap would therefore be present in the final point cloud even though it was closed in the polyline.

2.2 PointNet

PointNet is written using Python and TensorFlow, and the code has been made available by Qi (2019). The creation of training data described in this section was done using Python-code written by the authors and all training and inference were performed using a Nvidia GTX 970 GPU.

PointNet is capable of point cloud classification, part segmentation, and semantic segmentation, out of which semantic segmentation is most suitable for finding objects in MLS data. Semantic segmentation of point clouds is similar to semantic segmentation of images in the sense that each individual point is given a classification, and that objects of a given class may or may not be present in a sample presented to the network. PointNet requires data to be in the form of blocks where every block contains the same number of points. In Qi et al. (2017), the semantic segmentation functionality of PointNet was evaluated using the Stanford 3D semantic parsing data set (Armeni et al., 2016) that consists of point clouds created by indoor terrestrial laser scanning. The dataset consists of rooms, and for each room, the horizontal coordinate axes are aligned with the walls of the room. Each room was divided into 1×1-meter blocks in the horizontal plane, and the points in each block were either down-sampled or duplicated so that each block contained exactly 4096 points. Every point was represented by a vector consisting of normalized local 3D coordinates describing the position of the points within each block, normalized 3D coordinates describing the position of the points relative to the entire room, and the RGB values of the points. The coordinates are normalized in such a way that they range from 0 to 1 within each block and within each room.

To determine how well PointNet can identify game fences in the given point cloud, two data sets were created. The first data set consists of all points on the right side of the road along the stretches where game fences were present, and the second data set consists of the filtered points created from the first stage of IBPCS. These two data sets will from now on be referred to as the *full point cloud* and the *IBPCS subset*. For both data sets, four regions with varying topography (flat ground and forest) were extracted. Each region was then divided into five chunks creating a total of 20 chunks, see Figure 4. The chunks were divided into five groups so that each region was represented in every group.

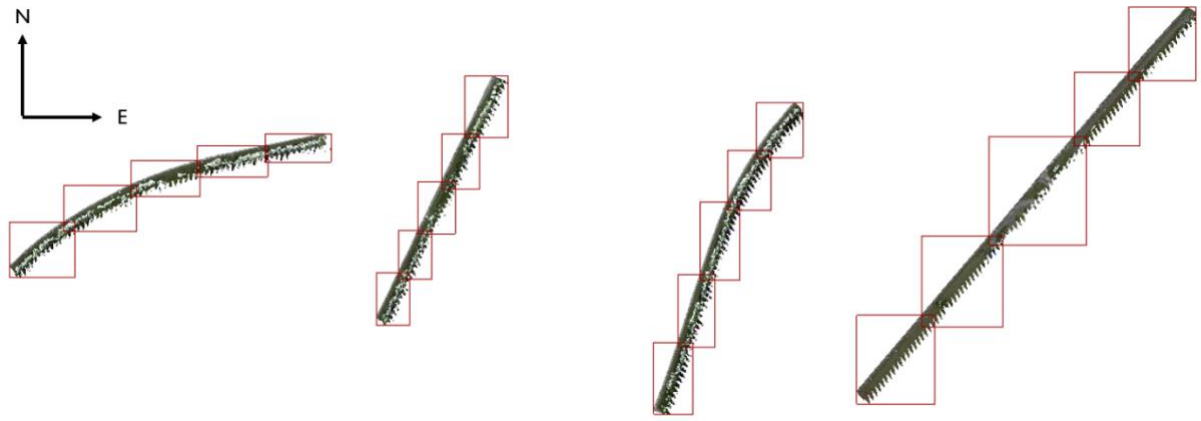

Figure 4. Regions and chunks used for training PointNet (full point cloud). The regions have been moved closer together, but their orientation has not been changed. The bounding boxes of the individual chunks are shown as red rectangles.

In each chunk, game fences were manually labeled using CloudCompare (2019). In order to efficiently divide each chunk into the blocks required to train PointNet, all chunks were rotated so that they were aligned with the coordinate axes in such a way that is shown in Figure 5.

After rotation, the point cloud was divided into a grid of 1×1-meter blocks and the number of points in each block was either up or down sampled so that each block contained exactly 256 points, which was close to the average point density of the point cloud. PointNet was then trained using cross validation between the five groups so that all topographies were always present in the training and validation pools.

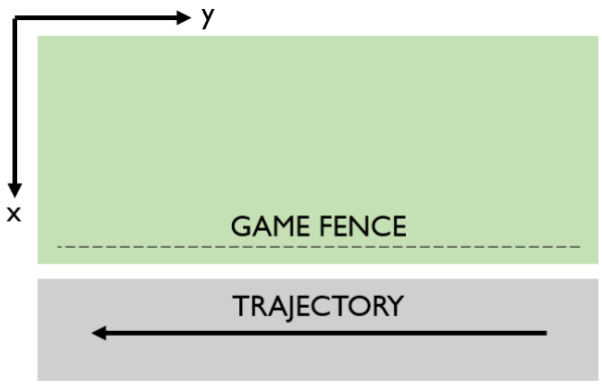

Figure 5. Orientation of point clouds used for PointNet. The grey area represents the road and the green area the roadside region.

2.3 RESULTS

The image segmentation was evaluated using precision, recall and Cohen's kappa coefficient K (Cohen, 1960):

$$Precision = \frac{TP}{TP+FP}\ , \quad Recall = \frac{TP}{TP+FN}\ , \quad \mathrm{K} = \frac{p_o - p_e}{1 - p_e}$$

where TP is true positive, FP is false positive, and FN is false negative. p_o is the observed agreement and p_e is the chance agreement. Precision is a measure of how many of the extracted features are of the sought-after class, recall is a measure of how many of the sought-after features were extracted, and K is a measure of how well the classifier performed in comparison to a random classifier. The numerical results shown in Table 1 are the average values from 5 separate training sessions with randomly sampled training and validation pools.

Table 1. Numerical results from the FCN training. The precision and recall have been rounded to whole percent, and the true positives (TP), false positives (FP), and false negatives (FN) are the number of pixels.

Precision	**Recall**	**TP**	**FP**	**FN**	p_o	p_e	**K**
95%	87%	57719542	3347501	8338351	0.989	0.899	0.89

Figure 6 shows two samples from the validation pool and the predictions made by the FCN for the respective samples. In the left sample, both the poles and the wiring of the game fence are clearly visible in the image, and in the right sample, only the poles are visible. It is clear that the prediction for the left sample is better, but the FCN still manages to predict a large portion of the game fence in the right sample even though the wiring is not visible.

Figure 6. Two samples from the validation pool and their respective predictions. The red color indicates pixels that have been identified as game fence by the FCN.

The final point classification was evaluated against a manual classification and point-wise precision and recall were computed. No kappa coefficient was computed for the points as the large number of true negatives would cause the kappa coefficient to be very close to 1 regardless of the results. The results from presented in Table 2.

Table 2. Precision and recall (rounded to whole percent) together with number of true positives (TP), false positives (FP), and false negatives (FN) after the first and second stage of IBPCS. Total number of points in the input data sets are shown in brackets.

Data set	Precision (%)	Recall (%)	TP	FP	FN
1st stage (160 000 000)	14	100	89864	569085	0
2nd stage (659 000)	88	99	89386	12150	478

Visual examples of the results are shown in Figure 7. The left column (a) shows the results after the first stage of IBPCS and it clearly explains the low precision in Table 2. The middle column (b) shows the results after the second stage of IBPCS and the third column (c) shows the manual classification.

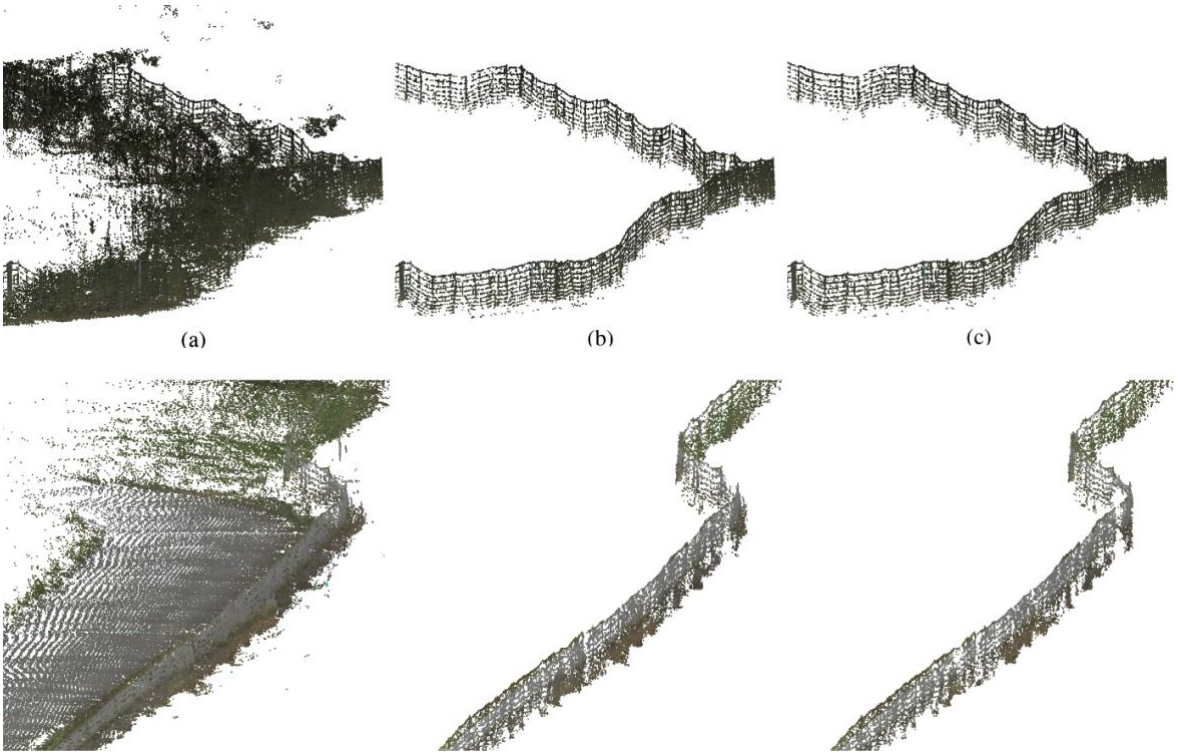

Figure 7. Points remaining after the first stage (a), points remaining after the second stage (b), and ground truth points (c), for two different locations in the data set. The upper row shows a game fence with forest behind it and the second row shows a game fence surrounded by flat ground.

The precision and recall from the different PointNet data sets are shown in Table 3. The results show that the accuracy of the classification is higher in the IBPCS subset compared to the full point cloud.

Table 3. Precision and recall (rounded to whole percent) together with number of true positives (TP), false positives (FP), and false negatives (FN) from PointNet cross validation. Total number of points in the input data sets shown in brackets.

Data set	Precision (%)	Recall (%)	TP	FP	FN
Full point cloud (12 223 000)	80	70	62958	15446	26275
IBPCS subset (659 000)	90	75	66904	7195	22780

Visual examples from the PointNet applications are shown in Figure 8 and Figure 9.

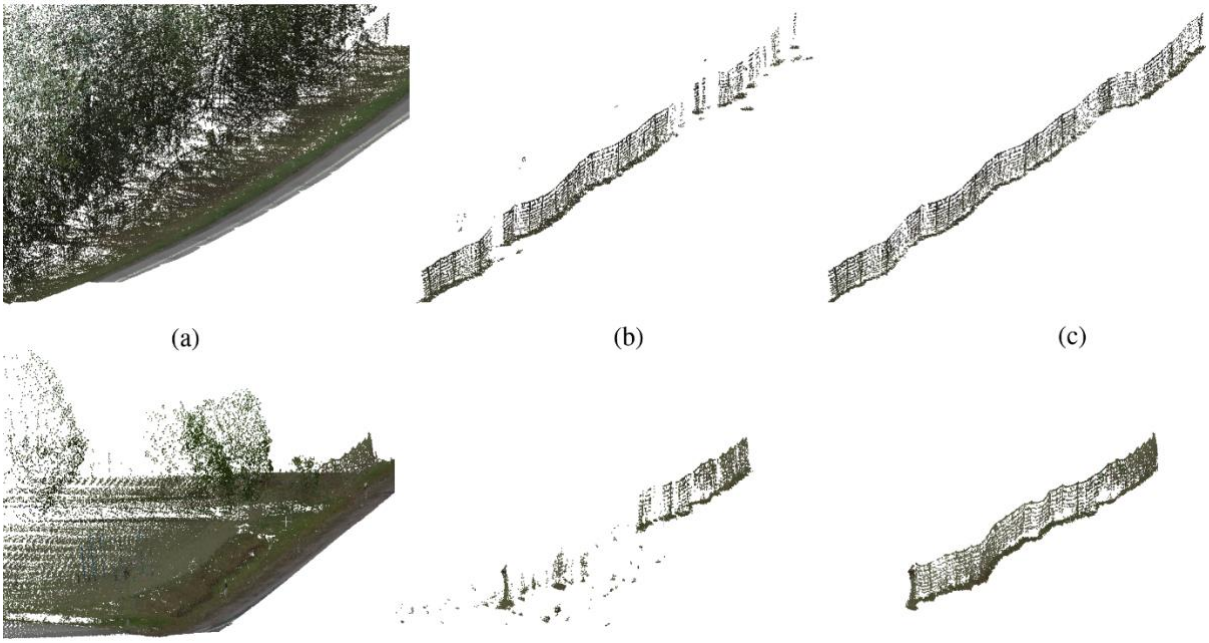

Figure 8. Example results from the PointNet cross validation of the full point cloud. Input data to the left (a), PointNet prediction in the middle (b), and manually classified ground truth to the right (c). The top row shows an area with forest, and the bottom row shows an area with flat ground.

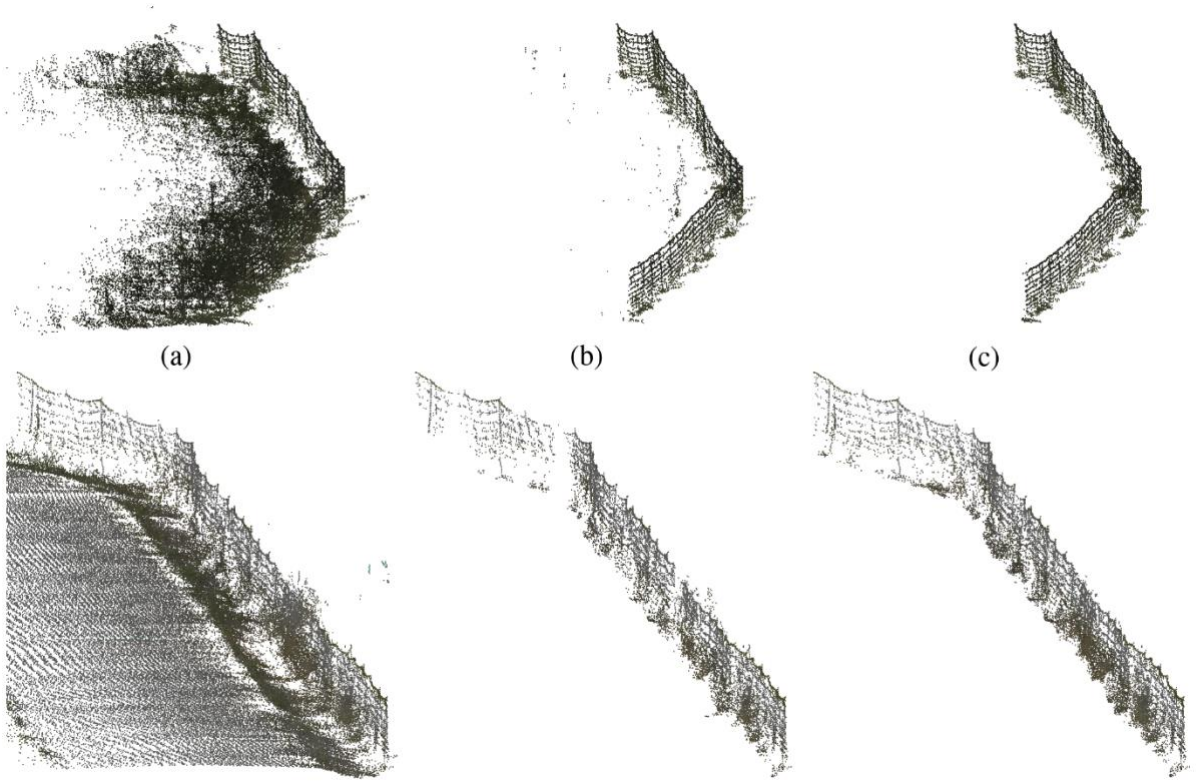

Figure 9. Example results from the PointNet cross validation of the IBPCS subset. Input data to the left (a), PointNet prediction in the middle (b), and manually classified ground truth to the right (c). The top row shows an area with forest, and the bottom row shows an area with flat ground.

3. COMPUTATIONAL EFFICIENCY

There are many factors influencing the computational cost of the two methods. The horizontal extent of the point cloud, the physical block size, and the number of points per block will affect the cost of using PointNet, while the number of images, the image resolution, the frequency of the sought-after objects, and the choice and implementation of the second stage algorithm will influence the cost of IBPCS. However, this does not mean that there is nothing to say regarding their relative efficiency. The first stage of IBPCS creates a subset of the initial point cloud, and if it is quicker to create this subset and apply an algorithm to it than it is to apply the same algorithm to the full point cloud, it would mean that IBPCS can reduce the computational cost over the entire data set. In equation form, this comparison can be written as:

$$C_S + C_T + C_{A(S)} < C_{A(F)}$$

where C_S is the cost of segmenting all images, C_T is the cost of transferring the pixel information from the positive predictions to the point cloud, $C_{A(S)}$ is the cost of applying an algorithm to the resulting subset, and $C_{A(F)}$ is the cost of applying the same algorithm to the full point cloud. In order to investigate this, a theoretical experiment was carried out using values taken from the implementations in this article. It is difficult to compare the cost of applying a geometric algorithm to the full point cloud to applying the same algorithm to a subset of the point cloud, especially since the second stage algorithm used in this article would not deliver the desired results when applied to a full point cloud. Also, most geometric algorithms are to some extent exponential in their complexity, and the methods used for clustering and indexing the point cloud would greatly affect the outcome. However, the complexity of PointNet is truly linear with respect to the horizontal extents of the point cloud, and PointNet could be applied to both a full point cloud as well as a subset. Therefore, the question this analysis tries to answer is: can it be more efficient to create a subset of a point cloud using IBPCS and applying PointNet to this subset compared to applying PointNet to the full point cloud?

The experiment considers a stretch of road that is 70 meters long. The images in the data set were captured roughly 7 meters apart, and there are therefore 10 image pairs covering this stretch, considering images from the two cameras that are facing right in direction of the vehicle. Five 1000×1000-pixel tiles were extracted from each image pair (2 and 3 tiles from the respective cameras) which means that there is a total of 50 image tiles for this stretch of road. The average width of the point cloud, assuming that there are trees on both sides of the road, is roughly 70 meters, which corresponds to 4900 blocks. If there were flat ground on both sides of the road, the width of the point cloud would be greater. The subset created by IBPCS was on average 6 meters wide, which in turn corresponds to 420 blocks. The approximate times required for the different computations are shown in Table 4.

Table 4. Computation times for different operations. All times are estimated averages.

Operation	**Time (s)**
Segmenting one image tile with FCN	0.33
Segmenting 50 image tiles with FCN	16.5
Projecting points for one image	0.50
Projecting points for 20 images	10.00
Processing one block with PointNet	0.01
Processing 420 blocks with PointNet	4.20
Processing 4900 blocks with PointNet	49.00

This means that the total computation time for IBPCS where PointNet is used as its second stage is 16.5 + 10 + 4.2 = 30.7 seconds, while applying PointNet directly would take 49 seconds. This is shown visually in Figure 10. It is possible that the extent of the point cloud could be limited in such a way that only the right side of the road and only the points closer than e.g., 15 meters from the scanner were considered by filtering the raw data from the scanner. This would mean that there are now only 1050 blocks for PointNet to process, and this would take 10.5

seconds. If the raw data from the scanner is not available, such a limitation would have to be computed from the trajectory of the car using e.g., the known camera positions.

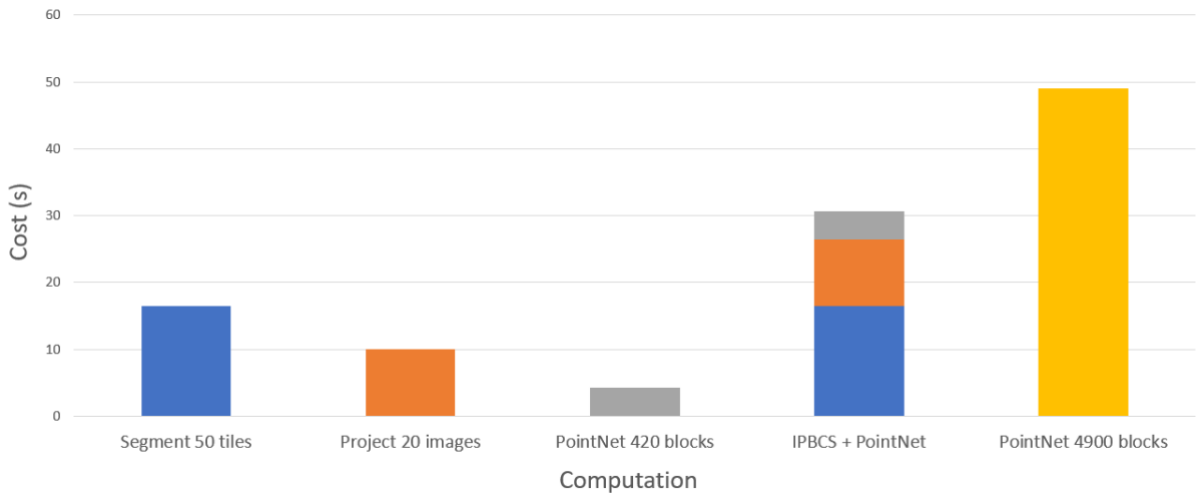

Figure 10. Bar chart showing the computational costs of the different computations. Filtering the point cloud using IBPCS before applying PointNet is more efficient than applying PointNet to the full point cloud.

This example shows us that using IBPCS with PointNet as its second stage can in fact be quicker than applying PointNet to the full point cloud. It also shows that if the extent of the point cloud can be limited in through other means, using PointNet without the first stage of IBPCS can also be the quicker alternative. In this scenario, it is assumed that the game fence is continuous throughout the area, which would rarely be the case when scanning larger road networks. For the cases where the sought-after objects are not continuous in the entire data set, the computational efficiency of IBPCS improves. Only the images that contain positive pixels would have to be considered during perspective projection, and the ratio between the size of the exported subset and the distance driven would decrease.

The results from this experiment do not include the computations required for cropping and stitching images or for splitting the point cloud into blocks. In addition, changes to the factors described in the beginning of the section heavily affects the outcome. For example, reducing the size of the image tiles from 1000×1000 pixels to 500×500 reduces the segmentation time from 0.33 to 0.09 seconds, and increasing the number of points per block from 256 to 4096 increases the computation time from 0.01 to 0.025 seconds.

4. DISCUSSION

The task and data set used in this article are limited in both size and scope, and one should be careful to draw too strong conclusions from the presented results. Nonetheless, the results do show that IBPCS manages to identify a challenging object type with higher accuracy and efficiency when compared to PointNet. The chosen scenario is challenging mainly due to two reasons: a majority of the points in the subset created through perspective projection do not belong to the sought-after object, and since the game fence is continuously present, all images have to be projected to the point cloud. Most other object types, given that they are more solid in their nature, will be easier to identify and extract, and if the sought-after objects are more infrequent, the efficiency of IBPCS will increase. It typically requires less effort to annotate training data in 2D images compared to 3D point clouds, and the availability of pre-trained 2D CNNs makes training a neural network for image segmentation a relatively easy task.

An obvious weakness of the IBPCS is that the second stage depends on the object type. The algorithm used in this article to separate game fences from trees is somewhat elaborate, but still much simpler than any algorithm that could be applied to the full point cloud. For many object types, the second stage algorithm would likely consist of ground point removal and noise filtering, which possibly could allow for approaches that are more generic. IBPCS does not require that a specific neural network is used for image segmentation. Therefore, implementations of the method are flexible and can be updated to follow advances made in the field without having to change other

components of the system. The image segmentation used in IBPCS is largely invariant to driving speed, given that fast enough shutter times can be used, but it is on the other hand dependent on ambient light. Since MLS data are usually not collected during the night or during heavy rainfall, the images will likely be taken under one of two light conditions – sunlight or overcast. The FCN architecture has shown strong performance on data sets consisting of many object classes and where the images have been captured by different cameras and under different conditions (Long et al., 2015; Everingham et al., 2012), so training an FCN that is robust to differences in ambient light is likely not too difficult. The second stage of IBPCS processes the point cloud directly and will therefore to some extent be affected by varying driving speeds and varying point densities.

Comparing the complexity and computational cost of the IBPCS and PointNet is difficult since there are many influencing factors. Without considering hardware, the two most influential factors for the cost of IBPCS are image resolution and point cloud density, while the cost of PointNet almost entirely depends on the point cloud density, or more exactly the number of points per block. The example in Section 3 shows that using the first stage of IBPCS before applying PointNet can be more efficient than applying PointNet to the entire point cloud even in a situation where the sought-after object is continuously present along the road. For continuous objects, it is possible that other filtering methods can be more efficient than IBPCS. Examples of such could be to use raw data from the scanner and filter the point cloud based on distance, or to use the trajectory of the vehicle and selecting areas within a certain distance from the trajectory. What IBPCS can do that these filtering methods cannot is to extract regions based on semantic information and visual appearance. Therefore, in situations where the sought-after objects are not continuously present (e.g., poles, lights, signs, and signals), the benefits of IBPCS become apparent.

This article focuses solely on mobile laser scanning, but IBPCS is compatible with point clouds created through photogrammetry as well. Since all points in a photogrammetric point cloud have a topological connection to the pixels in the images, it is not necessary to use perspective projection to transfer the semantic information. In the case of laser scanning there is a problem of occlusion, where points appearing behind a certain object from the perspective of the camera are assigned the same classification as the object. This does not happen with photogrammetry, as there are no points in the point cloud that are not visible in the corresponding images.

5. CONCLUSIONS AND FUTURE OUTLOOK

This article provides a description of the IBPCS method, an example where it was used to identify roadside game fences, and a discussion of its strengths, weaknesses, and when it is suitable to use. It was shown that IBPCS outperformed PointNet, in both terms of accuracy and efficiency, for a scenario that was not geared towards the strengths of IBPCS.

The strength of IBPCS is that it has close to linear complexity with respect to distance covered, and that it can identify relevant regions of based on their visual appearances in images. This in turn drastically reduces the number of points that have to be considered in the object identification and it allows much simpler algorithms to be used. Two characteristics of point clouds captured by MLS in rural areas are that they typically cover long distances and that most points often are irrelevant for object identification, and this makes IBPCS a suitable method for the task.

The main weakness of IBPCS is the second stage processing. Choosing and tuning an appropriate algorithm can be very dependent on the type of object and the density of the point cloud. Going forward, it would be valuable to investigate if the second stage processing techniques can be generalized, making the method more robust. For example, it could be possible that most objects with a solid appearance can be extracted by simply removing ground points and noise. This should be tested using data captured from different geographic regions, topographies, and MLS systems in order to identify algorithms that are robust to such variations.

The academic contribution of this work is a novel method for object identification in MLS data that uses images as its primary data source. The method takes advantage of data that often is captured but rarely used, and this article shows how this data can be used to create a subset of a point cloud, limiting both the number of points and the semantic content. The practical contribution of this work is an efficient method capable of identifying roadside objects that can be used for infrastructure documentation. Thanks to transfer learning and the relative ease of creating image training data, the method is easy to implement and use in practice.

ACKNOWLEDGMENT

The authors would like to acknowledge the support by the Swedish Transport Administration, Grant No. FUD 6240.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/about/bib. Last accessed 2019-12-10.

Arcos-García, A., Soilán, M., Alvarez García, J. A., and Riveiro, B. (2017). Exploiting synergies of mobile mapping sensors and deep learning for traffic sign recognition systems. *Expert Systems with Applications*, 89:286 – 295.

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M. and Savarese, S. (2016). 3D semantic parsing of large-scale indoor spaces. In proceedings: *IEEE International Conference on Computer Vision and Pattern Recognition 2016.*

Ballard, D. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122.

Che, E., Jung, J., and Olsen, M. J. (2019). Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review. *Sensors*, 19(4).

Chen, L., Lin, K. X., Siu, M. F., Wang, Y. H., Chan P. C., and Lau, C. F. (2019). Classification of construction trade and quantification of work efficiency using posture recognitions and deep neural networks. In proceedings: *CIB World Building Congress 2019*. Hong Kong.

Chen, L., Wang, Y. H., and Siu, M. F. (2020). Detecting semantic regions of construction site images by transfer learning and saliency computation. *Automation in Construction*, 114:103185.

Chollet, F. (2015). Keras. https://github.com/fchollet/keras. Last accessed 2019-12-10.

CloudCompare (version 2.9.1) [GPL software]. (2019). Retrieved from http://www.cloudcompare.org. Last accessed 2019-12-10.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20 (1), pp. 37-46.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E. and Darrell, T. (2013). DeCAF: a deep convolutional activation feature for generic visual recognition. arXiv:1310.1531 [cs.CV].

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2012). The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascalnetwork.org/challenges/VOC/voc2012/workshop/index.html. Last accessed 2019-12-10.

Geiger, A., Lauer, M., Wojek, C., Stiller, C. and Urtasun, R. (2014). 3D traffic scene understanding from movable platforms, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36 (5), pp. 1012-1025.

Guan, H., Li, J., Cao, S., and Yu, Y. (2016). Use of mobile LiDAR in road information inventory: a review. *International Journal of Image and Data Fusion*, 7(3):219–242.

Guan, H., Yu, Y., Ji, Z., Li, J. and Zhang, Q. (2015). Deep learning-based tree classification using mobile LiDAR data. *Remote Sensing Letters*, 6 (11), pp. 864-873.

Halvorsen, I. (2015). [Asset Database – Stakeholder analysis regarding data needs]. Swedish Transport Administration/TRV 2013/24513 (in Swedish).

Jones, E., Oliphant, T., Peterson, P., et al. (2001). SciPy: Open source scientific tools for Python. http://www.scipy.org/. Last accessed 2019-12-10.

Kong, S. and Fowlkes, C. C. (2018). Pixel-wise attentional gating for parsimonius pixel labeling. arXiv:1805.01556 [cs.CV].

Krizhevsky, A., Sutskever, I. and Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. In proceedings: *Advances in Neural Information Processing Systems*, 25, pp. 1090-1098.

LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *Nature*, 521, pp. 436-444.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 396– 404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Long, J., Shelhamer, E. and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 640-651.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Point Grey. (2017). Ladybug 5 specification. https://www.ptgrey.com/support/downloads/10150. Last accessed 2019-12-10.

Pu, S., Rutzinger, M., Vosselman, G. and Oude Elberink, S. (2011). Recognizing basic structures from mobile laser scanning data for road inventory studies. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66 (6), pp. 28-39.

Qi, C. R. (2019). PointNet GitHub repository. https://github.com/charlesq34/pointnet. Last accessed 2019-12-10.

Qi, C. R., Su, H., Kaichun, M., and Guibas, L. (2017). PointNet: Deep learning on point sets for 3D classification and segmentation. arXiv:1612.00593 [cs.CV].

Razavian, A. S., Azizpour, H., Sullivan, J. and Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In proceedings: *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Columbus, OH, pp. 512-519.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. and Li, F. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3), pp. 211-252.

Salakhutdinov, R. & Hinton, G. (2009). Deep Boltzmann machines. In proceedings: *International Conference on Artificial Intelligence and Statistics*, pp. 448-455.

scikit-learn (2018). DBSCAN. https://scikit-learn.org/stable/modules/clustering.html#dbscan. Last accessed 2019-12-10.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 [cs.CV].

Soilán, M., Riveiro, B., Martínez-Sánchez, J. and Arias, P. (2016). Automatic road sign inventory using mobile mapping systems. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLI-B3, pp. 717-723.

Uggla, G. (2019). Classification and object reconstruction in point clouds using semantic segmentation and transfer learning. In proceedings: *CIB World Building Congress 2019*. Hong Kong.

Uggla, G. and Horemuz, M. (2018). Geographic capabilities and limitations of Industry Foundation Classes. *Automation in Construction*, 96:554–566.

Vechersky, P., Cox, M., Borges, P. and Lowe, T. (2018). Colourising point clouds using independent cameras, *IEEE Robotics and Automation Letters*, 3 (4), pp. 3575-3582.

Yang, G., Zhao, H., Shi, J., Deng, Z., and Jia, J. (2018). Segstereo: Exploiting semantic information for disparity estimation. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision – ECCV 2018*, pp. 660-676, Cham. Springer International Publishing.

Yu, Y., Li, J., Guan, H., Jia, F. and Wang, C. (2015). Learning hierarchical features for automated extraction of road markings from 3-D mobile LiDAR point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8 (2), pp. 709-726.