AI-BASED 3D DETECTION OF PARKED VEHICLES ON A MOBILE MAPPING PLATFORM USING EDGE COMPUTING

J. Meyer [1, *], S. Blaser [1], S. Nebiker [1]

[1] Institute of Geomatics, FHNW University of Applied Sciences and Arts Northwestern Switzerland, Muttenz, Switzerland - (jonas.meyer, stefan.blaser, stephan.nebiker)@fhnw.ch

Commission I, ICWG I/IV

KEY WORDS: 3D Vehicle Detection, Deep Neural Networks, Edge Computing, Mobile Mapping, RGB-D, Robot Operating System, Point Clouds.

ABSTRACT:

In this paper we present an edge-based hardware and software framework for the 3D detection and mapping of parked vehicles on a mobile mapping platform for the use case of on-street parking statistics. First, we investigate different point cloud-based 3D object detection methods on our extremely dense and noisy depth maps obtained from low-cost RGB-D sensors to find a suitable object detector and determine the optimal preparation of our data. We then retrain the chosen object detector to detect all types of vehicles, rather than standard cars only. Finally, we design and develop a software framework integrating the newly trained object detector. By repeating the parking statistics of our previous work (Nebiker et al., 2021), our software is tested regarding the detection accuracy. With our edge-based framework, we achieve a precision and recall of 100% and 98% respectively on any parking configuration and vehicle type, outperforming all other known work on on-street parking statistics. Furthermore, our software is evaluated in terms of processing speed and volume of generated data. While the processing speed reaches only 1.9 frames per second due to limited computing resources, the amount of data generated is just 0.25 KB per frame.

1. INTRODUCTION

We are currently witnessing a transformation of urban mobility from motorized individual transport towards an increasing variety of multimodal mobility offerings, including public transport, dedicated bike paths, and various ridesharing services for cars, bikes, e-scooters, and the like. These offerings, on the one hand, are expected to decrease the need for on-street parking spaces and the undesirable traffic associated with searching for available parking spots, which has been shown to account for an average of 30% of the total traffic in major cities (Shoup, 2006). On the other hand, government agencies are interested in freeing street space – for example, that is currently occupied by on-street parking – to accommodate new lanes for bikes, public transport etc. to support and promote more sustainable traffic modes.
On-street parking statistics support government agencies and policy makers in reviewing and adjusting parking space availability, parking rules and pricing, and parking policies in general. However, creating parking statistics for city districts or even entire cities is a very labour-intensive process. For example, parking statistics for the city of Basel, Switzerland were obtained in 2016 and 2019 using low-cost GoPro videos captured from an e-bike in combination with manual interpretation by human operators (Rapp Trans AG Basel-Stadt, 2019).

Mathur et al. (2010), Bock et al. (2015) and Fetscher (2020) have successfully demonstrated the feasibility of reliable roadside parking statistics using observations from mobile mapping systems (MMS). However, the first two solutions are limited to parallel roadside parking, the second relies on an expensive MMS with two high-end LiDAR sensors, and the third utilizes 3D street-level imagery that does not provide the required revisit frequencies for time-of-the-day occupancy statistics.

To create reliable and cost-effective city-wide parking statistics with high revisit frequencies Nebiker et al. (2021) developed a mobile mapping platform equipped with consumer-grade sensors. Using AI-based 3D vehicle detection algorithms, the vehicles are detected in the collected RGB-D data and parking statistics are derived from it. However, this solution relies entirely on time-consuming and computationally intensive post-processing steps that require the acquisition, transfer, and anonymization of large amounts of data. In particular, the post-processing steps of data transfer and anonymization result in the evaluation taking about ten times the acquisition time. To reduce the need for data transfer and storage, to remove restrictions due to data privacy regulations, and to enable low latency, data is increasingly being processed close to the acquisition devices – at the edge. But combining edge computing with computationally intensive algorithms such as 3D object detection is still a significant challenge.

In this paper, we focus on reducing and further automating the post-processing steps of the evaluation workflow developed in Nebiker et al. (2021) by integrating the 3D vehicle detection into the system software using edge computing. Our main contributions are:

- Evaluation of a suitable 3D object detector and the optimal preparation of the acquired point cloud data.
- Training of the 3D object detector for any vehicle classes to increase reliability and accuracy of the parking statistics.
- Development of an edge-based software framework for data capturing and processing including 3D vehicle detection.

* Corresponding author

2. RELATED WORK

2.1 Detection of parked vehicles

Accurate detection of parked vehicles is crucial for both optimal utilization of the available parking space and for the creation of reliably parking statistics. Most work on parked vehicle detection relies on ground-based infrastructure. Thus, it is usually limited to indoor parking lots or off-street parking (Barriga et al., 2019; Paidi et al., 2018; Polycarpou et al., 2013). Several studies are aimed at supporting drivers in the actual search for free parking spaces (Houben et al., 2013; Suhr and Jung, 2018). However, these approaches are limited to the close vicinity of the vehicle and are not intended for global-scale mapping. While many studies are designed for smart parking, only a few approaches are aimed at or applicable to the acquisition of on-street parking statistics for city districts or entire cities. These works can be distinguished by the platform (ground or aerial vehicle), sensor technology (ultrasound, LiDAR, 2D and 3D imagery), detection algorithm and type (e.g., detection of gap or vehicle), and supported parking types.

Mathur et al. (2010) equipped probe vehicles with GPS and side-looking ultrasonic range finders mounted to the passenger door to determine parking spot occupancy. They achieved 95% accuracy in counting parking spaces and 90% accuracy in parking occupancy maps. However, this approach is only applicable to parallel parking lots and is unable to distinguish between vehicles and other objects with similar sensor responses. Bock et al. (2015) describe a method for extracting on-street parking statistics from 3D point clouds acquired with two 2D LiDAR sensors on a mobile mapping vehicle. Object segmentation followed by object classification with a random forest classifier is used to detect parked vehicles and results in 98.4% precision and 95.8% recall. The solution supports parallel and perpendicular parking, but its practical use is limited due to the expensive high-end dual LiDAR mobile mapping system. Recent studies investigated image-based methods for detecting parked vehicles or vacant parking spaces. For example, Grassi et al., (2017) describe an in-vehicle edge-based video analytics service for detecting vacant parking spaces in urban environments. Using dash-mounted smartphones, they achieve an average detection accuracy of close to 90% for parallel parking spaces only. On the basis of 3D street-level imagery (Nebiker et al., 2015) acquired by high-end mobile mapping system, Fetscher (2020) derives on-street parking statistics by detecting vehicles in 2D images and estimating the corresponding 3D bounding box (BB) in the masked point clouds. The author achieved a detection accuracy of 97% for parallel, perpendicular and angle parking types.

Due to the higher flexibility and the large field of view, aerial vehicles such as airplanes or drones equipped with image or LiDAR sensors are also used for the detection of parked vehicles (Peng et al., 2018; Sarkar et al, 2019; Yao et al., 2010). However, these approaches are not applicable in areas with trees or other objects that cover parking lots. In addition, high revisit frequencies are questionable because of the need for mission planning and potential restrictions.

To create reliable cost-effective city-wide parking statistics with high revisit frequencies Nebiker et al. (2021) developed a low-cost sensor setup on an electric tricycle as mobile mapping platform and the associated evaluation workflow. The sensor-setup consists of two active stereo RGB-D cameras, a GNSS and IMU-based navigation unit and an embedded single-board computer for data pre-processing and storage. Using the AI-based 3D object detector PointRCNN (Shi et al., 2019), parked vehicles are detected in the collected data and parking statistics are derived from the detection results. The solution achieves a recall and precision of 97% and 100% respectively for any parking type but cars only. Considering all types of vehicles in the test area Nebiker et al. (2021) achieved a recall of 87%. A further limitation is that the evaluation workflow is based entirely on time-consuming and computationally intensive post-processing steps that require the acquisition, transfer, and anonymization of large amounts of data. As a consequence, the evaluation takes about ten times the acquisition time

2.2 3D object detection

Object detection (OD) is generally defined as the fusion of object recognition and localization (Zhao et al., 2019). Since the localization of recognized objects within the 2D image plane is insufficient for many tasks such as path planning or collision avoidance in the field of autonomous driving, the third dimension is essential for the estimation of the exact, position, size and orientation of an object (Arnold et al., 2019). 3D information is incomplete and often noisy due to the sampling rate and accuracy of the sensors, which makes 3D OD a more challenging task than 2D OD. Arnold et al. (2019) divide 3D OD into three main categories based on the different sensor modalities: monocular, point cloud, and fusion. Point cloud methods are further divided into projection, volumetric and PointNet (Qi, et al., 2017) based on their main approach. Table 1 shows the different methods and briefly describes their methodology and limitations.

Modality		Methodology	Limitations
Monocular		Predict 2D BB in image plane and extrapolate them to 3D space via reprojection constraints or BB regression.	The lack of explicit depth information in input data leads to low accuracy in detection results.
Point cloud	projection	Project point clouds into 2D images, use established DNN architectures for 2D OD with extension to regress 3D BB.	Projection results in information loss and prevents explicit encoding of spatial information.
	volumetric	Generate 3D voxel representations from raw point clouds as input for Fully Convolutional Networks (FCN) to detect objects. The shape information is explicitly encoded.	Expensive 3D convolution increase inference time. Volumetric representation is sparse and computationally inefficient.
	PointNet	Use feed-forward networks consuming raw 3D point clouds to predict class and estimate BB.	Whole raw point cloud as input can massively increase run-time.
Fusion		Fuse image and point clouds for robust OD. Have typically multiple branches, one per modality, and rely on region proposals from 2D OD.	Require calibration between sensors and can be computationally expensive depending on the architecture.

Table 1. Comparison of 3D Object detection methods categorized by modality (Arnold et al., 2019).

Currently, point cloud and fusion based methods achieve the best results with about 83% detection accuracy (Geiger et al., 2022a). Monocular methods cannot compete with these results due to the lack of 3D information. For comparison, the best 2D object detection methods achieve detection results of better than 96% (Geiger et al., 2022b).

2.3 Edge computing

Edge Computing refers to a process where open platforms converge the core capabilities of networks, computing, storage and applications and provide intelligent services at the network edge – near the source of the objects or data – to meet the critical requirements of agile connection, real-time services, data optimisation, application intelligence, security and privacy protection of industry digitisation (Garcia Lopez et al., 2015). Sittón-Candanedo et al. (2019) and Yousefpour et al. (2019) provide comprehensive overviews of edge computing and their related computing paradigms. To provide computing resources at the network edge, often highly optimized devices with low power consumption such as the nVidia Jetson Series are utilized. Due to the low power consumption of the devices, their computing power is limited and the application of complex algorithms such as 3D vehicle detection is a major challenge. Therefore, several current works such as Demilew et al. (2020) or Aghdam et al. (2021) deal with fast 3D object detection on embedded systems.

3. MATERIALS AND METHODS

3.1 Electric mobile mapping system (eMMS)

The MMS and the evaluation workflow of our previous work (Nebiker et al., 2021) are the foundation for this work. The versatile eMMS consists of an electric tricycle as mapping platform and a low-cost sensor setup on the front luggage carrier (Figure 1). The sensor setup integrates two *Intel RealSense D455* RGB-D cameras (Intel Corporation, 2020), a GNSS and IMU-based navigation unit *SwiftNav Piksi Multi* (Swift Navigation Inc., 2019) and an embedded single-board computer *nVidia Jetson TX2* (nVidia Developers, 2022).

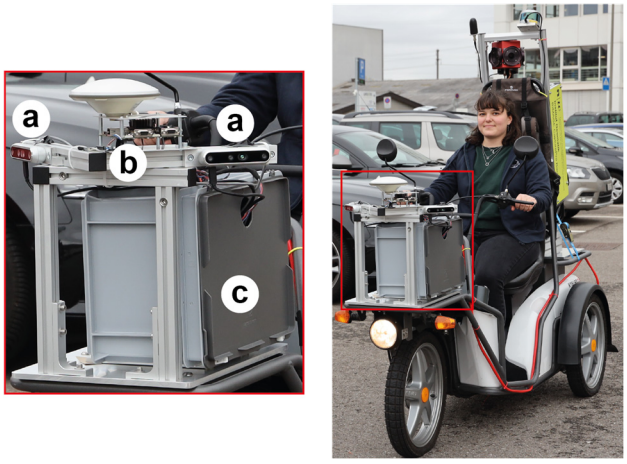


Figure 1. eMMS with low-cost sensor setup consisting of **a)** two RGB-D cameras, **b)** a GNSS and IMU-based navigation unit and **c)** a computer.

During the campaign, the computer triggers both RealSense cameras as well as the navigation unit with 5 frames per second (fps). The RealSense cameras, which are each mounted at a yaw angle of 45° to either side, capture the curb on both sides of the street. For post-processing the image poses in a geodetic reference frame (subsequently referred to as world coordinates), the navigation unit records the GNSS and IMU raw data and creates precise timestamps of the trigger events.
In the existing workflow, the evaluation of the acquired data comprises five different postprocessing steps. The acquired raw data are transferred and the RGB images are directly anonymized for privacy purposes. Using tightly coupled sensor data fusion of GNSS and IMU sensors, the trajectory travelled is calculated and the image poses are derived by interpolating the time stamps of the corresponding trigger events. Subsequently, the AI-based 3D object detector PointRCNN (Shi et al., 2019) is used to detect parked cars in the point clouds obtained from depth maps. The detection results are then transformed into world coordinates based on the image pose. Finally, the parking statistics are derived as part of a GIS analysis (Nebiker et al., 2021).

3.2 3D object detector and input data

Nebiker et al. (2021) showed that applying state of the art 3D OD methods – trained with the KITTI 3D object detection dataset (Geiger et al., 2012) – to the own RealSense point clouds generally resulted in significantly inferior detection results and inference times than reported in the original publications. The inferior detection results are due to the massively higher noise and the large data gaps, while the lower inference times are caused by the higher point density of the RealSense point clouds. Taking into account the mean acquisition distances of 30-40 m in the KITTI point clouds and of around 4 m in the RealSense point clouds in addition to the different sampling rates of the sensors (Intel Corporation, 2020; Velodyne Lidar Inc., 2014), the RealSense point clouds contain about 200-400 times more points than the KITTI point clouds.

In addition to a high detection accuracy, a low inference time is essential for the use of an 3D OD method at the edge, especially considering the limited computing resources of the nVidia Jetson TX2. Based on the findings of Nebiker et al. (2021), we assumed that reducing the number of points in the RealSense point clouds lowers the inference times of any 3D OD methods. At the same time, adapting the characteristics of the training data to the RealSense point clouds was expected to increase the detection accuracy. We conducted two independent experiments, one focusing on reducing the inference time and the other on increasing the detection accuracy. The combination of both results led to a suitable object detector and the optimal pre-processing of the input data.

Since many different state of the art 3D OD methods exist and the installation of such methods is often cumbersome due to ongoing developments and strict dependencies, we focused our investigations on the 3D OD methods provided in OpenPCDet (OpenPCDet Development Team, 2020) in order to limit the setup effort. OpenPCDet is a Python-based open-source library for point cloud-based 3D object detection. At the time of investigations OpenPCDet contained the following seven state of the art 3D OD methods:

- Part-A² Net anchor-free and -based (Shi et al., 2020b)
- PointPillars (Lang et al., 2019)
- PointRCNN (Shi et al., 2019)
- PV-RCNN (Shi et al., 2020a)
- SECOND (Yan et al., 2018)
- Voxel R-CNN (Deng et al., 2021)

3.2.1 Data preparation

To conduct the experiments different point clouds had to be prepared. This involved reducing the number of points and the noise by thinning and smoothing respectively. Since the individual discrete points within a point cloud have no topological information, such as belonging to a surface or neighbouring points, thinning and smoothing are computationally expensive. To avoid such expensive operations, we instead utilize the depth maps captured by the RealSense cameras (Figure 2, left). Smoothing filters such as Gaussian or Median can be applied directly to the depth maps represented as image data. To create a thinned point cloud (Figure 2, right), specific

points in the depth map are selected (Figure 2, middle) and converted to 3D points via known interior camera geometry and depth values.

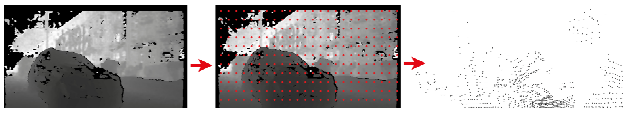
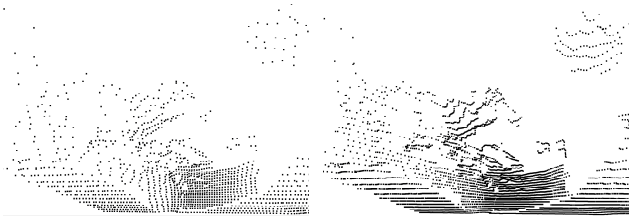


Figure 2. Preparation of a point cloud. Depth map is smoothed, if necessary, specific points are selected and transformed via interior camera geometry and depth values into 3D points.

For the point selection, we define a grid within the depth map (Figure 2, middle) with arbitrary horizontal and vertical step sizes. This approach enables the efficient thinning of point clouds while ensuring the preservation of geometry information without expensive scene interpretation. For the investigations on the inference time, point clouds with a reduced number of points were prepared. The horizontal and vertical step size of the grid were chosen evenly (Figure 3, a).



a) **b)**

Figure 3. Thinned point clouds. **a**) evenly distributed grid with step size 15 pixel. **b**) unevenly distributed grid with horizontal and vertical step size 3 and 15 pixel respectively.

To optimally adapt the characteristics of the KITTI point clouds to our own data, the noise was first reduced with a gaussian filter. However, pixels without values (NaN-values) led to a distortion of the smoothing, which is why a NaN-gaussian filter and a median filter were used for smoothing instead. Subsequently, the point clouds were thinned by defining a grid on the depth maps. According to the angular resolution of the sensor used to capture the KITTI dataset (Velodyne Lidar Inc., 2014), the vertical step size was five times the horizontal step size. This resulted in point clouds with a pattern similar to multi-profile laser scanners (Figure 3, b).

3.3 Training object detector

To create complete parking statistic, we had to overcome the limitation that tall vehicles cannot be detected. Therefore, we retrained the chosen 3D object detector with the KITTI 3D object detection dataset (Geiger et al., 2012). The comparison of the classes and the number of 3D objects included showed that only the class car has sufficient training data, while the other classes are severely underrepresented, except the class pedestrians. Considering the vehicles which could not be detected by Nebiker et al. (2021), the classes car, van, and truck were used to train the 3D object detector. Before training, 3D objects with less than five 3D points and detection difficulty classified as 'hard' were removed from the training set, resulting in the number of objects per class shown in Table 5.

Class	Car	Van	Truck
Num. 3D BB	10'759	835	316

Table 2. Classes and the number of contained 3D bounding boxes (3D BB) in KITTI 3D object detection dataset (Geiger et al., 2012) after filtering.

The trained model was evaluated on the evaluation set of the KITTI 3D object detection dataset and achieved the average precision (AP) values for the different difficulty levels as shown in Table 3. The intersection over union (IoU) threshold was set at 70% by default. While the AP values for the class car are as expected, the values for the classes truck and van indicate poor training results.

Class	Easy	Moderate	Hard
Car	89.01	78.59	77.82
Van	48.37	35.12	30.54
Truck	9.41	9.09	9.09

Table 3. Achieved AP_{70} values on the evaluation set of the KITTI 3D object detection dataset (Geiger et al., 2012) per class and detection difficulty (easy, moderate, hard).

3.4 Edge-based software framework

Processing the acquired data directly at the edge, on the on-board computer of the introduced eMMS, necessitated the extension of the system software developed by Nebiker et al. (2021). Since it is based on the graph-based robotic framework Robot Operating System (ROS) (Quigley et al., 2009) and has a modular and flexible design, we only had to add a new node for the 3D vehicle detection task (Figure 4, red ellipse). Thus, the developed edge-based software framework consists of three nodes (Figure 4). The trigger node creates electric impulses to perform hardware-based device triggering of the RGB-D cameras and navigation unit. Furthermore, the ROS wrapper for Intel RealSense devices (Dorodnicov and Hirshberg, 2021) is used for the RealSense camera control. By taking the trigger events and depth maps (Figure 4, pink squares) provided by the other nodes as input, the new node integrates 3D vehicle detection into the system software.

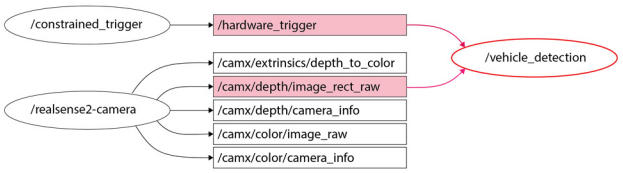


Figure 4. ROS computation graph of the developed edge-based software framework. Nodes are represented as ellipses and data topics as squares.

3.4.1 Vehicle detection node

Our developed vehicle detection node can be divided into three steps: data preparation, vehicle detection and data storage. In the data preparation step, as explained before, the depth map is converted into a thinned point cloud. Afterwards vehicles are detected in the point cloud using the functionality provided in OpenPCDet. Finally, the detection results are filtered by their detection probability score and stored to an external hard disk. As soon as a depth map is published, the three processing steps are performed in sequence. If the node is busy when the depth map arrives, it will be deleted without processing. This ensures that there are no delays or excessive queues if the trigger frequency is higher than the processing speed.

3.4.2 Evaluation workflow

By integrating 3D vehicle detection into the system software and directly storing the detection results, the need to transfer large amounts of data, anonymize RGB images and convert depth maps into point clouds is eliminated. Therefore, the number of remaining post-processing steps is reduced to three. First, the image poses are calculated, analogous to the original workflow.

Then the detection results are transformed into world coordinates via the image poses. Finally, a GIS Analysis is performed to derive the parking statistics.

4. EXPERIMENTS AND RESULTS

4.1 3D object detector and input data

The experiments for finding a suitable 3D object detector and the optimal data preparation, were conducted using data from a mobile mapping campaign with the introduced MMS. The test site is a representative European residential street in the city of Basel with parking spaces along both sides of the street. A total of 593 RGB-D images containing 34 parked cars were collected. The following experiments were conducted on a powerful workstation equipped with a graphics card.

4.1.1 Inference time

To investigate the influence of the number of points in the point clouds on the inference times of 3D OD methods, the test data were thinned by the factors 196 and 400. The factors correspond to the calculated difference in point density between the KITTI datasets and our own point clouds. The prepared and original data were processed with all seven 3D OD methods and the inference time was measured. Table 4 shows the achieved inference times in fps and the relative time reduction to the original data.

Method	Reduction point cloud	Inference. time [fps]	Red. inference time [%]
Part-A² Net anchor-based	-	7.6	
	196	11.4	33.4
	400	11.6	34.6
Part-A² Net anchor-free	-	3.6	
	196	13.0	72.4
	400	13.2	72.9
PointPillars	-	22.3	
	196	44.5	49.7
	400	45.1	50.5
PointRCNN	-	3.5	
	196	4.6	22.9
	400	4.9	28.0
PV-RCNN	-	1.7	
	196	10.9	84.1
	400	11.1	84.4
SECOND	-	15.5	
	196	28.9	46.2
	400	29.0	46.7
Voxel R-CNN	-	7.2	
	196	20.9	65.4
	400	22.3	67.6

Table 4. Inference time on original (-) and thinned point clouds (reduction factors 196 & 400).

The results clearly show that the thinning of point clouds leads to a significant reduction of the inference time for all 3D OD methods. However, depending on the method, the relative time gain varies between 28 and 84.4%. By reducing the number of points of our own data by a factor of 400, the published inference times for all methods except PointRCNN could be reproduced.

4.1.2 Detection accuracy

First, the test data were smoothed by nan-gaussian and median filters with different smoothing strengths. Subsequently, the smoothed and the original depth maps were thinned to different degrees by using the horizontal step sizes 6, 7, 8, 9 and 10. A total of 81 point cloud datasets were prepared with different combinations of smoothing filters and thinning factors. This data as well as the original data were processed with all seven 3D OD methods, the detection results were counted manually and the evaluation metrics recall, precision and F1 score were derived.

A first assessment of the results showed that the prior smoothing of the depth map does not lead to a significant improvement of the detection results. Furthermore, the smoothing took too much time considering the targeted processing speed of 5 fps. Therefore, the smoothing of the depth maps was not pursued further in this work. Table 5 shows the best detection results per 3D OD method obtained on thinned point clouds compared to the detection results on the original point clouds (no thinning). The performance of the detection results is assessed by the F1 score, as this measure considers both false positives and false negatives.

Method	P	R	F	Reduction point cloud
Part-A² Net anchor-based	1.00	0.97	**0.99**	180
	0.80	0.12	0.20	-
Part-A² Net anchor-free	1	0.97	**0.99**	320
	1.00	0.18	0.30	-
PointPillars	0.92	0.65	0.76	245
	0	0	undef.	-
PointRCNN	0.97	1	**0.99**	500
	0.92	0.97	**0.94**	-
PV-RCNN	1	0.94	0.97	320
	0.79	0.44	0.56	-
SECOND	0.88	0.85	0.87	245
	0.59	0.38	0.46	-
Voxel R-CNN	1	0.94	0.97	180
	1	0.21	0.34	-

Table 5. Best detection results on different **thinned point clouds** and **original point clouds** are displayed bold. P: precision; R: recall; F: F1 score.

The results in Table 5 prove that the thinning of the point clouds leads to a significant increase in detection accuracy for all 3D OD methods investigated. The 3D OD methods Part-A² Net anchor-based and free, and PointRCNN achieved the best detection results with an F1 score of 0.99. However, the optimal reduction factors of the point clouds differ greatly with 180, 320 and 500.

4.1.3 Suitable object detector and data preparation

Considering the obtained inference times (Table 4) and the detection accuracies (Table 5) of the investigated 3D OD methods, it appeared that the object detector Part-A² Net anchor-free is most suitable for the further course of this work. Further, it was shown that the optimal data preparation only involves thinning the point clouds by applying horizontal and vertical step sizes of 8 and 40, respectively.

4.2 Training object detector

By repeating the parking statistics conducted by Nebiker et al. (2021), we evaluated the newly trained object detector Part-A² Net anchor-free. This parking statistic includes 350 parking lots, of which 283 were occupied. The vehicle detections were manually counted and verified analogous to Nebiker et al. (2021). Table 6 presents the results broken down by type of parking space. The newly trained object detector achieved a precision of 100% and a recall of 98% across all parking and vehicle types, outperforming the pre-trained version by 11%. The detections of the van and truck classes were in the correct location, but the

estimated dimensions of the 3D bounding boxes were mostly wrong, which explains the poor AP values in Table 3.

	Parallel	Angle	Perpend.	2 x 2	Total
TP	181	26	65	4	276
TN	35	5	26	1	67
FP	0	0	0	0	0
FN	0	0	4	3	7
Precision	1.00	1.00	1.00	1.00	1.00
Recall	1.00	1.00	0.94	0.57	0.98

Table 6. Results of repeated parking statistics conducted by Nebiker et al. (2021) using the newly trained object detector Part-A² Net anchor-free. Results are broken down by the type of parking space (parallel, angle, perpendicular and 2 x 2).

4.3 Edge-based software framework

To determine the effectiveness of our edge-based software framework, the criteria of latency, detection accuracy, memory requirements and data volume are particularly relevant. The detection accuracy of the object detector used has already been demonstrated in the previous sections. To determine the latency and data volume criteria, a campaign saved as a ROS-bag file was replayed and processed by our edge-based software framework with the determined point cloud preparation settings.

4.3.1 Latency

The system software developed by Nebiker et al. (2021) worked at 5 fps and was tuned to the targeted distance between two consecutive images of 1-2 metres and the average acquisition speed of 25-30 km/h. Therefore, we aimed to achieve the same processing rate with our edge-based software framework. Since the processing rate depends directly on the available computing resources and the tasks to be performed, we could not estimate in advance whether this goal would be achieved. The experiment showed that we achieved a processing rate of 1.9 fps with our edge-based software framework on the nVidia Jetson TX2.

4.3.2 Memory requirements

Since the available memory is usually critical in embedded systems like the utilised nVidia Jetson TX2 it is good to know how much memory is required by each software module. Our developed ROS-based software requires memory of about. 0.9 GB. Including all the necessary Python modules and the trained model of the object detector, the required memory space amounts to about 2.5 GB.

4.3.3 Data volume

Furthermore, the amount of data generated by the software is of great interest, as this data must be transferred. While the old system software generated around 15 MB of data per frame, the amount of data generated by the new edge-based software framework is just 0.25 KB. With a capture rate of 5 fps, 1.25 KB of data per second is now generated instead of the previous 75 MB/s, which represents a reduction by a factor of 60’000.

5. DISCUSSION

5.1 3D object detector and optimal input data

The investigations showed that the input data for 3D OD methods have a significant influence. By reducing the RealSense point clouds by a factor of 196 and 400, relative time reductions of 22.9-84.1% and 28.0-84.4% could be achieved, depending on the 3DOD method (Table 4). However, the influence of the number of points on the 3D OD methods varies greatly. For the object detector PV-RCNN, for example, a reduction in inference time of 84% was achieved, whereas with PointRCNN the reduction was only 28%. In order to determine the exact reasons for the differences in acceleration due to reduced point clouds, the implementations of the different 3D OD methods would need to be investigated in detail. A possible reason could be the utilization of pre-processing steps like voxelization or projections into images which are performed at the beginning of an 3D OD method to limit the number of operations.

Furthermore, the investigations showed that adapting the characteristics of the training data to our own data leads to significantly better detection results. Five of seven 3D OD methods achieved an F1 score greater than 95% on the thinned point clouds, while only one of seven methods achieved an F1 score greater than 95% on the original point clouds (Table 5).

We also found that smoothing the depth maps beforehand did not lead to the expected additional improvement in detection accuracy, but rather worsened it. One reason for this could be the weakening of edges which results from smoothing in addition to the reduction of noise. This would further weaken the weak edges in our own point clouds and thus correspond even less to the characteristics of the KITTI point clouds, which have very exact and sharp edges.

5.2 Training object detector

To create valid parking statistics, the reliable and accurate three-dimensional detection of all vehicle types is of crucial importance. By retraining the Part-A² Net anchor-free object detector, all vehicles could be detected and a precision and recall of 100% and 98% respectively was achieved. However, the estimation of the bounding box dimensions for the classes truck and van was very poor. The AP values (Table 3) of the evaluation of the trained model on the evaluation set of the KITTI 3D object detection dataset match this finding. Since the vehicles were correctly recognised, the estimated dimensions did not meet the IoU criterion of 70%, resulting in the poor AP-values (Table 3).

The main reason for this is certainly the small number of training data of 835 and 316 objects respectively (Table 2). Moreover, the classes van and truck are very heterogeneous regarding the object dimensions. Family vans and large delivery vans differ in length by up to 2 metres and in height by up to 1.5 metres. Nevertheless, both vehicle categories are in the same class. It is therefore not surprising that the object detector could not reliably learn the estimation of the correct object dimensions on the limited amount of training data available.

5.3 Edge-based software framework

Due to the development of the edge-based software framework and the integration of 3D vehicle detection, the storage and transfer of large amounts of data as well as the time-consuming post-processing steps, namely the anonymization of RGB images, the processing of point clouds from depth maps and 3D vehicle detection, could be eliminated. By using OpenPCDet as the framework for 3D object detection within the software we have developed, all its 3D OD methods are available. This offers great flexibility, for example, for using newly trained models or for easily testing and applying new 3D OD methods.

The performance evaluation of our software shows great potential for improvement in latency, which averages around 0.53 seconds or 1.9 fps. The latency is strongly dependent on the available computing resources of the nVidia Jetson TX2 and should therefore be significantly reduced with a more powerful

embedded hardware module. Furthermore, it would be possible to accelerate the inference of the object detector by model compression and acceleration techniques for example demonstrated in Nousias et al. (2021).

Our edge-based software framework requires a total of 2.5 GB of memory. As there are no other applications running on the on-board computer, the 2.5 GB memory requirement is not a problem. If memory becomes scarce, the pre-trained models can be swapped out to an external hard drive. In addition to the memory requirement, the amount of data generated by the system software was determined. With our new software, the amount of generated data could be reduced from 15 MB per frame to 0.25 KB. This reduction of generated data by a factor of 60,000 is one of the main advantages of edge computing and clearly demonstrates the effectiveness of our developed edge-based software framework for 3D vehicle detection.

5.4 Overall Capabilities and Performance

With our mobile mapping system including the newly developed edge-based software framework and the associated evaluation workflow, we outperform all the other work known to us for the creation of on-street parking statistics (Bock et al., 2015; Fetscher, 2020; Grassi et al., 2017; Mathur et al., 2010; Nebiker et al., 2021) regarding the detection accuracy of parked vehicles, but also with respect to the applicability to different parking types and a high revisit frequency. However, for the efficient creation of on-street parking statistics our edge-based software framework has to be accelerated to reach the minimal required capturing frequency of 5 fps.

6. CONCLUSION AND OUTLOOK

In this work, we further automated the creation of parking statistics by using edge computing for on-board 3D object detection on the low-cost mobile mapping system developed by Nebiker et al. (2021). Our new edge-based software framework integrates the 3D vehicle detection directly into the acquisition phase, which eliminates time-consuming and costly post-processing steps – such as the transfer of large amounts of data – and the anonymisation of RGB images. For the development of the edge-based software framework, we first evaluated a suitable 3D object detection method as well as the optimal processing of the point clouds from RealSense D455 RGB-D cameras in terms of low inference times and high detection accuracy. We then trained the selected object detector to detect all types of vehicles and developed the edge-based software framework based on the existing system software.

The newly trained object detector achieved a precision of 100% and a recall of 98% for all vehicles and all known parking types by repeating the parking statistics of Nebiker et al. (2021). However, the estimation of vehicle dimensions is still insufficient. By integrating the 3D vehicle detection, our developed software generates 60’000 times less data than previously. However, the computationally intensive algorithms of the 3D vehicle detections led to severely limited processing rate of 1.9 fps on the available hardware.

In future work we will accelerate the processing speed by acquiring a new on-bord computer with competitive performance. Furthermore, we will improve the detection of vehicles, especially the estimation of BB dimensions, by collecting additional training data directly with our vehicle. Alternatively, other public training data sets can be used. However, the data preparation determined would most likely have to be adjusted. Finally, a robust real-time sensor orientation ensuring sub-meter position in challenging urban environments would eliminate post-processing altogether and allow for a variety of additional use cases.

ACKNOWLEDGEMENTS

This research was initiated and partly funded by the “Amt für Mobilität, Bau- und Verkehrsdepartement des Kantons Basel-Stadt” as part of the pilot study “Bildbasiertes Parkplatz-monitoring”.

REFERENCES

Aghdam, H. H., Heravi, E. J., Demilew, S. S., Laganiere, R., 2021. RAD: Realtime and Accurate 3D Object Detection on Embedded Systems. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2869-2877. doi.org/10.1109/CVPRW53098.2021.00322.

Arnold, E., Al-Jarrah, O. Y., Dianati, M., Fallah, S., Oxtoby, D., Mouzakitis, A., 2019. A Survey on 3D Object Detection Methods for Autonomous Driving Applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10), 3782-3795. doi.org/10.1109/TITS.2019.2892405.

Barriga, J. J., Sulca, J., León, J. L., Ulloa, A., Portero, D., Andrade, R., Yoo, S. G., 2019. Smart Parking: A Literature Review from the Technological Perspective. *Applied Sciences*, 9(21), 4569. doi.org/10.3390/app9214569.

Bock, F., Eggert, D., Sester, M., 2015. On-street Parking Statistics Using LiDAR Mobile Mapping. *Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Gran Canaria, Spain, 2812-2818. doi.org/10.1109/ITSC.2015.452.

Demilew, S. S., Aghdam, H. H., Laganière, R., Petriu, E. M., 2020. FA3D: Fast and Accurate 3D Object Detection. In G. Bebis, Z. Yin, E. Kim, J. Bender, K. Subr, B. C. Kwon, J. Zhao, D. Kalkofen, G. Baciu (Eds.), *Advances in Visual Computing*, 397-409. Springer International Publishing, Cham.

Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., Li, H., 2021. Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(2), 1201-1209.

Dorodnicov, S., Hirshberg, D., 2021. realsense2_camera. http://wiki.ros.org/realsense2_camera (14 March 2022).

Fetscher, S., 2020. Automatische Analyse von Streetlevel-Bilddaten für das digitale Parkplatzmanagement. Bachelor's Thesis, FHNW University of Applied Sciences and Arts Northwestern Switzerland, Muttenz, Switzerland (unpublished).

Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., Riviere, E., 2015. Edge-centric Computing: Vision and Challenges. *ACM SIGCOMM Computer Communication Review*, 45(5), 37-42. doi.org/10.1145/2831347.2831354.

Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2022a. The KITTI Vision Benchmark - 3D Object Detection Evaluation 2017. http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d (9 March 2022).

Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2022b. The KITTI Vision Benchmark - Object Detection Evaluation 2012. http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=2d (10 March 2022).

Geiger, A., Lenz, P., Urtasun, R., 2012. Are we ready for autonomous driving? The KITTI vision benchmark suite. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, 3354-3361. doi.org/10.1109/CVPR.2012.6248074.

Grassi, G., Jamieson, K., Bahl, P., Pau, G., 2017. Parkmaster: An in–vehicle, edge–based video analytics service for detecting open parking spaces in urban environments. *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, San Jose, CA, USA, 1-14. doi.org/10.1145/3132211.3134452.

Houben, S., Komar, M., Hohm, A., Luke, S., Neuhausen, M., Schlipsing, M., 2013. On-vehicle video-based parking lot recognition with fisheye optics. *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, The Hague, The Netherlands, 7-12. doi.org/10.1109/ITSC.2013.6728595.

Intel Corporation, 2020. Intel®RealSense - Product Family D400 Series: Datasheet. https://www.intelrealsense.com/wp-content/uploads/2020/06/Intel-RealSense-D400-Series-Datasheet-June-2020.pdf (10 March 2022).

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O., 2019. Pointpillars: Fast encoders for object detection from point clouds. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 12689-12697. doi.org/10.1109/CVPR.2019.01298.

Mathur, S., Jin, T., Kasturirangan, N., Chandrasekaran, J., Xue, W., Gruteser, M., Trappe, W., 2010. ParkNet. *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services - MobiSys '10*, San Francisco, CA, USA, 123. doi.org/10.1145/1814433.1814448.

Nebiker, S., Cavegn, S., Loesch, B., 2015. Cloud-Based Geospatial 3D Image Spaces—A Powerful Urban Model for the Smart City. *ISPRS International Journal of Geo-Information*, 4(4), 2267-2291. doi.org/10.3390/ijgi4042267.

Nebiker, S., Meyer, J., Blaser, S., Ammann, M., Rhyner, S., 2021. Outdoor Mobile Mapping and AI-Based 3D Object Detection with Low-Cost RGB-D Cameras: The Use Case of On-Street Parking Statistics. *Remote Sensing*, 13(16), 3099. doi.org/10.3390/rs13163099.

Nousias, S., Pikoulis, E.-V., Mavrokefalidis, C., Lalos, A. S., Moustakas, K., 2021. Accelerating 3D scene analysis for autonomous driving on embedded AI computing platforms. *International Conference on Very Large Scale Integration (VLSI-SoC)*, Singapore, 1-6. doi.org/10.1109/VLSI-SoC53125.2021.9606990.

nVidia Developers, 2022. Jetson TX2 Module. https://developer.nvidia.com/embedded/jetson-tx2 (9 March 2022).

OpenPCDet Development Team, 2020. OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds. https://github.com/open-mmlab/OpenPCDet (8 March 2022).

Paidi, V., Fleyeh, H., Håkansson, J., Nyberg, R. G., 2018. Smart parking sensors, technologies and applications for open parking lots: a review. *IET Intelligent Transport Systems*, 12(8), 735-741. doi.org/10.1049/iet-its.2017.0406.

Peng, C., Hsieh, J., Leu, S., Chuang, C., 2018. Drone-Based Vacant Parking Space Detection. *International Conference on Advanced Information Networking Applications Workshops (WAINA)*, Krakow, Poland, 618-622. doi.org/10.1109/WAINA.2018.00155.

Polycarpou, E., Lambrinos, L., Protopapadakis, E., 2013. Smart parking solutions for urban areas. *2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Madrid, Spain, 1-6. doi.org/10.1109/WoWMoM.2013.6583499.

Qi, C. R., Su, H., Mo, K., Guibas, L. J., 2017. PointNet: Deep learning on point sets for 3D classification and segmentation. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Honolulu, USA, 77-85. doi.org/10.1109/CVPR.2017.16.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A., 2009. ROS: an open-source Robot Operating System. *ICRA*, 3(3.2) 5.

Rapp Trans AG Basel-Stadt, 2019. Erhebung Parkplatzauslastung Stadt Basel 2019. Basel, Switzerland.

Sarkar, S., Totaro, M. W., Elgazzar, K., 2019. Intelligent drone-based surveillance: application to parking lot monitoring and detection. In C. M. Shoemaker, H. G. Nguyen, P. L. Muench (Eds.), *Unmanned Systems Technology XXI*, 11021, 13–19. doi.org/10.1117/12.2518320.

Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H., 2020a. PV-RCNN: Point-voxel feature set abstraction for 3D object detection. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, 10526-10535. doi.org/10.1109/CVPR42600.2020.01054.

Shi, S., Wang, X., Li, H., 2019. PointRCNN: 3D object proposal generation and detection from point cloud. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 770-779. doi.org/10.1109/CVPR.2019.00086.

Shi, S., Wang, Z., Shi, J., Wang, X., Li, H., 2020b. From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1. doi.org/10.1109/tpami.2020.2977026.

Shoup, D. C., 2006 Cruising for parking. *Transport Policy*, 13(6), 479-486. doi.org/10.1016/j.tranpol.2006.05.005.

Sittón-Candanedo, I., Alonso, R. S., Corchado, J. M., Rodríguez-González, S., Casado-Vara, R., 2019. A review of edge computing reference architectures and a new global edge proposal. *Future Generation Computer Systems*, 99(October), 278-294. doi.org/10.1016/j.future.2019.04.016.

Suhr, J., Jung, H., 2018. A Universal Vacant Parking Slot Recognition System Using Sensors Mounted on Off-the-Shelf Vehicles. *Sensors*, 18(4), 1213. doi.org/10.3390/s18041213.

Swift Navigation Inc., 2019. PiksiMulti GNSS Module Hardware Specification. https://www.swiftnav.com/latest/piksi-multi-hw-specification (08 March 2022).

Velodyne Lidar Inc., 2014. High Definition Lidar HDL-64E - Datasheet.

Yan, Y., Mao, Y., Li, B., 2018. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10), 3337. doi.org/10.3390/s18103337.

Yao, W., Hinz, S., Stilla, U., 2010. Automatic vehicle extraction from airborne LiDAR data of urban areas aided by geodesic morphology. *Pattern Recognition Letters*, 31(10), 1100-1108. doi.org/10.1016/j.patrec.2010.02.006.

Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., Jue, J. P., 2019. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98(September 2019), 289-330. doi.org/10.1016/j.sysarc.2019.02.009.

Zhao, Z. Q., Zheng, P., Xu, S. T.,Wu, X., 2019. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning System*s, 30(11), 3212-3232.