SAE INTERNATIONAL

ARTICLE INFO
Article ID: 12-08-04-0032

doi:10.4271/12-08-04-0032

Data-Driven Performance Evaluation of Machine Learning for Velocity Estimation Based on Scan Artifacts from LiDAR Sensors

Lukas Haas,[1,3] Arsalan Haider,[1,3], Ludwig Kastner,[1] Matthias Kuba,[2] Thomas Zeh,[1] Martin Jakobi,[3] and Alexander Walter Koch[3]

[1]Kempten University of Applied Sciences, Institute for Driver Assistance Systems and Connected Mobility, Germany
[2]Kempten University of Applied Sciences, Faculty of Electrical Engineering, Germany
[3]Technical University of Munich, Institute for Measurement Systems and Sensor Technology, Germany

Abstract

Light detection and ranging (LiDAR) sensors are increasingly applied to automated driving vehicles. Microelectromechanical systems are an established technology for making LiDAR sensors cost-effective and mechanically robust for automotive applications. These sensors scan their environment using a pulsed laser to record a point cloud. The scanning process leads in the point cloud to a distortion of objects with a relative velocity to the sensor. The consecutive generation and processing of points offers the opportunity to enrich the measured object data from the LiDAR sensors with velocity information by extracting information with the help of machine learning, without the need for object tracking. Turning it into a so-called 4D-LiDAR. This allows object detection, object tracking, and sensor data fusion based on LiDAR sensor data to be optimized. Moreover, this affects all overlying levels of autonomous driving functions or advanced driver assistance systems. However, since such sensor-specific effects are rarely available in public datasets and the velocities of target objects are not included as ground truth in these datasets, it makes sense to enrich the limited real-world data with synthetic data. Therefore, this article discusses how such datasets can be created and combined to efficiently estimate velocities on real-world data using the novel method named VeloPoints.

History
Received: 06 Feb 2024
Revised: 20 Sep 2024
Accepted: 24 Dec 2024
e-Available: 21 Jan 2025

Keywords
LiDAR sensor, Deep learning, Motion distortion effect, Point cloud, Simulation, Advanced driver assistance systems, Highly automated driving, Velocity estimation

Citation
Haas, L., Haider, A., Kastner, L., Kuba, M. et al., "Data-Driven Performance Evaluation of Machine Learning for Velocity Estimation Based on Scan Artifacts from LiDAR Sensors," *SAE Int. J. of CAV* 8(4):493–503, 2025, doi:10.4271/12-08-04-0032.

ISSN: 2574-0741
e-ISSN: 2574-075X

Introduction

Currently, the data of light detecting and ranging (LiDAR) sensors are used in many autonomous systems, such as autonomous driving (AD) systems and advanced driver assistance systems (ADAS). The sensor data is used not only for object detection and object tracking but also for determining the vehicle's velocity. The information is then made available to the AD and ADAS functions. Until now, object tracking has been used to determine the velocity of objects. However, this is associated with increased computing effort and has the disadvantage that no velocity information is initially available for any newly detected object [1, 2]. In [3], the authors introduced VeloPoints, a method based on the motion distortion effect, with which the velocity of an object can be determined with the help of machine learning based on a single LiDAR frame without object tracking. The motion distortion effect occurring in scanning time of flight (ToF) LiDAR sensors has often been considered a problem so far [4]. However, this effect also offers the opportunity to enrich the measured sensor data with velocity information through information extraction and to turn it into a so-called 4D LiDAR without object tracking. For different driving functions the direction of movement and velocity are essential [5]. Furthermore, the resulting velocity information can be used prevent the false positives based on the motion distortion effect in object detection described in [6] based on the LiDAR data and improve object tracking as described in [7]. In addition, the velocity information can be provided to a sensor data fusion more quickly to make it more robust.

However, such sensor-specific effects are rarely available in publicly released datasets due to different LiDAR sensor types. In addition, the velocities of target objects are rarely included as ground truth in the datasets. For this reason, enriching the small amount of real-world data with synthetic data seems efficient. Therefore, in this article, we investigate the effect of simulated data, real-world data, and a combination of the two types of data on velocity estimation based on scan artifacts from scanning ToF LiDAR sensors.

This article is structured as follows. First, the theoretical background of LiDAR sensors, the motion distortion effect for the sensor used in this work, and LiDAR-based velocity estimation are generally discussed, and the state of the art is given. Then, the generation of the simulated and real-world datasets is explained in detail. After this, the effect of simulated, real, and combined datasets, as well as the point intensity, on the estimation of the VeloPoints network is described. Subsequently, the results are summarized, and an outlook on future applications and research is given.

Theoretical Background

LiDAR Sensors

LiDAR sensors can be divided into three categories based on the measurement principle, amplitude-modulated continuous wave (AMCW), frequency-modulated continuous wave, and ToF LiDAR sensors [8]. FMCW LiDAR sensors emit a wave with a modulated frequency f_0, typically with a wavelength of around 1550 nm. The emitted wave is reflected by an object in the distance d with a frequency of f_r. After the propagation time Δt, the reflected wave reaches the sensor. In the sensor the transmit and received signal get mixed coherently in a heterodyne detector. This results in the intermediate frequency f_{IF} (Figure 1) [9].

The following applies:

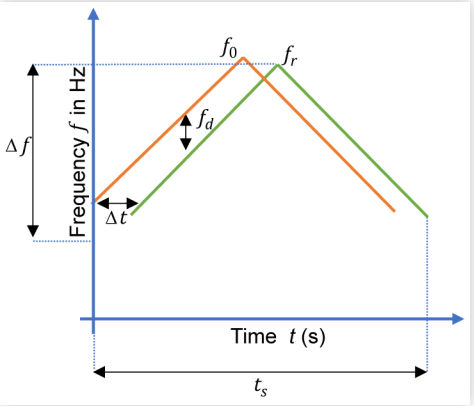
$$d \sim \Delta t \sim f_{IF} \qquad \text{Eq. (1)}$$

The reflected frequency is measured in the sensor using a frequency detector, and the intermediate frequency f_{IF} calculated. The distance of the reflection is calculated using the ramp duration t_s and the bandwidth of the frequency ramp Δf [9, 12]:

$$d = \frac{c \cdot f_{IF}}{2 \cdot \frac{\Delta f}{t_s}} \qquad \text{Eq. (2)}$$

The position of each point in (x_s, y_s, z_s) in the Cartesian sensor coordinate system is calculated based on the spherical coordinates consisting of the azimuth and elevation angles and the distance d calculated using Equation 2.

FIGURE 1 FMCW principal in LiDAR sensors with triangle modulation cf.



Data taken from Refs. [9, 10, 11]. © Lukas Haas, Arsalan Haider, Ludwig Kastner, Matthias Kuba, Thomas Zeh, Martin Jakobi, Alexander W. Koch

ToF LiDAR sensors emit laser pulses in a wavelength range, typically ~905 nm or ~1550 nm, into their surroundings, unlike FMCW LiDAR sensors. The emitted laser pulses are partly reflected by the object's surfaces in the sensor's field of view (FoV) and detected by the LiDAR sensor. Then, the LiDAR sensor measures the propagation time τ between the emission and detection of a laser pulse to calculate the distance d via the speed of light c, as given in Equation 3. The working principle of a ToF LiDAR sensor is shown in Figure 2.

$$d = \frac{c \cdot \tau}{2} \qquad \text{Eq. (3)}$$

The position of the reflection in the Cartesian sensor coordinate system (x_s, y_s, z_s) is calculated based on the known azimuth and elevation angle and the distance at which the laser pulse was reflected from the object.

ToF LiDAR sensors can be categorized into scanning and flash LiDAR sensors. Scanning LiDAR sensors emit multiple laser pulses in different defined directions in their FoV to capture 3D points in the sensor's surroundings within one frame. These scanning LiDAR sensors typically can measure distances of up to 200 m with a horizontal and vertical resolution of 0.1° [13, 14]. Flash LiDAR sensors represent the second category. In contrast to scanning LiDAR sensors, these sensors illuminate their entire FoV with one laser pulse. However, they typically can only measure distances of up to 50 m due to the limited laser emission power resulting from eye safety regulations [15, 16, 17]. Due to the more considerable maximum measurement distance, the data from scanning LiDAR sensors are used as the basis for various ADAS and AD functions, such as lane departure warnings (LDW), simultaneous localization and mapping (SLAM), forward collision warning (FCW), blind spot detection (BSD), and adaptive cruise control (ACC) [17].

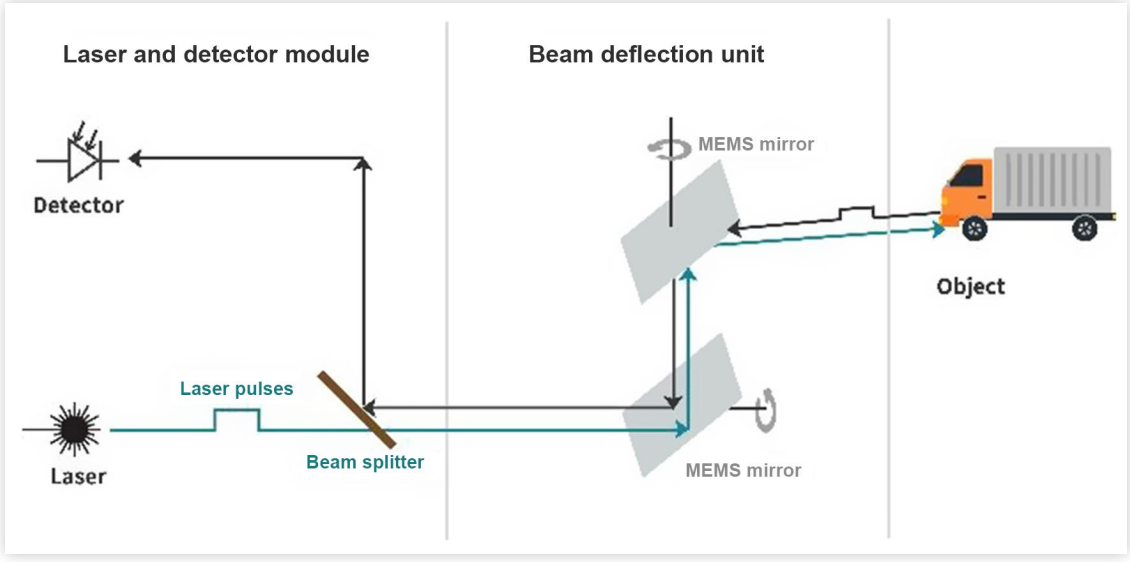
Microelectromechanical system (MEMS)-based LiDAR sensors offer a promising concept for making scanning LiDAR sensors cost-effective and mechanically robust for automotive use.

Motion Distortion Effect

The scanning process of scanning LiDAR sensors results in the motion distortion effect. The motion distortion effect distorts objects in the point cloud due to the relative movement of the objects during the scanning process. This correlation between the sensor's frame rate and the moving object's velocity and trajectory can also be found in [18, 19]. The effect of motion distortion is particularly noticeable with MEMS-based ToF LiDAR sensors [20]. In the case of the Cube1 LiDAR sensor from the Blickfeld GmbH used in this work, the FoV is scanned twice during the scanning process for a higher resolution, once in the so-called up-ramp and once in the down-ramp [21]. This leads to multiple detections of a moving object of interest (OoI) in the middle of the FoV rather than a continuous distortion of the OoI. The sensor for this work allows different scan patterns for a maximum FoV of ±36° horizontal and ±15° vertical, with a vertical resolution between 6° and 0.075° per frame and a horizontal resolution of 0.4°–1.0°. The frame rate of the sensor varies with the resolution [6, 22]. The time duration between each scan point is t_p. At a frame scan frequency of 5.4 Hz, t_p = 10.2 µs [4]. The distance that the object has traveled between n points can be calculated by

$$d = n \cdot t_p \cdot v \qquad \text{Eq. (4)}$$

FIGURE 2 Working principle of a scanning ToF LiDAR sensor using two 1D microelectromechanical system-based mirrors.



Equation 4 applies: The higher the relative velocity v between the object and the lower the frame rate (higher t_p), the greater the distortion of the object and, therefore, the greater the motion distortion effect. Figure 3 visualizes two measured point cloud sections, with and without the motion distortion effect. On the top, a point cloud section of a vehicle driving in the x-direction in the sensor coordinate system is shown, once with a relative velocity of 0.0 m/s (without the motion distortion effect) (a) and once with a relative velocity of 18.61 m/s (with the motion distortion effect) (b). The bottom of Figure 3 shows a point cloud section of a vehicle driving in the y-direction in the sensor coordinate system, once with a relative velocity of 0.0 m/s (without the motion distortion effect) (c) and once with a relative velocity of 16.94 m/s (with the motion distortion effect) (d).

This distortion can lead to possible false positives in object recognition making correct classification in object detection more difficult. Thereby, different methods to correct the motion distortion in a point cloud caused by a moving ego vehicle have been presented in several papers, e.g., by Renzler et al. [23] and Muro et al. [24].

Velocity Measurement

The velocity information of objects in the environment of a vehicle is a crucial information basis for many decisions regarding the functions of AD and ADAS, such as in ACC [5].

FMCW LiDAR sensors can measure the velocity of an object based on the Doppler effect. The object's velocity in the radial direction can be calculated using Equation 6. The coherent wave emitted by the sensor is reflected by the object to be detected. Due to the Doppler effect, there is a frequency shift Δf_D of the wave when reflected at an object with a radial velocity v_r [10, 11]:

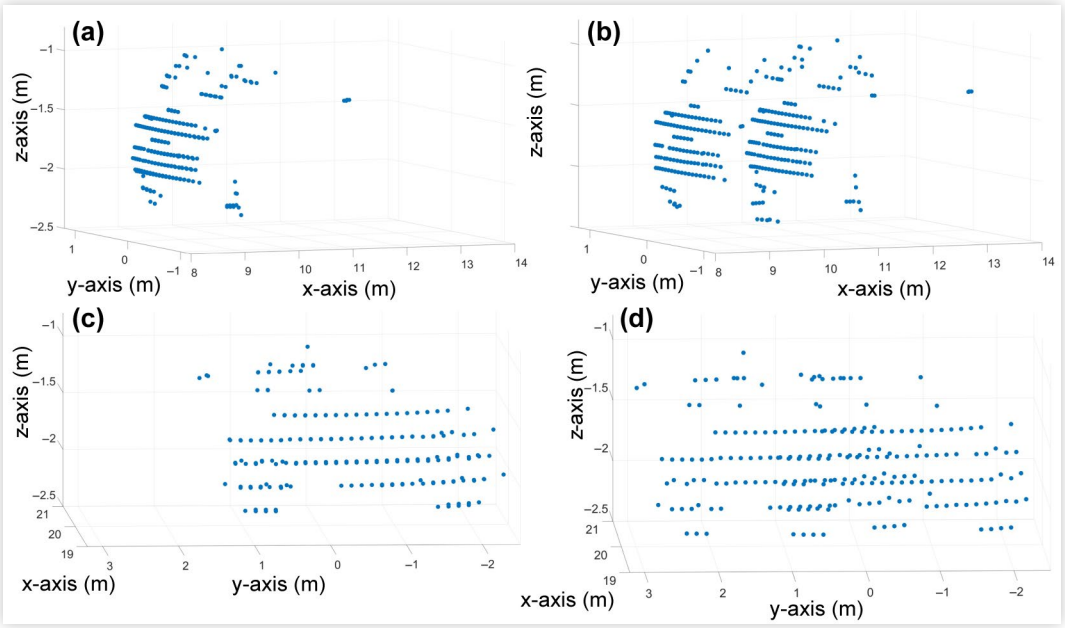
$$\Delta f_D = \frac{2 \cdot v_r}{\lambda_{LiDAR}} \qquad \text{Eq. (5)}$$

As the OoI approaches the sensor, the mixed frequency f_d is increased by Δf_D during downward chirping and decreased by Δf_D during upward chirping. By calculating the difference and the sum of these two frequencies, both the radial velocity v_r and the distance d of the OoI can be calculated using Equations 6 and 7 despite a frequency shift due to the Doppler effect [10, 11]:

$$v_r = \frac{c \cdot 2 \cdot \Delta f_D}{4 \cdot f_0} \qquad \text{Eq. (6)}$$

$$d = \frac{\left|\Delta f_D - f_{IF}\right| \cdot t_s \cdot c}{4 \cdot \Delta f} \qquad \text{Eq. (7)}$$

FIGURE 3 Synthetic point clouds of a vehicle. The top left shows a point cloud section of a vehicle with driving orientation in the x-direction with a relative velocity of 0.0 m/s (without a motion distortion effect) (a). The top right shows a point cloud section of a vehicle moving in the x-direction with a relative velocity of 18.61 m/s (b). The bottom left shows a point cloud section of a vehicle with driving orientation in the y-direction with a relative velocity of 0.0 m/s (without a motion distortion effect) (c). The bottom right shows a point cloud section of a vehicle driving in the y-direction with a relative velocity of 16.94 m/s (d).



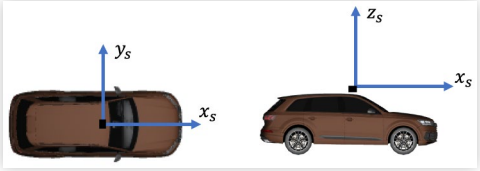
To determine the velocity of the OoI longitudinally and laterally to the sensor, FMCW LiDAR sensors must track the detected objects over several frames. Siamese tracking is often used for single-object tracking. Here, the OoI is searched for in sequential frames in the surroundings of the position in which the object has been detected in the previous frames [25]. For LiDAR sensors, the object's shape in the point cloud is considered in this approach, as in [26, 27]. With multi-object tracking, several different objects can be tracked. These methods track the bounding boxes based on the shape and movement of the objects over several frames, as in [28, 29]. As a result, the velocity can be determined by deriving the position of the object over time. In [7], Gu et al. show that the measured radial velocity can enhance the tracking method.

Both object tracking methods and the VeloPoints method, which is based on the motion distortion effect, can be used to estimate an object's longitudinal and lateral velocity using scanning ToF LiDAR sensors. As with the FMCW LiDAR sensor, the single-object and multi-object tracking methods can be used for object tracking. Furthermore, as described by the authors in [3], the velocity can also be estimated using machine learning, specifically a multi-regression convolutional artificial neural network, based on the motion distortion effect. For this purpose, the 3D point cloud of the sensor, which is distorted by the relative velocity of the object, is converted into a 2D pseudo-image, as also described in [30], and used as input data for an artificial neural network. This network consists of a multi-regression convolutional neural network with two convolutional layers, two pooling layers, and six hidden layers. This outputs the velocities for the object in both the longitudinal and lateral directions to the sensor. This means the object does not have to be tracked over several frames, and the velocity information is already available in the first frame in which the object is detected. Depending on the application, this velocity estimation can then be used as the final velocity estimation, or optionally, as with the Doppler effect-based velocity information of an FMCW LiDAR, it can be transferred as an input parameter to a tracking method. The authors of [3] demonstrate the efficient development of the neural network and the performance of the methodology. However, the influence of real data on the existing network still needs to be addressed. For this reason, in this work, we evaluate and optimize the neural network, called VeloPoints, with real and synthetic datasets and analyze their influence on the network performance.

Simulation Dataset

A labeled dataset is required to train and validate artificial neural networks using supervised learning. Simulation is often used to develop neural networks and corresponding datasets efficiently. Depending on the required level of detail and scenario, datasets can often be created more efficiently in simulation than in reality. To generate a simulated dataset in this work, the simulation software CarMaker from IPG Automotive GmbH [31] was combined with a high-fidelity sensor model of the Cube1 scanning LiDAR sensor presented in [4] that reproduces the sensor-specific imperfections. As described in [4], the motion distortion effect was modeled on the simulated point cloud using an analytical approach to reduce the total computation time. For the simulation of a dataset, a virtual vehicle was equipped with a virtual LiDAR sensor on the roof in the middle of the car at a height of 2.3 m, corresponding to the sensor's height on an Audi Q7 with a sensor roof mount. Figure 4 shows the position of the sensor on the virtual vehicle and the orientation of the sensor coordinate system.

FIGURE 4 Position of the sensor and the sensor coordinate system alignment on the virtual vehicle.



Reprinted from Ref. [3] and licensed under the Creative Commons Attribution 4.0 International license (https://creativecommons.org/licenses/by/4.0/)

The virtual LiDAR sensor was configured with a sensor-specific scan pattern with the parameters shown in Table 1. Within the parameters possible in the LiDAR sensor used in this work, the chosen scan pattern exhibited a sufficiently large FoV and a sufficiently accurate resolution for the use case of an ADAS application.

Different trajectories were simulated from a vehicle of interest (VoI) in the FoV of the sensor to simulate the point clouds. The different trajectories were varied in distance between 2.0 m and 50.0 m from the ego vehicle, with different offsets in the x_s-direction and y_s-direction, based on the following and crossing scenarios from the field of ADAS, with different angles between ±30.0° rotated around the z_s-axis. The ego vehicle stands still to represent only the influence of the object's velocity.

TABLE 1 Parameters of the scan pattern used to generate the dataset.

Scan pattern parameter	Value
FoV horizontal	72.0°
FoV vertical	30.0°
Number of scan lines	100
Horizontal spacing	0.4°
Frame rate	5.4 Hz

FIGURE 5 Sensor setup on the measurement vehicles.

However, this circumstance should not cause any difficulties for future applications concerning velocity estimation based on the motion distortion effect since the additional distortion of the point cloud due to the self-motion of the sensor can be compensated based on the data of the IMU of the sensor or the vehicle. The velocities were within a vehicle typical range between 5.55 m/s and 33.33 m/s (20.0–120.0 km/h). In total, 127,249 different point clouds with different VoI positions and velocities were simulated.

Real-World Dataset

Two vehicles were equipped with additional measurement technology to generate a real-world dataset corresponding to the simulated dataset. Two Cube 1 LiDAR sensors from Blickfeld [22] (see 1 and 2 in Figure 5) and the Automotive Dynamic Motion Analyzer (ADMA) ADMA-G:Pro+ from GeneSys Elektronik GmbH [32] were installed in the ego vehicle. Their data was processed using the Robot Operating System (ROS) [33] on computers integrated into the vehicles. An ADMA-Slim [32], also from GeneSys Elektronik GmbH, was installed in the VoI, and the data was transmitted to the ego vehicle via a WiFi antenna and processed in the ego vehicle. The velocity and direction of movement relative to the ego vehicle were recorded in the ego vehicle's coordinate system. Two LiDAR sensors were installed in the ego vehicle to increase the efficiency of the measurement runs, as twice the number of frames can be measured with each measurement run. In addition, the different mounting positions offer different perspectives on a measurement scene, thereby increasing the variance of the data. In addition to the manufacturer's internal calibration, the sensors were again calibrated on the day of the measurement by calibrating the Kalman filter of the ADMA and using a calibration chart for the LiDAR sensor. The temporal calibration was implemented using a global PTP server [34].

The scenarios are driven within the simulation parameter space with the test vehicles described. For this purpose, with the ego vehicle stationary, the VoI drives longitudinally past the ego vehicle in the x_s-direction, laterally past the ego vehicle in the y_s/–y_s-direction and at an angle of 30° to the x_s-axis, as shown in Figure 6.

To avoid the driver having to maintain exact velocities, but instead velocity ranges, all the different directional scenarios are driven in 2.7 m/s (10 km/h) levels from 5.5 m/s to 27.77 m/s (20.0 km/h–100.0 km/h), as shown in Table 2.

This allows to generate a dataset with 2116 different data points.

It should be noted that the operational design domain is limited to the conditions depicted in the dataset, i.e., the scenarios listed and the present weather, slightly cloudy without rainfall, as this also has a considerable influence, as described in [35]. In addition, only vehicles are regarded as objects. Other objects or significant occlusion of the vehicles can lead to a deviating performance of the method.

FIGURE 6 Scenarios for dataset generation.

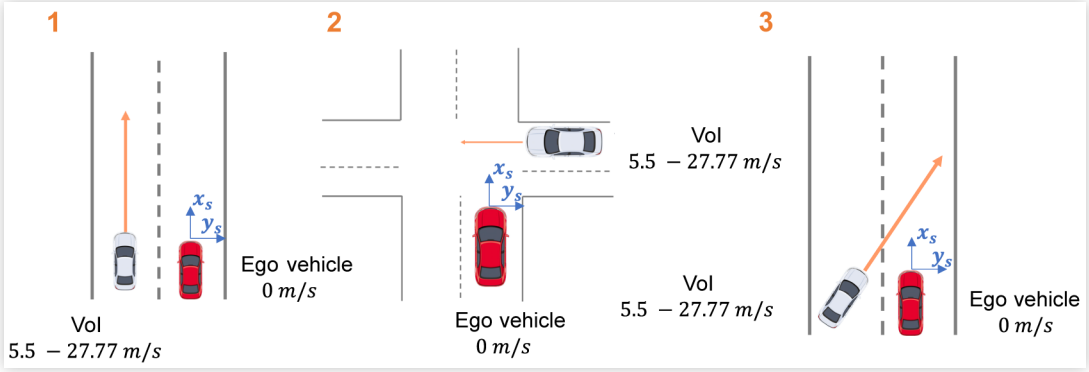


TABLE 2 Driving scenarios for real-world dataset generation.

Scenario	Velocity range in m/s
Parallel left to ego vehicle x_s-direction (Figure 6: 1)	5.5–27.77
Parallel left to ego vehicle $-x_s$-direction (Figure 6: 1)	5.5–27.77
Parallel right to ego vehicle x_s-direction (Figure 6: 1)	5.5–27.77
Parallel right to ego vehicle $-x_s$-direction (Figure 6: 1)	5.5–27.77
Crossing y_s-direction (Figure 6: 2)	5.5–27.77
Crossing $-y_s$-direction (Figure 6: 2)	5.5–27.77
30° to the x_s-axis of the ego vehicle (Figure 6: 3)	5.5–27.77

Coordinate Transformation

To synchronize the sensor coordinate system of the simulation with the real vehicle data from ROS, a coordinate transformation is performed for the point clouds recorded on the real vehicle in the preprocessing. This way, the data was merged into a standard coordinate system, and the sensors were calibrated. This ensures that the viewing directions of all real and virtual sensors in the point cloud are the same, that there is no offset between the coordinate systems, and that all point clouds have a common ground plane.

Results

During the development of the VeloPoints method, described in the "Velocity measurement" section, only synthetic data from the simulation was used for training and hyperparameter optimization of the neural network. This way, the artificial neural network converges for the velocity estimation before the effort of a real measurement campaign is made. After training the neural network and optimizing the hyperparameters shown in Table 3, a combined root mean squared error $RMSE_c = 0.3247$ is achieved for a synthetic test dataset. The $RMSE$ is calculated as a combined $RMSE_c$, as shown in Equation 8, from the number of samples n in the test data, the axis-wise velocity predictions $v_{x,s}$, $v_{y,s}$ and the axis-wise ground truth velocities $\hat{v}_{x,s}$, $\hat{v}_{y,s}$ as two regression values are involved in the prediction. The $RMSE_c$ is not only a KPI for the deviation of the velocities estimated by the model, but, as it is based on the two individual velocity vectors, it is also a KPI for the quality of the velocity direction estimate in which the VoI is moving. Consequently, the velocity estimations of the VeloPoints method have the same quality in all directions of movement of the object. The resulting direction of movement of the VoI can be calculated from the two velocities estimated by the model.

$$RMSE_c = \sqrt{\sum_{i=1}^{n} \frac{\left(\hat{v}_{x,s} - v_{x,s}\right)^2}{n}} + \sqrt{\sum_{i=1}^{n} \frac{\left(\hat{v}_{y,s} - v_{y,s}\right)^2}{n}} \quad \text{Eq. (8)}$$

When the neural network trained on synthetic data is tested with test data from the real measurement data, it only achieves an $RMSE_c = 27.0160$ calculated using formula (8) from all the samples in the test data of the real-world data. According to this, there is a discrepancy between simulated and real-world data for the neural network.

TABLE 3 Optimized hyperparameters.

Hyperparameter	Value
Number of convolutions	2
Number of layers in the multilayer perceptron	6
Network architecture	Convolution + multilayer perceptron + two-layer perceptron
Activation function hidden layer	Relu/linear
Activation function output layer	Relu/linear
Loss function	Huber loss [36]
Initial learning rate	0.0000125
Max momentum	0.99
Batch size	16

Performance on Different Datasets

The neural network is optimized with real training data to ensure it performs better for the later application domain of real-world data. For this purpose, three approaches are compared:

1. Real-world data: The neural network is reinitialized with random weights using the hyperparameter optimum developed on the synthetic data and trained with the real-world data from the measurement campaign.
2. Sequential training: The network already pre-trained from the development process with synthetic data including the hyperparameter set is trained further with real-world data. As a result, the neural network "forgets" part of what it has learned on synthetic bases and converges for the new dataset.
3. Combined training: The neural network is reinitialized with random weights using the hyperparameter optimum developed on the synthetic data and trained on a combined dataset. Therefore, the smaller dataset with real data is randomly combined with the synthetic data and then randomly subdivided into the sub-datasets: train data and test data.

To create a combined dataset for the combined training, 95% synthetic data is combined with 5% real-world data to create a dataset with a total of 42,300 data points. In addition, a second dataset with 98.5% synthetic data and 1.5% real-world data is combined to create a dataset with a total of 127,249 data points. The results of the different neural networks on test data from both domains, purely synthetic and real, are shown in Table 4.

When the neural network is trained with the real-world dataset, it achieves a $RMSE_c$ = 12.0661 on real test data, which is 55.3% lower than the neural network trained with purely synthetic data. However, it only achieves an $RMSE_c$ = 71.7531 for purely synthetic test data, again highlighting the discrepancy between real and synthetic data recognizable for the neural network. In the sequential training approach, the network already trained during development with synthetic data is trained additionally with real-world data. After training, the neural network achieves an optimal $RMSE_c$ = 11.1592 for real test data, 58.7% lower than the neural network trained purely on synthetic data. Using the combined training approach with a share of 5% real-world data, the best neural network on real test data achieves an 86.5% smaller $RMSE_c$ = 3.6350 compared to the neural network trained with purely synthetic data. If a neural network is trained with the larger dataset containing 1.5% real-world data, it only achieves an $RMSE_c$ = 7.3603, which is 13.7% points less improvement than the dataset with 5%. About the purely synthetic data, however, the model achieves the best $RMSE_c$ = 0.2538, which is 21.8% better than the model trained purely on synthetic data, indicating a conformity between the synthetic and real-world data.

With the third training approach, training on a combined dataset with a share of 5% real-world data, the best neural network convergence for real-world data is achieved in this case.

Considering the discrepancy in the $RMSE_c$s between real and synthetic data, the sequential training method achieves the most balanced prediction for both datasets with an $RMSE_c$ for simulated data that is 69.69% higher than on real-world data. However, the discrepancy existing for the neural network is recognizable across all training methods. The discrepancy between the simulated and real-world data can be attributed to three reasons: the simulation model of the sensor, the simulated scenarios, and the measurement uncertainties in the real measurements. The high-fidelity sensor model of the Cube1 sensor is highly accurate. As described in [4], it only has a mean absolute percentage error (MAPE) of 0.08% for the distance error. In addition, the model has a MAPE of 9.20% for the intensity values of the points in a point cloud compared to real-world data. The measurement error of the real measurement with the ADAM-Slim is within a RMSE = 0.0111 m/s [22]. In addition, the velocity was measured at a frequency of 100 Hz, which is a sufficient sampling rate in relation to the 5.4 Hz of the LiDAR sensor, especially in terms of the comparatively low dynamics of a vehicle. The modeled scenario in the simulation offers a variety of possible deviations from reality. It is very time-consuming to model the exact intensities in the point cloud, as the current reflectivities of all the different materials in the scenario would have to be known for this. In addition, there are deviations due to different vehicle parameters between the simulation and reality, as well as due to geometric differences such as the road profile. Despite the simple scenarios selected, there are numerous possible deviations from reality in the scenario creation. It is impossible to define a precise reason for the error due to the black box of the artificial neural network model. However, it can be assumed that the real measured velocities have sufficient accuracy for the application. To attribute the discrepancy to the sensor model or differences in the scenarios, the possible influence of the intensities on the velocity predictions is examined in the following.

TABLE 4 RMSEs of the neural networks based on different test and training datasets.

Test data Train data	Sim. data $RMSE_c$ (m/s)	Real-world data $RMSE_c$ (m/s)
Sim. data	0.3247	27.0160
Real-world data	71.7531	12.0661
Sim. data + real-world data (sequential)	18.9367	11.1592
Sim. data + 5% real-world data	46.7132	3.6350
Sim. data + 1.5% real-world data	0.2538	7.3603

Effect of Intensity as Input Parameter

The intensities of the individual points of a point cloud are very complex to simulate, as they depend on the very various reflectivities of the different materials, which also vary, for example, due to pollution or weather conditions. To determine the effect of the intensities on the velocity estimation and to investigate the influence of the deviation between the simulated and real measured intensity on the velocity estimates of the artificial neural network, multiple networks were trained without intensities as input variables. For this purpose, the artificial neural networks were trained again on the synthetic, real, and combined data, with 5% real-world data tested, and the results obtained were compared with the networks in the previous section. Table 5 shows the $RMSE_c$s achieved.

For the input parameters x_s, y_s, z_s for each point in the point cloud, the neural networks show an average $RMSE_c$ that is 14.2356% higher than for the input parameters of x_s, y_s, z_s + intensity. The networks trained with only x_s, y_s, z_s coordinates are tested with the synthetic and real-world data. In that case, the $RMSE_c$s shown in

TABLE 5 $RMSE_c$s of the neural networks based on different test and training datasets.

Input variables Train/test data	x_s, y_s, z_s + intensity $RMSE_c$ (m/s)	x_s, y_s, z_s $RMSE_c$ (m/s)
Sim. data	0.3247	0.3818
Real-world data	12.0661	13.2709
Sim. data + 5% real-world data	3.0045	3.4593

Table 6 are obtained, with the corresponding $RMSE_c$s of the models, including intensities for comparison in brackets.

Even without intensities as input parameters, there is a discrepancy between the synthetic and real-world datasets for the models. However, deviations in the intensities only contribute to a small extent to this. The deviations are, therefore, due to the abstract detailing of the scenarios.

Conclusions

This article shows that the VeloPoints method is converging for real measurement data. The discrepancy between real and synthetic data for the neural network was also investigated. For this purpose, a real-world dataset was measured on the proving ground, and a corresponding synthetic dataset was simulated. Based on both datasets, a discrepancy was determined between the simulated and real-world data for VeloPoints. By comparing different training strategies, it is shown that by enriching synthetic data with 5% real measured data, a decrease of up to 69.87%, compared to a model only trained with real-world data, is achieved in $RMSE_c$ the velocity predictions tested with real-world data. It is also shown that training a newly initialized model with a combined dataset of synthetic and real-world train data leads to a 67.42% better $RMSE_c$ for real-world test data than sequential training with a pre-trained model using synthetic train data. In addition, it is shown that the intensities of the individual points in the point cloud have a negligible effect on the velocity estimation of VeloPoints and that the neural network produces the prediction via geometric correlations. By examining the existing discrepancy between simulated and real-world data for the neural network, the influences of scenario modeling could predominantly be determined. Finally, developing neural networks for LiDAR applications can be efficiently achieved with highly accurate simulation models. Combining the data can reduce the discrepancy between the real and simulated data. The data discrepancy strongly depends on the degree of abstraction of the modeled scenarios.

TABLE 6 $RMSE_c$s of the neural networks based on different test and training datasets.

Test data Train data	Sim. data (x_s, y_s, z_s) $RMSE_c$ (m/s)	Real-world data (x_s, y_s, z_s) $RMSE_c$ (m/s)
Sim. data	0.3818 (0.32)	47.1356 (27.01)
Real-world data	72.1951 (71.75)	13.2709 (12.06)
Sim. data + 5% real-world data	50.01 (46.71)	2.95 (3.63)

Contributions and Outlook

This article provides various contributions. First, it shows that the VeloPoints method also converges for real data, with geometric correlations being more significant features for this method than intensity. In addition, a real dataset was presented and used to analyze the effect of datasets from the real and simulation domains on the VeloPoints method. As a further contribution, it was shown how the best possible results could be achieved with little real data, and cost-efficient simulated data through the cross-domain combination of data and different training methods without increasing the parameter space during training and, thus, the computational effort. This shows the challenge for dataset generation in simulation and reality for the efficient development of deep learning-based methods.

Based on these results, in the future, the optimal degree of abstraction of the simulated scenarios should be assessed. It is expected that with a low degree of abstraction, the discrepancy between the synthetic and real-world data for the neural network decreases, but the design effort and computing time for the simulation increases. For this, the relationship between real and simulated data, an expansion of the ODD, and the related performance changes of the VeloPoints method need to be examined. Based on the results, in the subsequent step, transfer learning will be investigated as a further training option to enable the transition from a synthetic domain to the real domain for the neural network. This offers the possibility of saving training time, as the synthetically pre-trained models can be used, but it has the disadvantage that a further optimization parameter is introduced. Furthermore, a combination of the VeloPoints method with object detection is considered. In addition, the VeloPoints method is to be transferred to the application, and the related challenges, such as computational requirements, integration with existing vehicle systems, and scalability issues across different types of vehicles, are to be investigated. Moreover, a benchmark for velocity estimation about human drivers and tracking or Doppler-based methods, including those based on other sensor modalities, need to be performed, and the advantage of the estimated velocity information for sensor data fusion has to be worked out.

Acknowledgements

The authors thank the Institute for Driver Assistance Systems and Connected Mobility (IFM) for providing technical assistance and equipment for measurements.

Contact Information

Lukas Haas, corresponding author
lukas.haas@hs-kempten.de

Definitions/Abbreviations

ACC - Adaptive cruise control
AD - Autonomous driving
ADAS - Advanced driver assistance systems
ADMA - Automotive dynamic motion analyzer
FCW - Forward collision warning
FMCW - Frequency-modulated continuous wave
FoV - Field of view
LDW - Lane departure warnings
LiDAR - Light detection and ranging sensors
MAPE - Mean absolute percentage error
MEMS - Microelectromechanical system
OoI - Object of interest
RMSE - Root mean squared error
ROS - Robot operating system
SLAM - Simultaneous localization and mapping
ToF - Time of flight
VoI - Vehicle of interest

References

1. Rachman, A., "3D-LiDAR Multi Object Tracking for Autonomous Driving," MS thesis, Delft University of Technology, 2017.
2. Krämer, S., "LiDAR-Based Object Tracking und Shape Estimation," PhD dissertation, Karlsruher Institut für Technologie (KIT), 2020.
3. Haas, L., Haider, A., Kastner, L., Zeh, T. et al., "Velocity Estimation from LiDAR Sensors Motion Distortion Effect," *Sensors* 23, no. 23 (2023): 9426, doi:https://doi.org/10.3390/s23239426.
4. Haider, A., Haas, L., Koyama, S., Elster, L. et al., "Modeling of Motion Distortion Effect of Scanning LiDAR Sensors for Simulation-Based Testing," *IEEE Access* 12 (2024): 13020-13036, doi:10.1109/ACCESS.2024.3355066.
5. Liesner, L., *Automatisierte Funktionsoptimierung von Adaptive Cruise Control* (Herzogenrath, Germany: Shaker Verlag, 2017)
6. Haider, A. et al., "Performance Evaluation of MEMS-Based Automotive LiDAR Sensor and Its Simulation Model as per ASTM E3125-17 Standard," *Sensors* 23, no. 6 (2023): 3113.
7. Gu, Y., Cheng, H., Wang, K., Dou, D. et al., "Learning Moving-Object Tracking with FMCW LiDAR," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan, October 23–27, 2022, 3747-3753.
8. Roriz, R., Cabral, J., and Gomes, T., "Automotive LiDAR Technology: A Survey," *IEEE Transactions on Intelligent Transportation Systems* 23, no. 7 (2022): 6282-6297, doi:10.1109/TITS.2021.3086804.
9. Lee, J., Hong, J., and Park, K., "Frequency Modulation Control of an FMCW LiDAR Using a Frequency-to-Voltage Converter," *Sensors* 23 (2023): 4981.
10. Kim, T., "Realization of Integrated Coherent LiDAR," PhD dissertation, UC Berkeley, 2019.
11. Seidel, F., "Implementierung und Evaluierung einer radarbasierenden Abstandsmessung für einen Quadrocopter," Julius-Maximilians-Universität Würzburg, 2013.
12. Kim, C., Jung, Y., and Lee, S., "FMCW LiDAR System to Reduce Hardware Complexity and Post-Processing Techniques to Improve Distance Resolution," *Sensors* 20 (2020): 6676, doi:https://doi.org/10.3390/s20226676.
13. Petit, F., "Entmystifizierung von LiDAR—Ein Überblick über die LiDAR-Technologie," August 19, 2020, accessed December 12, 2023, https://www.blickfeld.com/de/blog/was-ist-lidar/#:~:text=Wie%20funktioniert%20die%20Technologie%3F,zum%20 Detektor%20des%20Sensors%20zur%C3%BCckkehren.
14. InnovizOne, "LiDAR Technology Automotive-Grade Solid-State LiDAR," accessed December 12, 2023, https://innoviz.tech/innovizone#top
15. International Electrotechnical Commission, "Safety of Laser Products—Part 1: Equipment Classification and Requirements," Technical Report IEC-60825-1; International Electrotechnical Commission, Geneva, Switzerland, 2014.
16. Voigt, E. and Kremsreiter, M., "LiDAR in Anwendung," IAV GmbH Ingenieurgesellschaft Auto und Verkehr, Berlin, Germany, 2020.
17. Thakur, R., "Scanning LIDAR in Advanced Driver Assistance Systems and Beyond: Building a Road Map for Next-Generation LIDAR Technology," *IEEE Consum. Electron. Mag.* 5 (2016): 48-54.
18. Ballard, P. and Vacherand, F., "Simulation and Understanding of Range Images Acquired in Fast Motion," in *Proceedings of the IEEE International*

Conference on Robotics and Automation, San Diego, CA, May 8–13, 1994, 2242-2247.

19. Livox Technology Company Limited Co. Ltd., "Livox Open Source Sharing: About LiDAR Distortion Removal," December 20, 2021, accessed August 14, 2024, https://www.livoxtech.com/showcase/211220.
20. Yang, W., Gong, Z., Huang, B., and Hong, X., "Lidar With Velocity: Correcting Moving Objects Point Cloud Distortion from Oscillating Scanning Lidars by Fusion with Camera," *IEEE Robot Autom. Lett.* 7 (2022): 8241-8248.
21. Blickfeld GmbH, "Blickfeld Software Manual, Scanpattern," accessed December 14, 2023, https://docs.blickfeld.com/cube/latest/scan_pattern.html.
22. Blickfeld GmbH, "Cube 1 v2.1, Datasheet," 2022, accessed December 12, 2023, https://www.blickfeld.com/wp-content/uploads/2022/10/blickfeld_Datasheet_Cube1_v2.1.pdf.
23. Renzler, T., Stolz, M., Schratter, M., and Watzenig, D., "Increased Accuracy for Fast Moving LiDARS: Correction of Distorted Point Clouds," in *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, Dubrovnik, Croatia, May 25–28, 2020, 1-6.
24. Muro, S., Yoshida, I., Hashimoto, M., and Takahashi, K., "Moving-Object Detection and Tracking by Scanning LiDAR Mounted on Motorcycle Based on Dynamic Background Subtraction," *Artif. Life Robot.* 26 (2021): 412-422.
25. Zheng, C., Yan, X., Zhang, H., Wang, B. et al., "Beyond 3D Siamese Tracking: A Motion-Centric Paradigm for 3D Single Object Tracking in Point Clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, LA, June 18–24, 2022, 8111-8120.
26. Fang, Z., Zhou, S., Cui, Y., and Scherer, S., "3D-SIAMRPN: An End-to-End Learning Method for Real-Time 3D Single Object Tracking Using Raw Point Cloud," *IEEE Sensors J.* 21 (2020): 4995-5011.
27. Giancola, S., Zarzar, J., and Ghanem, B., "Leveraging Shape Completion for 3D Siamese Tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, June 15–20, 2019, 1359-1368.
28. Yin, T., Zhou, X., and Krahenbuhl, P., "Center-Based 3D Object Detection and Tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, TN, June 20–25, 2021, 11784-11793.
29. Chiu, H.K., Li, J., Ambrus, R., and Bohg, J., "Probabilistic 3D Multi-Modal, Multi-Object Tracking for Autonomous Driving," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, May30–June 5, 2021, 14227-14233.
30. Lang, A.H., Vora, S., Caesar, H., Zhou, L. et al., "PointPillars: Fast Encoders for Object Detection from Point Clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, June 15–20, 2019.
31. IPG Automotive GmbH, "CarMaker," accessed January 22, 2024, https://ipg-automotive.com/de/produkte-loesungen/software/carmaker/.
32. GeneSys Elektronik GmbH, "Automotive Testing Equipment," accessed December 12, 2023, http://www.arnold-messtechnik.de/GeneSys_Produktkatalog_ger_19_05_04_2web.pdf.
33. ROS, "ROS—Roboter Operating System," accessed December 12, 2023, https://www.ros.org/.
34. Eidson, J.C., Fischer, M., and White, J., "IEEE-1588™ Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," in *Proceedings of the 34th Annual Precise Time and Time Interval Systems and Applications Meeting*, Reston, VA, 2002.
35. Haider, A. et al., "A Methodology to Model the Rain and Fog Effect on the Performance of Automotive LiDAR Sensors," *Sensors* 23, no. 15 (2023): 6891.
36. Huber, P.J., "Robust Estimation of a Location Parameter," *Ann. Math. Statist.* 35 (1964): 73-101.