SIMULATING LIDAR TO CREATE TRAINING DATA FOR MACHINE LEARNING ON 3D POINT CLOUDS

J. Hildebrand*, S. Schulz, R. Richter, J. Döllner

Hasso Plattner Institute, University of Potsdam, Germany
(justus.hildebrand, sebastian.schulz, rico.richter, juergen.doellner)@hpi.uni-potsdam.de

Commission IV, WG IV/9

KEY WORDS: LiDAR, 3D Point Clouds, Data Synthesis, Machine Learning, Deep Learning, Semantic Segmentation

ABSTRACT:

3D point clouds represent an essential category of geodata used in a variety of geoinformation applications. Typically, these applications require additional semantics to operate on subsets of the data like selected objects or surface categories. Machine learning approaches are increasingly used for classification. They operate directly on 3D point clouds and require large amounts of training data. An adequate amount of high-quality training data is often not available or has to be created manually. In this paper, we introduce a system for virtual laser scanning to create 3D point clouds with semantics information by utilizing 3D models. In particular, our system creates 3D point clouds with the same characteristics regarding density, occlusion, and scan pattern as those 3D point clouds captured in the real world. We evaluate our system with different data sets and show the potential to use the data to train neural networks for 3D point cloud classification.

1. INTRODUCTION

3D point clouds represent objects, environments, and spatial phenomena and have been established as a universal category of geodata. They can be captured with remote sensing techniques and are used to create spatial digital twins of our physical world. The data acquisition is well established and can be performed on different scales for indoor and outdoor environments (Figure 1) with platforms such as aircrafts, UAVs, vehicles and terrestrial scanners. The use of 3D point clouds for application-specific tasks often requires a classification. This means that 3D point clouds must be enriched with semantics and each point must be assigned to a surface category (e.g., ground, vegetation, building) or object class (e.g., sign, pole, furniture, tree).
Traditional classification approaches analyze the 3D point cloud structure based on predefined rules and parameters according to the 3D point cloud characteristic (e.g., density, distribution). In recent years, machine learning (ML) and deep learning (DL) approaches for the classification of 3D point clouds have been developed as promising alternatives. They do not require parameterization and configuration by the user. However, they require already classified data as training data. This data should represent as closely as possible the data to be classified. The generation of comprehensive, high quality and for the application domain suitable training data is usually a manual and time-consuming process.
In this paper, we present a method to automate the process of generating semantically enriched 3D point clouds that are suitable for training. 3D models with semantics information are used to simulate a LiDAR scan to produce 3D point clouds that have almost identical properties as in real acquisition campaigns. The *virtual LiDAR scan (vLiDAR)* allows to determine semantic information per point from the scanned 3D object. Hence, the output data can be used directly for training neural networks. This in turn can be used to classify 3D point clouds of real environments. An exemplary high-level vLIDAR workflow is shown in Figure 2.

1.1 Background

LiDAR is a well-established technology for capturing objects, structures, and assets of the real world. By measuring the time of flight of a rapidly firing laser beam LiDAR scanners are able to sample hundreds of thousands of points per second from an environment with high precision, often with an error margin of a few millimeters. Scanners can be mounted on various vehicles and platforms. From handheld laser scanners used to scan individual objects (Otero et al., 2020), over car-mounted scanners able to scan road environments (Haala et al., 2008), to plane-mounted scanners that can scan thousands of hectares from an aerial perspective (Yan et al., 2015). LiDAR is used to capture environments at any scale with high levels of precision. Hence, it is used in many different industries, such as urban planning (Hu et al., 2003), construction(Wang et al., 2015), robotics (Bansal et al., 2011), agriculture (Rosell et al., 2009), and many others.
Depending on the used scan platform and the type of scan conducted, the resulting 3D point clouds have several properties that are specific to the scenario at hand. Vehicle-mounted scanners (i.e., mobile mapping) usually rotate on a single plane, relying on the movement of the vehicle to capture the entire environment. This results in 3D point clouds with a lower density when the car is moving faster. Stationary scanners on the other hand create very high point densities close to the scanner position but a density drop-off with increasing distance to the device.
Whenever a scan is not aiming to fully cover the surface of an object, the 3D point clouds usually contain shadows and occlusions. Objects close to the scanner are represented with a high point density, as they cover a large part of the scanner's field of view, and objects far away are sampled with a lower point density for the same reason. Reflective surfaces such as glass can lead to artifacts where reflected objects appear behind the reflecting surface. Some of these scan properties can be seen in Figure 1.
The high data resolution, large-scale and area-wide data ac-

* Corresponding author

(a) Mobile mapping scan recorded in Essen, Germany. The 3D point cloud shows typical properties such as scan line patterns and occlusions.

(b) Terrestrial scan recorded in an office. Shadows, reflections, and distance related point density differences can be seen.

Figure 1. Typical real-world LiDAR scans from outdoor and indoor environment colored in grey-scale based on intensity data.

quisition (e.g., entire cities and countries) as well as regular data acquisition (e.g., daily, weekly, annually) result in massive amounts of data (e.g., Terabytes or even Petabytes). An automated analysis and classification is essential to make use of the data. A promising approach to solve several common 3D point cloud analysis problems, i.e., object detection, semantic segmentation and anomaly detection is the use of deep neural networks (DNNs). Several works have achieved improving results in recent years, e.g., PointNet (Qi et al., 2017a), PointNet++ (Qi et al., 2017b), DGCNN (Wang et al., 2019), KPConv (Thomas et al., 2019), and PointMLP (Ma et al., 2022).

DNNs require large amounts of training data to perform well in real-world scenarios. The mentioned networks were trained and evaluated on either small or unrealistic (in comparison to real scans) benchmark data sets, such as ModelNet40 (Wu et al., 2015), ShapeNet (Chang et al., 2015), or SemanticKITTI (Behley et al., 2021). ModelNet40 and ShapeNet consist of several thousand instances of standalone mesh models of objects from different classes, each uniformly sampled into 3D point clouds. SemanticKITTI is a real-world mobile mapping data set, consisting of several kilometers of mobile mapping LiDAR scans recorded in and around the city of Karlsruhe, Germany. While the variety of ModelNet40 and ShapeNet is not negligible, the data sets do lack realism regarding sampling patterns, as well as scene complexity encountered in real life. Since SemanticKITTI is based on real-world data, it does present all the properties of a real-world scan, however, it is still limited in size, and the labeling of the data set alone required over 1.700 hours of work. Majgaonkar et al. (2021) argue that much larger data sets are required still, especially with an increasing complexity of scenes. Further, the learnings from classification tasks in street environments are not easily transferable into other domains such as indoor or aerial scans, as vastly different geometry needs to be analyzed. To process data from other spaces, a new dataset of similar size would have to be labeled, which seems uneconomical based on experience from the SemanticKITTI project. Other machine learning domains such as 2D image recognition have successfully circumvented the problem of not having enough real-world training data by using synthetic data instead (Georgakis et al., 2017, Su et al., 2015). In 3D point cloud analysis, very few synthetic data sets have aimed to create realistic 3D point clouds.

This paper addresses the problem of creating suitable training data for deep learning and machine learning on 3D point clouds. It introduces concepts and techniques to generate suitable training data for different application fields in terms of quality and quantity. The generated synthetic 3D point clouds are already labeled and correspond to real acquisition data in terms of their characteristics (e.g., resolution, scan pattern, artifacts), thus ensuring and optimizing the quality of training and prediction.

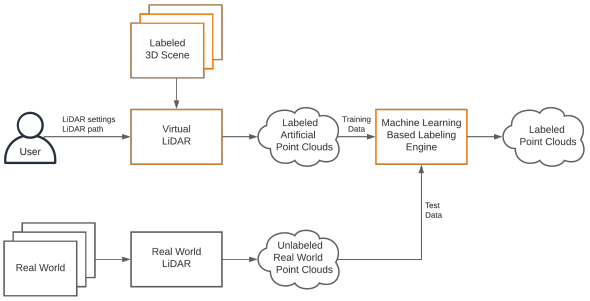


Figure 2. Machine learning based 3D point cloud classification workflow using vLIDAR (orange components).

1.2 RELATED WORK

Synthetic data has successfully been used as training data in multiple machine learning domains. Georgakis et al. superimpose 2D cutouts from labeled 2D image data sets onto photos of indoor scenes, virtually placing labeled objects into the scenes. The generated synthetic data was used in combination with real data to train state-of-the-art object detectors, which showed an increase in performance compared to the object detectors trained solely on real-world data.

Su et al. (2015) synthesize training data from 3D models for a neural network capable of object viewpoint estimation. In their paper, they use the annotated 3D model data set ShapeNet (Chang et al., 2015) to render 2D images combined with the corresponding viewpoint attached to each render. Their work showed increased performance over other state-of-the-art approaches for viewpoint estimation with the large amount of generated training data.

Several works propose different approaches to generate synthetic data for 3D point cloud analysis research. ShapeNet and ModelNet40 are frequently used benchmark datasets that consist of large collections of stand-alone 3D mesh models. In state-of-the-art research, 3D point clouds are generated from the data sets by uniformly sampling points from the object surfaces. While the object counts of the data sets are large enough to perform basic performance benchmarks on novel analysis approaches, the used uniform sampling stands in direct contrast to the complex properties of real-world LiDAR measurements. Further, real-world scenes consist of more than a singular object. Therefore, these benchmark data sets and the associated uniform sampling do not lend themselves to real-world LiDAR scan analysis. Other works aim to generate 3D point clouds that

are more similar to real-world LiDAR scans.

PreSIL (Hurl et al., 2019) implements a LiDAR simulation in the popular video game Grand Theft Auto V (GTA V) to generate 3D point clouds for autonomous driving applications. The PreSIL data set uses the advantage of GTA V's large virtual open world consisting of urban and rural scenery to generate diverse, yet realistic, pre-labeled 3D point clouds of outdoor road scenes. While GTA V provides a highly detailed realistic environment, the approach is strongly limited by the size of the virtual world, as there is no easy way of expanding the pre-existing game world to generate even more diverse data.

HELIOS++ (Winiwarter et al., 2022) implements a LiDAR simulation framework designed to realistically simulate real-world laser scanners with full waveform simulation. The framework is designed to simulate LiDAR surveys, to aid with planning and designing terrestrial, mobile mapping, and aerial LiDAR surveys in the real world, and to visualize survey simulations in real-time 3D. It provides close integration with GIS software by providing a plugin for the QGIS software[1] allowing for GUI-based survey planning and execution. While HELIOS++ is able to perform quite realistic LiDAR simulations, its focus does not appear to be on generating training data for machine learning experiments, and its ability to deal with arbitrary input data seems lacking, only providing direct support for OBJ and GeoTIFF models.

BlenSor (Gschwandtner et al., 2011) is a Blender add-on that does basic LiDAR simulation in Blender. It does not work with modern versions of Blender, so a new implementation is required.

2. VLIDAR

Labeled real-world data as training data is not available in scales that are large enough to train networks robust enough to perform well in most real-world scenarios. To solve this problem of data scarcity, we propose a new method to synthesize realistic 3D point clouds containing extensive semantic information. For this, we first define several goals that our method aims to fulfill:

1. Imitate real-world scan properties, such as distance-dependent point density, scan patterns, shadows, and occlusions.
2. Enrich 3D point clouds with additional data that can not be sampled in the real world, i.e., semantic data, surface normals, or segmentation data.
3. Integrate the solution closely into existing 3D modeling workflows, to enable rapid iteration of virtual environment design, as well as compatibility with complex scenes.
4. Provide robust support for different formats used in 3D modeling (e.g., OBJ, IFC, FBX) to enable use of diverse modeling options.
5. Synthesize 3D point clouds at high speeds even in complex scenes, to encourage generation of large data sets.

In this section, we explain how our solution fulfills the stated goals by implementing a virtual LiDAR (*vLiDAR*) as an add-on to the open-source 3D computer graphics tool Blender[2].

2.1 LiDAR Simulation

To generate 3D point clouds containing realistic scan properties such as occlusions, shadows, and distance-dependent point densities, we choose to closely imitate time-of-flight LiDAR scanners. Where real-world LiDAR uses laser beams to precisely sample points from the environment, we use ray casting algorithms to achieve the same effect in virtual scenes. To do this, a vLiDAR scanner is first placed in the virtual 3D scene which is represented as a collection of triangle meshes. The vLiDAR then casts a ray into the scene to perform intersection tests with the geometry of the objects. The closest intersection is chosen as sample point and stored as a point with corresponding coordinates. Additional per-point attributes such as semantic class or object ID of the hit object, as well as the surface normal of the hit surface can optionally be stored as well. After the ray cast, the vLiDAR scanner is rotated and moved according to the scanner's rotation and movement speeds, before conducting the next ray cast. By repeatedly sampling the scene in this manner, a realistic 3D point cloud representing the scene is created with the same principles of operation of a real-world LiDAR scanner.

In Figure 3 multiple scenarios for vLiDAR scans are presented. The figure shows the use cases of stationary scanning an indoor scene, using mobile mapping to scan an urban road environment, and using aerial laser scanning to scan an entire urban area from a birds-eye perspective. For each use case, an overview of the scanned 3D model is presented, next to an overview of the resulting 3D point clouds and a closeup of the scan results to showcase the realistic scan properties of the synthetic 3D point clouds, i.e., shadows and scan patterns. The images presenting the 3D models also contain the graphical representation of the vLiDAR scanner, or the path of the moving scanner respectively. The colorization of the 3D point clouds represents the semantic data encoded in the original 3D meshes, sampled for each point. Each color represents a different class in the scan.

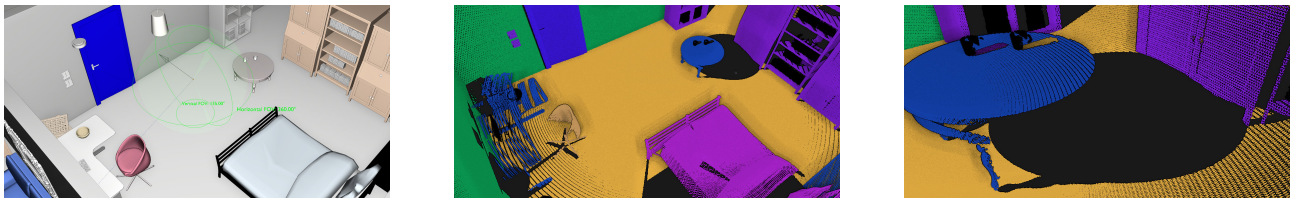
2.2 Integration into Blender

The vLiDAR is implemented as an add-on to the commonly used open-source 3D computer graphics tool Blender. The tool is actively used in the field of 3D modeling, and as such we see it as a good fit for integrating the vLiDAR to close the gap between modeling workflows and 3D point cloud synthesis. Blender has many qualities that aid in fulfilling the stated design goals. It has a suitable representation of 3D models, both in its user interface and in its internal data representation, and can import data from many different file formats, such as OBJ, Alembic, FBX, IFC, and others. Further, Blender has a Python scripting environment. Through an API named bpy, programmers have full access to all of the data accessible through Blender's user interface. The API also provides a lot of functions to directly work with 3D data, and even modify Blender's user interface. Finally, custom data can be attached to objects through the API, allowing for easy addition and access of semantic information, surface normals, and other data that would be of interest for 3D point cloud generation. The combination of its sophisticated 3D data modeling capabilities, the open-source nature of the project, and its powerful Python scripting environment make Blender a good fit to be used as a framework for the vLiDAR project.

2.3 vLiDAR Design

The vLiDAR add-on is designed to give its users the freedom to simulate LiDAR surveys with high variability. As such, it implements different scan modes that cover the most common mounting platforms, as well as the most common scan beam rotation patterns. Scanners can be simulated to be mounted on a mobile mapping vehicle, fully rotating on a single axis to cover

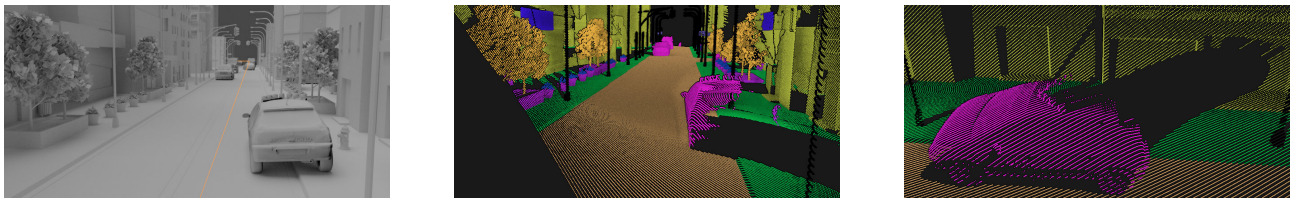
[1] `https://www.qgis.org/`
[2] `https://www.blender.org/`



(a) Indoor scene with visible stationary vLiDAR scanner overlay.

(b) Stationary vLiDAR scan of indoor scene with semantic information.

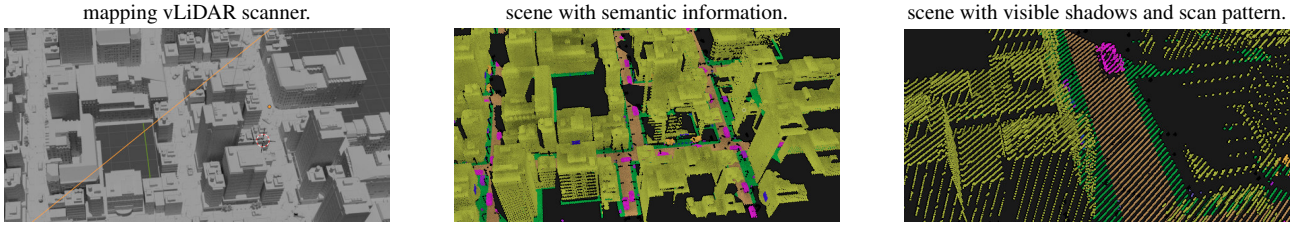
(c) Stationary vLiDAR scan of indoor scene with visible shadows and scan patterns.



(d) Urban scene with visible path of mobile mapping vLiDAR scanner.

(e) Mobile mapping vLiDAR scan of urban scene with semantic information.

(f) Mobile mapping vLiDAR scan of urban scene with visible shadows and scan pattern.



(g) Large urban scene with aerial vLiDAR flight path visible.

(h) Aerial vLiDAR scan of large urban scene with semantic information.

(i) Aerial vLiDAR scan of large urban scene with visible shadows and scan pattern.

Figure 3. Synthetic 3D scenes for indoor (top), mobile mapping (middle) and outdoor (bottom) scenario. 3D models (left) as input are used to generate 3D point clouds with semantics (middle). Detailed view of vLiDAR characteristics is illustrated (right).

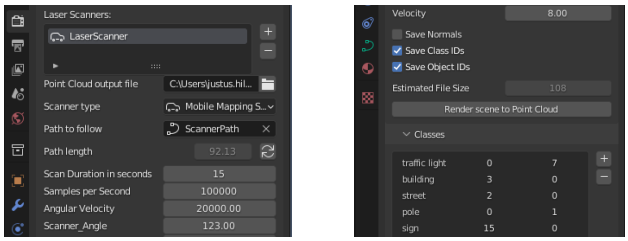


Figure 4. The vLiDAR user interface as part of the Blender GUI. The pictured classes are linked to a two-level class ID system.

the vehicle's surroundings, or mounted on a plane, scanning downwards and panning the scan beam from left to right. Further, stationary scanners can be simulated, scanning in a modifiable field of view of up to 360 degrees left to right and 170 degrees up and down, similar to real-world stationary scanners. The add-on can be extended with new rotation patterns to simulate further types of LiDAR scanners.

All scanner types can be attached to a user-defined path to define the movement of the scanner during the scan simulation. This movement, as well as the right rotational patterns, is key for simulating realistic LiDAR behavior to produce realistic 3D point clouds.

The user is given full control over many different parameters of the vLiDAR scanners. They can modify rotation speeds, sampling rate, scan duration, as well as movement speed when a path is to be followed. With these options, users can directly simulate most real-world scanner devices. All controls are fully integrated into Blender's user interface to stay as close as possible to the modeling workflow (Figure 4). The user can assign semantic class information to the 3D models, which can then be appended to the sampled points during scan time. Further metadata such as object affiliation and surface normals can also be extracted by the vLiDAR, to allow for generating information-rich data sets. Scan results are stored in CSV files with the raw XYZ, semantic class, normal, and object ID data as columns to allow for compatibility with most state-of-the-art 3D point cloud processing software. This can be easily extended to support other formats such as PLY, LAS/LAZ, HDF5, or DBMS in the future.

To be able to generate large amounts of synthetic 3D point clouds as training data for machine learning approaches, the vLiDAR needs to scan scenes with good performance. Since the vLiDAR is required to perform millions of ray casts for one scan simulation, it is important that these ray casts are performed as efficiently as possible. For this, it is inevitable to utilize a high-performance data structure as basis for the ray casting algorithm. Dos Santos et al. (2014) compare several high-performance data structures that can be used in ray traversal algorithms, finding that octrees and bounding volume hierarchies (BVHs) are the most suitable structures for ray casting. Octrees perform slightly faster than BVHs when it comes to pure ray casting performance, however, octrees are also much more memory intensive. They conclude that for complex scenes used in non-real-time rendering, BVHs are the more suitable option due to their lighter memory requirements. This goes in line with BVHs being the industry standard being used in state-of-the-art non-real-time rendering engines, such as Blender's Cycles engine.

As such, we opt to use BVHs as the underlying data structure for the vLiDAR's ray casting operations. With the BVH in place, the vLiDAR is able to scan over 100,000 points per second in small scenes with under 500 objects (e.g., indoor scenes), and can still scan over 10,000 points per second in highly complex scenes with more than 5,000 objects. Section 4 contains an in-depth run time performance evaluation of the vLiDAR.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W2-2022-105-2022 | © Author(s) 2022. CC BY 4.0 License.

3. DEEP LEARNING PIPELINE

Deep learning approaches like PointNet and DGCNN require appropriately labeled data sets for the training stage, which ideally is relatively close to the target data in their properties.
Manual labeling of real-world data sets is very tedious and existing labeled data sets might not necessarily fulfill requirements for the intended target data. For example, labeled data sets such as SceneNN and S3DIS (Armeni et al., 2017) are captured using photogrammetric methods, and therefore do not have the precision or artifacts shown in real-world LiDAR scans. They are sometimes not labeled consistently, and their scenes might not always match up with the ones of the target data sets. vLiDAR offers a cheap solution to these issues by supplying realistic, automatically labeled data sets that can be used in training stages at a comparatively small cost.
To demonstrate the feasibility of using vLiDAR data in real-world use cases, we propose a proof-of-concept classification pipeline using DGCNN (Wang et al., 2019) with a newly created vLiDAR data set as training data, and evaluate it against both vLiDAR and real-world LiDAR data sets.

3.1 Training Data Creation

Since the objective is to classify common indoor scenes, a combination of office and apartment rooms is used as a reference. The 3D model is created in Blender. The general structures – walls, windows, doors, stairs – are modeled using Archipack[3], an open-source plugin that is provided with Blender, which includes tools and models for quick and easy creation of architectural models.
The smaller 3D models used to populate the scene come from multiple sources: Most of the 3D models used, especially the furniture, come from IKEA, being published in multiple formats for architectural use cases on the platform Polantis[4]. Further 3D models, especially smaller objects, clutter, and wall paintings were taken from Blend Swap[5], a platform for Blender 3D models licensed under various Creative Commons licenses. A small number of very simple 3D models have additionally been created for this model. This includes objects such as light switches, certain tables, or lamps. The modeled floor for the evaluation has eight rooms (Figure 6).
The resulting 3D point clouds from the vLiDAR implementation contain XYZ and normal data. However, the normal information is not used in the actual training of the model, since doing so would otherwise require the target 3D point clouds to also have normal data. Additionally, since the normal information is generated directly from the model geometry, they do not show the artifacts that would normally arise when generating them from a 3D point cloud, and therefore are not comparable to the normal data that would be present in any real-world target 3D point clouds. Rooms 2, 3, 4, 5, 7 and 8 (Figure 6) are used as training data.

3.2 Evaluation Data

Rooms 1 and 6 (Figure 6) from the vLiDAR data set are used as test data for a direct, best-case comparison.
To evaluate the model using real-world data, the real-world LiDAR scans from the Redwood data set (Park et al., 2017) is used, as shown in Figure 6. Because the original data sets do not have any semantic annotations, the Redwood boardroom 3D point clouds are manually annotated for this evaluation, using the same semantic classes that are used for the vLiDAR data set.
As a current limitation, the data set includes objects that do not exist within the vLiDAR training data, and are therefore likely to cause problems when being classified with the trained models. This includes for example kitchen counters (which are annotated as storage), stucco (which are annotated as part of the walls), and wooden wall panels (which are also annotated as part of the walls).

[3] https://blender-archipack.org/
[4] https://www.polantis.com/ikea
[5] https://www.blendswap.com/

3.3 Classification Pipeline

The overall classification pipeline used for evaluation purposes consists of multiple preprocessing steps, one DGCNN-based semantic classification step, and further postprocessing, as seen in Figure 5.

3.3.1 Preprocessing To bring the 3D point clouds into a uniform format, various preprocessing steps are applied before the actual classification.
Individual scans of the same scene are **merged** into a single 3D point cloud. This way, the shadowing artifacts from (v)LiDAR 3D point clouds are less severe, and there is more spatial context available when sampling points for the classification step. For the vLiDAR data set, scans of each room are merged, resulting in eight 3D point clouds total. Since the individual scans all use the same global coordinate system, this can be done easily by combining the sets of points, without elaborate point cloud registration methods.
Imperfections are added artificially to the input data to both make the input data sets more uniform and prevent overfitting on the properties of vLiDAR data. This is done to compensate for vLiDAR training data sets that are "too perfect", given that the virtual scanner is always perfectly precise and that the 3D models that they are based on might not be modeled to the level of detail that would be required to be completely on par with real-world data. In the implementation of this processing pipeline, this is done by shifting points along the direction of their surface normals which were calculated by the vLiDAR implementation, to generate a rougher surface for specific semantic classes such as beds and sofas. The strength of this shifting is determined by a noise value that is calculated using Perlin Noise for each axis separately. A similar approach could be used to simulate inaccuracies of the scanning platform. However, because of the later applied density reduction and sampling, this was skipped for our pipeline.
(v)LiDAR data sets are often captured with a point density far beyond what is usually necessary for correct classification. Since this additional information usually has a negative effect on the hardware requirements and the performance, the **density** is **reduced** down to a more appropriate level. This is done by subdividing the 3D point clouds using a regular voxel grid and choosing only one representative point from each voxel cell.
While spatial information (and in the case of the training data, the semantic class annotations) is a requirement, it is an important decision which additional point attributes – like color information, intensity values (in case of LiDAR scans), or normal values – to include in the used data sets. This is always a trade-off between potential classification quality, performance, and specialization in regard to potential target data sets. Supplying additional point attributes to the ML-based model might support the decision-making, however, this would require both the training and the target data sets to have these point attributes, to have sufficient variation regarding these attributes, and it might

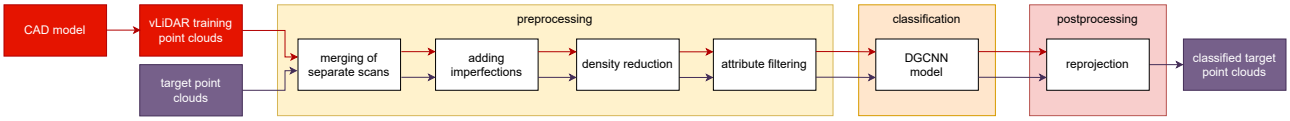


Figure 5. Overview classification pipeline.

Figure 6. From left to right: Overview of created 3D model, two synthetic example scenes, and the Redwood boardroom scene used as real-life comparison.

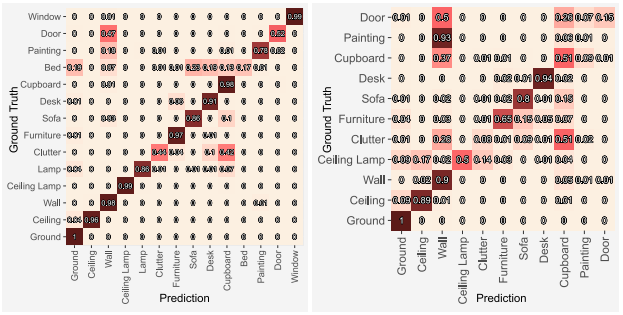
come with additional performance requirements. Therefore, all **attributes** except the XYZ data are **removed** in a last preprocessing step.

3.3.2 Classification The actual classification step is done using a **DGCNN** model trained on the vLiDAR data. DGCNN extracts neighborhood relations between points and their neighbors and can group points in Euclidian and semantic space. As a result of heuristic evaluations, a sampling radius of 1.0 m with 1024 sampled points is used.
DGCNN was chosen since in earlier tests of ours it offered the best combination of quality and performance compared to PointNet and PointNet++. PointNet, while being the fastest of the three, very often showed artifacts in the form of extremely noisy results, especially in relatively crowded areas (e.g., with cabinets or TVs standing directly next to a wall), and needed a relatively large amount of training epochs to reduce this kind of artifacts. While PointNet++ with its hierarchical architecture with different sampling radii in theory has the advantage of using more context of the input 3D point cloud, this did not seem to have a significant influence in the context of these indoor scans, presumably because of the already limited differences in scale of object classes. The actual classification results seemed to be on par with the original PointNet, however with much higher hardware requirements (especially in terms of used VRAM) and much longer processing times. DGCNN seemed to handle the issues with noisy areas the best of the three. Although the network has longer per epoch processing times than PointNet, it produced usable results after fewer epochs compared to PointNet or PointNet++. Therefore, we chose DGCNN for our pipeline.

3.3.3 Postprocessing Since the preprocessing steps apply irreversible changes to the classified data sets, it might be desirable to **reproject** the classification results back onto the original data sets. By applying the semantic class of each points nearest neighbor from the preprocessed and then classified 3D point cloud, the result has both the original point density, all original attributes, etc. and the semantic information from the DGCNN classification step.
For the calculation of quality metrics, this reprojection step is, however, skipped, since this might skew the distribution of points among the semantic classes. If the scanner positions were chosen in a way that some regions of the scene were captured with far higher point counts than other parts, for example, because of overlapping scans, these specific parts would otherwise overproportionally contribute to the overall metrics and lead to a result that would be far less intuitive.

3.4 Results

For the vLiDAR data set, the trained network seems to detect objects of most semantic classes reasonably well, especially the rough structures like walls, ground, and ceiling. Only specific classes were problematic, as can be seen in Figure 7 (a), with specific examples shown in Figures 8 (a) and (b). Beds and sofas seem to be mixed up, likely because of the similarities of some of the models used. Clutter often gets wrongly classified as furniture because there often seem to be slight issues with correctly separating the large, main structures from the clutter laying on top, especially in tight spaces such as cupboards filled with clutter.



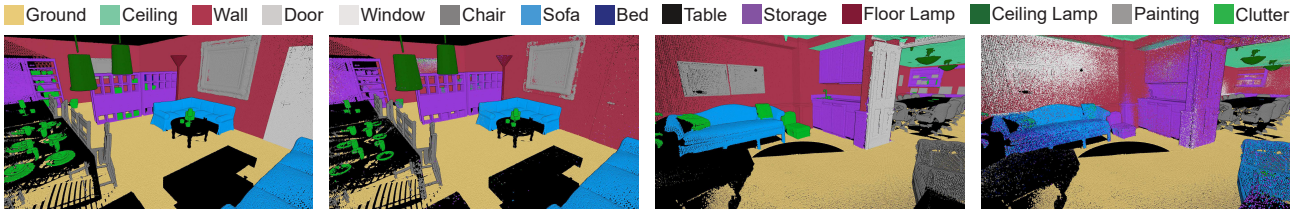
(a) Classification result for the vLiDAR data set.

(b) Classification result for the Redwood data set.

Figure 7. Confusion matrices for the classification results.

The metrics when classifying the real-world data set are noticeably worse than their vLiDAR counterpart, as can be seen in Figure 7 (b), with specific examples shown in Figures 8 (c) and (d). This was expected, given how it differs from the vLiDAR data set. For example, the real-world 3D point clouds contain a kitchen sink, wood paneling on the walls, and a ledge near the wall, all of which do not occur within the vLiDAR data set. While the vLiDAR data set only included ceiling lamps that were hanging from the ceiling, the real-world data set also includes ones that are included as panels within the ceiling, and are therefore wrongly classified as part of the ceiling. Nevertheless, many classes, like ground, ceiling, walls, furniture, and desks, are still detected reliably.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W2-2022-105-2022 | © Author(s) 2022. CC BY 4.0 License.



(a) vLiDAR ground truth. (b) vLiDAR classification result. (c) Real-world ground truth. (d) Real-world classification result.

Figure 8. Classification comparison using both the vLiDAR and the real-world data set.

4. VLIDAR RUN TIME EVALUATION

To evaluate the vLiDAR's scanning performance, two benchmarks are run. The first benchmark evaluates the impact of varying scene complexity, i.e., the number of objects in the scene, and the second benchmark evaluates the impact of varying object complexity, i.e., the number of primitives of each object. The default setup for the benchmarks is a stationary vLiDAR scanner, placed at the origin of the scene, with a horizontal rotational speed of 50,000°/s, and a vertical rotational speed of 10,000°/s. The scanner runs for 10 in-simulation seconds with a sample rate of 100,000 points/s, resulting in a total of 1 million points sampled for each benchmark configuration. All benchmarks are run on an AMD Ryzen 7 5800X. For both benchmarks, the BVH generation time, as well as the scan time is measured, as for highly complex scenes the BVH build time has been not negligible.

4.1 Scene Complexity Benchmark

To measure the impact of scene complexity on the vLiDAR performance, the benchmark configuration is tested with an increasing amount of sphere objects with 400 faces each, placed in an 8 unit radius around the vLiDAR scanner. The benchmark starts with a low-complexity scene of only 5 objects and continually increases scene complexity to up to 5,000 objects to represent a highly complex scene. Figures 9 (a) and (b) show the results of this benchmark.
As shown in Figure 9 (a), there is a logarithmic relationship between the number of spheres present in the scene and the time needed to complete the benchmark. With Blender's current API, only a linear run time was possible, by building object-wise BVH trees. A custom BVH generator for the entire scene was therefore implemented into Blender to reach logarithmic scan time. The vLiDAR is able to scan with over 20,000 points/s even in the extremely complex scene containing 5,000 objects, which in our experience is the upper limit for object count in many virtual scenes.

4.2 Object Complexity Benchmark

To measure the impact of object complexity on the vLiDAR performance, the benchmark configuration is tested with a stable amount of sphere objects with increasing face count, placed in an 8 unit radius around the vLiDAR scanner. The benchmark starts at a low object complexity of 200 faces per sphere, equaling 300,000 faces in the scene in total, and increases to 10,000 faces per sphere, equaling 15,000,000 faces in the scene. Figures 9 (c) and (d) show the result of this benchmark.
Object complexity does not have a big impact on scan performance. As can be seen in Figure 9 (c), the efficiency of the BVH and the associated logarithmic run time increase come into effect. A five-fold face count increase leads to a run time increase of 9 seconds, 4 of those being due to longer BVH generation times when inspecting the run time differences between 1 million and 5 million faces.
The benchmarks show that the vLiDAR is capable of scanning large scenes containing complex objects with good performance, even in scenes with a high complexity of 5,000 objects. They further show that object complexity has a measurable, but small impact on overall vLiDAR scan speed.

5. CONCLUSION

In this paper, we have introduced a system for virtual LiDAR scans to create 3D point clouds with semantics for machine learning applications. Through this, we solved the problem of missing or insufficient amounts of training data. We have implemented a system for scanning virtual 3D objects to create 3D point clouds with typical characteristics and artifacts regarding real on-site acquisition. The vLiDAR is integrated into the 3D software tool Blender to support a variety of common 3D formats and allow easy creation of 3D scenes. It performs fast enough to generate large amounts of training data. We have shown that the training of neural networks with synthetic data is a promising way to classify real-world data. It circumvents the process of manually labeling training data, increases the amount of available training data, and improves the overall quality of machine learning based classification approaches for 3D point clouds. Thus, we made an important contribution to reduce the entry barrier to machine learning approaches for a variety of applications in the geospatial domain. Further research directions could focus on a more physically correct simulation of LiDAR (e.g., pulse, intensity, material information) or a simulation of image-based data acquisition for 3D point cloud creation (e.g., dense image matching). Synthetic point cloud data could be used for various analysis research, such as large-scale outdoor analysis, transfer learning between different data sets, or combining synthetic and real-world training data.

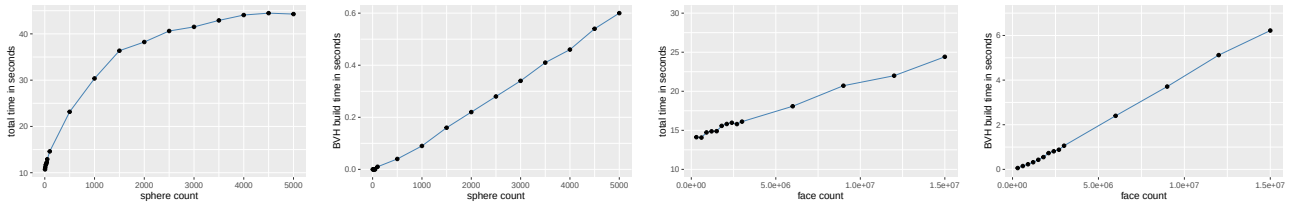
ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their valuable feedback. This work was partially supported by the Federal Ministry of Education and Research (BMBF), Germany (PunctumTube, 01IS18090) and by the Federal Ministry for Digital and Transport (BMDV), Germany (TWIN4ROAD, 19F2210).

REFERENCES

Armeni, I., Sax, S., Zamir, A. R., Savarese, S., 2017. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*.

Bansal, M., Matei, B., Southall, B., Eledath, J., Sawhney, H., 2011. A lidar streaming architecture for mobile robotics with



(a) Total scan time relative to scene complexity.

(b) BVH build time relative to scene complexity.

(c) Total scan time relative to object complexity.

(d) BVH build time relative to object complexity.

Figure 9. vLiDAR benchmark measurements. The data shows that larger scenes do impact performance, but the vLiDAR still performs well on complex scenes with many objects, being able to scan over 20,000 points/s even with 5,000 objects in the scene. Further, BVH build time is negligibly small compared to overall scan time, especially with large scans.

application to 3d structure characterization. *IEEE International Conference on Robotics and Automation*, 1803–1810.

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Gall, J., Stachniss, C., 2021. Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset. *The International Journal on Robotics Research*, 40(8-9), 959-967.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H. et al., 2015. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*.

dos Santos, A. L., Teichrieb, V., Lindoso, J., 2014. Review and comparative study of ray traversal algorithms on a modern gpu architecture. *22nd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 203–212.

Georgakis, G., Mousavian, A., Berg, A. C., Kosecka, J., 2017. Synthesizing Training Data for Object Detection in Indoor Scenes. *CoRR*, abs/1702.07836.

Gschwandtner, M., Kwitt, R., Uhl, A., Pree, W., 2011. Blensor: Blender sensor simulation toolbox. *International Symposium on Visual Computing*, Springer, 199–208.

Haala, N., Peter, M., Kremer, J., Hunter, G., 2008. Mobile LIDAR Mapping for 3D Point Cloud Collection in Urban Areas - A Performance Test. *Proc. of XXI ISPRS Congress, Beijing, China, July 3-11, 2008*, 37.

Hu, J., You, S., Neumann, U., 2003. Approaches to large-scale urban modeling. *IEEE Computer Graphics and Applications*, 23(6), 62–69.

Hurl, B., Czarnecki, K., Waslander, S., 2019. Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. *IEEE Intelligent Vehicles Symposium (IV)*, 2522–2529.

Ma, X., Qin, C., You, H., Ran, H., Fu, Y., 2022. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. *arXiv preprint arXiv:2202.07123*.

Majgaonkar, O., Panchal, K., Laefer, D., Stanley, M., Zaki, Y., 2021. Assessing LIDAR Training Data Quantities for Classification Models. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 46, 101–106.

Otero, R., Lagüela, S., Garrido, I., Arias, P., 2020. Mobile indoor mapping technologies: A review. *Automation in Construction*, 120, 103399.

Park, J., Zhou, Q.-Y., Koltun, V., 2017. Colored point cloud registration revisited. *Proc. of the IEEE international conference on computer vision*, 143–152.

Qi, C. R., Su, H., Mo, K., Guibas, L. J., 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Qi, C. R., Yi, L., Su, H., Guibas, L. J., 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 5099–5108.

Rosell, J. R., Llorens, J., Sanz, R., Arno, J., Ribes-Dasi, M., Masip, J., Escolà, A., Camp, F., Solanelles, F., Gràcia, F. et al., 2009. Obtaining the three-dimensional structure of tree orchards from remote 2D terrestrial LIDAR scanning. *Agricultural and Forest Meteorology*, 149(9), 1505–1515.

Su, H., Qi, C. R., Li, Y., Guibas, L. J., 2015. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. *Proceedings of the IEEE International Conference on Computer Vision*, 2686–2694.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., Guibas, L. J., 2019. Kpconv: Flexible and deformable convolution for point clouds. *Proc. of the IEEE/CVF international conference on computer vision*, 6411–6420.

Wang, J., Sun, W., Shou, W., Wang, X., Wu, C., Chong, H.-Y., Liu, Y., Sun, C., 2015. Integrating BIM and LiDAR for real-time construction quality control. *Journal of Intelligent & Robotic Systems*, 79(3), 417–432.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., Solomon, J. M., 2019. Dynamic graph cnn for learning on point clouds. *ACM Transactions On Graphics (tog)*, 38(5), 1–12.

Winiwarter, L., Esmorís Pena, A. M., Weiser, H., Anders, K., Martínez Sánchez, J., Searle, M., Höfle, B., 2022. Virtual laser scanning with HELIOS++: A novel take on ray tracing-based simulation of topographic full-waveform 3D laser scanning. *Remote Sensing of Environment*, 269.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1912–1920.

Yan, W. Y., Shaker, A., El-Ashmawy, N., 2015. Urban land cover classification using airborne LiDAR data: A review. *Remote Sensing of Environment*, 158, 295–310.