Automatic Label Injection Into Local Infrastructure LiDAR Point Cloud for Training Data Set Generation

ZSOLT VINCZE, ANDRÁS RÖVID, AND VIKTOR TIHANYI

Department of Automotive Technologies, Faculty of Transportation Engineering and Vehicle Engineering, Budapest University of Technology and Economics, 1111 Budapest, Hungary

Corresponding author: Zsolt Vincze (vincze.zsolt@kjk.bme.hu)

This work was supported by the National Research Development and Innovation Fund through the Budapest University of Technology and Economics (TKP2020 National Challenges Subprogram) Based on the Charter of Bolster issued by the National Research Development and Innovation Office under the auspices of the Ministry for Innovation and Technology under Grant BME-NC.

ABSTRACT The representation of objects in LiDAR point clouds is changed as the height of the mounting position of sensor devices gets increased. Most of the available open datasets for training machine learning based object detectors are generated with vehicle top mounted sensors, thus the detectors trained on such datasets perform weaker when the sensor is observing the scene from a significantly higher viewpoint (e.g. infrastructure sensor). In this paper a novel Automatic Label Injection method is proposed to label the objects in the point cloud of the high-mounted infrastructure LiDAR sensor based on the output of a well performing “trainer” detector deployed at optimal height while considering the uncertainties caused by various factors described in detail throughout the paper. The proposed automatic labeling approach has been validated on a small scale sensor setup in a real-world traffic scenario where accurate differential GNSS reference data where also available for each test vehicle. Furthermore, the concept of a distributed multi-sensor system covering a larger area aimed for automatic dataset generation is also presented. It is shown that a machine learning based detector trained on differential GNSS-based training dataset performs very similarly to the detector retrained on a dataset generated by the proposed Automatic Label Injection technique. According to our results a significant increase in the maximum detection range can be achieved by retraining the detector on viewpoint specific data generated fully automatically by the proposed label injection technique compared to a detector trained on vehicle top mounted sensor data.

INDEX TERMS Object label, automatic label injection, roadside infrastructure sensors, training dataset.
Adverse Driving Conditions Dataset [8], Ford Campus LiDAR dataset [9], Sydney Urban Objects [10], Stanford Track Collection [11] contain data acquired by various types of sensors one of which is the LiDAR (Light Detection And Ranging). These sensors are mounted on top of different types of passenger cars. At these mounting heights, the representation of objects in the point cloud differs significantly from the case, when the sensor is deployed in the infrastructure (the mounting height of infrastructure sensors might be several meters). LiDAR-based object detector networks trained on these open datasets, perform poorly on point clouds acquired by infrastructure LiDAR sensors.

### B. RELATED WORKS

In [12] the different representation of objects in point clouds acquired by different types of LiDAR sensors is handled by training a neural network to increase the density of points along the surface of objects. A recently presented unique infrastructure data set [13] on the other hand, enables to train machine learning-based object detectors to detect objects from a specific (higher) viewpoint. Creating sensor and mounting position specific training datasets for infrastructure LiDAR sensors manually requires significant time and effort, nevertheless in case of manual labeling there might be relevant variance in the quality of labels. One solution to overcome this problem is to generate training dataset in a simulated environment, such as described in [14]. However, simulation based datasets lack all the real-life noise, which can reduce the performance of the trained detector on real-life data.

Another way to find objects of interest in point clouds yielded by LiDAR sensors is to filter out the background, thus the remaining points will represent the searched objects regardless their representation in the point cloud. In [15] the authors aggregated consecutive point cloud frames, then with map cleaning methods they separated the estimated static map from the moving objects. To generate static background with vehicle mounted LiDAR sensors authors of [16] made recordings at the same location at several different occasions, building upon the concept that mobile objects are unlikely to stay in the same place in every recording session. In case of infrastructure mounted sensors the background remains the same over time due to the fix installation, however the shaking of the mounting structure must be considered. Therefore, in [17] the authors voxelised the point cloud, then modeled the average height and the number of points for each voxel as Gaussian distribution in order to filter out the static background.

Because of the representation of objects in the point cloud is dependent of the channel number and beam configuration and the mounting height of the given LiDAR sensor, the corresponding detector network requires retraining if the used LiDAR sensor has different features than the one which recorded the dataset which the detector network was originally trained on. In case of background segmentation, good estimation of the bounding boxes of objects can only be made when the objects are represented in the point cloud with larger number of points. This criteria limits the maximum detection distance of the trained detector network, because with the increase of the distance between the object and the LiDAR sensor, the amount of points representing the object reduces. For example, in [17] it means that the maximum detection distance of the trained detector network was set to 66.32m.

### C. CONTRIBUTION

The main contribution of this work is an automatic label injection method, which realizes the automatic transformation of object labels (provided by an object detector with good performance which processes the point cloud frames of a temporarily deployed LiDAR sensor) to the recorded point clouds acquired by a high mounted infrastructure LiDAR sensor by considering the time synchronisation and sensor calibration related errors. Here we would like to emphasize the problems caused by the independent operation of the LiDAR sensors, namely their laser beams are not synchronised, thus moving objects are scanned in different moments which (if not handled properly) may have significant impact on the accuracy of the injected labels. The proposed system was validated under real conditions on real-world data.

Further contribution is a conceptual distributed multi-sensor based labeling system, which relies on the proposed Automatic Label Injection method to create a training dataset being tailored for the specific infrastructure sensors deployed at elevated positions into the given traffic environment. With the help of such tailored datasets, the performance of the detectors might be optimised for these specific LiDAR sensors and their specific locations. This system can be adopted by intelligent traffic systems [18], which have the capability of sharing perception and object level information.

With the help of our proposed Automatic Label Injection method, LiDAR sensor and sensor placement specific training datasets can be generated automatically. This means that for an arbitrary LiDAR device (independently on its laser beam distributions, sensitivity and other sensor specific factors) and its hosting environment (where the sensor is deployed), a tailored training dataset suited for that particular setup can be generated. Furthermore, for the automated dataset generation, there is no need for any sensors (such as for instance GNSS device, LiDAR etc.) to be present in vehicles. Furthermore, the trained detector network which processes the point clouds from the infrastructure sensor can estimate appropriate bounding boxes from less object representing points, thus it has greater detection range than the detectors which were trained on data obtained by background segmentation techniques.

### D. STRUCTURAL OVERVIEW

This article consists of the following sections: II-A gives the detailed description of the proposed automatic label injection procedure, II-B introduces the concept of the training dataset generator system, II-C shows how the detectability of objects can be increased by training on a custom...
dataset which is tailored for the mounting position of the sensor. Section III-A contains the description of the measurement setup. The Intersection Over Union (IOU) metric of GNSS-based ground truth and the labels given by the detector (which processes the point clouds of the Trainer station) is described in section III-B. Section III-C is devoted to the description of the IOU metric in case of the GNSS-based ground truth and the injected labels. Section III-D reports performance metrics of the detectors trained on the following datasets independently: KITTI, GNSS based, the proposed label injection-based. Section IV is devoted to the evaluation of the measurement results reported in III-C and III-D and the discussion of the limitations of the proposed method.

II. METHODOLOGY

A. AUTOMATIC LABEL INJECTION

In order to create a training dataset for a neural network which is aimed for detecting objects in LiDAR point clouds, each frame must contain a point cloud and labels for the objects represented in it. The problem is how to generate the labels for the objects to avoid the huge amount of human effort. The proposed automatic label injection method relies on the detection capabilities of a neural network trained on open datasets. This detector network processes the point clouds of a LiDAR sensor, which is deployed at similar height as vehicle mounted LiDARs, thus the neural network can operate with optimal performance. Let us call this low mounted LiDAR unit and the detector network together as the TRAINER SENSOR. On the other hand the TRAINED SENSOR is a fix mounted infrastructure sensor deployed at an increased height above the road surface in order to observe the traffic from an elevated position. From such a viewpoint, the point cloud representation of traffic participants is changed, which leads to degraded detection performance. The detector of the Trained sensor needs to be trained on a custom dataset in order for the detector to adapt to the altered patterns of object representation caused by the higher viewpoint of the LiDAR. The point cloud of each frame of the dataset is recorded by the Trained sensor. The labels for all the objects in the saved point cloud are provided by the Trainer sensor itself. The labels are processed and transformed from the local coordinate system of the Trainer sensor into the local coordinate system of the Trained one. Fig. 1 shows the Trainer and the Trained sensors during operation, and Fig. 6 gives an overview of the Automatic Label Injection process. In the followings the proposed method is described in detail.

First, the point cloud recorded by the Trainer sensor goes through a two stage preparation process. Let us denote the \( k \)th frame acquired by the sensor by \( F_k \). When the sensor starts recording the frame \( F_k \), it starts the measurement from a certain yaw angle \( \omega \), and sweeps the environment with laser beams in clockwise direction. The whole 360° horizontal field of view of the sensor is divided into \( N_{\text{seg}} \) segments where \( N_{\text{seg}} \in \{512, 1024, 2048\} \) depending on the actual configuration of the lidar sensor. Let us denote these segments by \( S_q \), where \( q = 1..N_{\text{seg}} \). The LiDAR sensor records the time \( t_{S_q} \) when the scanning of the segment \( S_q \) was finished. From the scanning results of all segments a single point cloud is composed and the timestamp of the last segment \( S_{N_{\text{seg}}} \) is assigned as the timestamp \( t_{F_k} \) of the whole point cloud frame. Let us define \( f_S : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \) which maps the point index of point \( p_{i,k} \) to the index of its corresponding segment. During the first preparation stage, each point in the point cloud frame on top of the \( x, y, z \) parameters, gets the timestamp \( t_{S_q} \) of the corresponding segment. Therefore each point of the \( k \)th frame is represented as follows:

\[
p_{i,k} = [x_{i,k}, y_{i,k}, z_{i,k}, t_{S_q}],
\]

where \( q = f_S(i, k) \).

In the second stage, the whole point cloud is rotated to a pose which is considered to be optimal for the object detector network Let us denote this rotation by \( R_{\text{det}} \).

As next step the point cloud is processed by the PointPillars [19] detector network which was trained on the KITTI open dataset. For the object detection phase, any well performing detector network can be utilised instead of the one considered in this paper. We have selected the PointPillars network because it performs well on the KITTI benchmark and has the ability to process point clouds in real-time (~16ms). In order to reach our future goal, namely the online operation of the proposed method, high performance (fast and accurate) object detectors have to be considered.

When processed by the PointPillars detector network, the point cloud is rasterized along the x-y plane with a preset resolution and divided into so called pillars. Each point in the point cloud is assigned as the timestamp \( t_{S_q} \) when the scanning of the segment \( S_q \) was finished. The LiDAR sensor records the time \( t_{S_q} \) when the scanning of the segment \( S_q \) was finished. From the scanning results of all segments a single point cloud is composed and the timestamp of the last segment \( S_{N_{\text{seg}}} \) is assigned as the timestamp \( t_{F_k} \) of the whole point cloud frame. Let us define \( f_S : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \) which maps the point index of point \( p_{i,k} \) to the index of its corresponding segment. During the first preparation stage, each point in the point cloud frame on top of the \( x, y, z \) parameters, gets the timestamp \( t_{S_q} \) of the corresponding segment. Therefore each point of the \( k \)th frame is represented as follows:

\[
p_{i,k} = [x_{i,k}, y_{i,k}, z_{i,k}, t_{S_q}],
\]

where \( q = f_S(i, k) \).

In the second stage, the whole point cloud is rotated to a pose which is considered to be optimal for the object detector network Let us denote this rotation by \( R_{\text{det}} \).

As next step the point cloud is processed by the PointPillars [19] detector network which was trained on the KITTI open dataset. For the object detection phase, any well performing detector network can be utilised instead of the one considered in this paper. We have selected the PointPillars network because it performs well on the KITTI benchmark and has the ability to process point clouds in real-time (~16ms). In order to reach our future goal, namely the online operation of the proposed method, high performance (fast and accurate) object detectors have to be considered.

When processed by the PointPillars detector network, the point cloud is rasterized along the x-y plane with a preset resolution and divided into so called pillars. Each point in the point cloud is assigned as the timestamp \( t_{S_q} \) when the scanning of the segment \( S_q \) was finished. The LiDAR sensor records the time \( t_{S_q} \) when the scanning of the segment \( S_q \) was finished. From the scanning results of all segments a single point cloud is composed and the timestamp of the last segment \( S_{N_{\text{seg}}} \) is assigned as the timestamp \( t_{F_k} \) of the whole point cloud frame. Let us define \( f_S : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \) which maps the point index of point \( p_{i,k} \) to the index of its corresponding segment. During the first preparation stage, each point in the point cloud frame on top of the \( x, y, z \) parameters, gets the timestamp \( t_{S_q} \) of the corresponding segment. Therefore each point of the \( k \)th frame is represented as follows:

\[
p_{i,k} = [x_{i,k}, y_{i,k}, z_{i,k}, t_{S_q}],
\]

where \( q = f_S(i, k) \).

In the second stage, the whole point cloud is rotated to a pose which is considered to be optimal for the object detector network Let us denote this rotation by \( R_{\text{det}} \).

As next step the point cloud is processed by the PointPillars [19] detector network which was trained on the KITTI open dataset. For the object detection phase, any well performing detector network can be utilised instead of the one considered in this paper. We have selected the PointPillars network because it performs well on the KITTI benchmark and has the ability to process point clouds in real-time (~16ms). In order to reach our future goal, namely the online operation of the proposed method, high performance (fast and accurate) object detectors have to be considered.

When processed by the PointPillars detector network, the point cloud is rasterized along the x-y plane with a preset resolution and divided into so called pillars. Each point in the point cloud is assigned as the timestamp \( t_{S_q} \) when the scanning of the segment \( S_q \) was finished. The LiDAR sensor records the time \( t_{S_q} \) when the scanning of the segment \( S_q \) was finished. From the scanning results of all segments a single point cloud is composed and the timestamp of the last segment \( S_{N_{\text{seg}}} \) is assigned as the timestamp \( t_{F_k} \) of the whole point cloud frame. Let us define \( f_S : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \) which maps the point index of point \( p_{i,k} \) to the index of its corresponding segment. During the first preparation stage, each point in the point cloud frame on top of the \( x, y, z \) parameters, gets the timestamp \( t_{S_q} \) of the corresponding segment. Therefore each point of the \( k \)th frame is represented as follows:

\[
p_{i,k} = [x_{i,k}, y_{i,k}, z_{i,k}, t_{S_q}],
\]

where \( q = f_S(i, k) \).

In the second stage, the whole point cloud is rotated to a pose which is considered to be optimal for the object detector network Let us denote this rotation by \( R_{\text{det}} \).

As next step the point cloud is processed by the PointPillars [19] detector network which was trained on the KITTI open dataset. For the object detection phase, any well performing detector network can be utilised instead of the one considered in this paper. We have selected the PointPillars network because it performs well on the KITTI benchmark and has the ability to process point clouds in real-time (~16ms). In order to reach our future goal, namely the online operation of the proposed method, high performance (fast and accurate) object detectors have to be considered.
this information the eight corners of the bounding box are determined for each label $l^{(k)}_i$. Every point $p_{i,k}$ of the $k$th frame is then evaluated whether or not it is encapsulated by one of the bounding boxes and if it is, it is assigned to the corresponding object. Every point $p_{i,k}$ holds the time stamp $t_{S}$, where $q = f_{S}(i,k)$ which is the time when the corresponding segment was measured. Let us define the following membership function:

$$
\mu(i,j,k) = \begin{cases} 
1, & \text{if } p_{i,k} \in j\text{th bounding box} \\
0, & \text{otherwise} 
\end{cases}
$$

(2)

Let $t^{(j)}_{obj}$ denote the timestamp assigned to the $j$th object. $t^{(j)}_{obj}$ is determined as the mean of timestamps of all the points which have been assigned to the $j$th object, i.e.:

$$
t^{(j)}_{obj} = \frac{\sum_{i=1}^{N_{pts}} \mu(i,j,k)t_{S}(i,k)}{\sum_{i=1}^{N_{pts}} \mu(i,j,k)},
$$

(3)

where and $N_{pts}$ represents the number of points of the $k$th frame. The time stamps are written in Unix Epoch Clock format. Fig. 3 shows the segmented horizontal field of view of the Trainer Sensor.

By using the estimated rotations $R_{UTM \to Tr}$, $R_{UTM \to Td}$ and translations $t_{UTM \to Tr}$, $t_{UTM \to Td}$ from the UTM to the local coordinate system (see Section III-A) of both devices and the saved rotation $R_{det}$ (see above in this section) from the preparation step, the corner points of the bounding boxes are transformed from the local coordinate system of the Trainer sensor into the local coordinate system of the Trained sensor. Additionally, each transformed bounding box $l^{(k)}_j$ are augmented with the time $t^{(j)}_{obj}$. The pseudo code for the above described process is listed in Table 1.

The Trained Sensor has a temporary storage which operates in a First In First Out (FIFO) manner for the recorded point clouds $F_{FIFO} = \{F_{FIFO}(0), \ldots, F_{FIFO}(N_{FIFO}-1)\}$, where $N_{FIFO}$ denotes the maximum number of frames in the queue. When a new frame is recorded by the Trained Station, this queue is updated accordingly, thus its first element (the oldest frame) is dropped. In the Trained Station, each frame in addition to its timestamp is augmented by the timestamps $t^{(0)}_{S_0}, \ldots, t^{(N_{seg})}_{S_N}$ corresponding to the individual segments. We have set the size of this FIFO queue to contain 25 frames at most, i.e. $N_{FIFO} = 25$. When a new set of labels $L^{(k)}_j = \{l^{(k)}_1, l^{(k)}_2, \ldots, l^{(k)}_{N^{obj}}\}$ with timestamp $t^{(0)}_{obj}$ arrives to the Trained Station, the frame having the closest time stamp is selected from the queue based on the segment time stamps of the frames.

The timestamp for the point cloud selection from the FIFO storage is determined as follows: For the first label from the given frame $l^{(0)}_0$, the center point of the bottom side of the bounding box is calculated. It is sufficient because the goal is to find the frame measured by the Trained sensor being nearest in time to the scan time $t^{(j)}_{obj}$ of the objects yielded by the Trained Station. The labels arriving from the Trained Station fall in a narrow region of the perception field of the Trained Station, therefore the difference in the scan times $t^{(j)}_{obj}$ of the detected objects is much less than the time difference between two consecutive frames $(t^{(j)}_{obj} - t^{(j)}_{obj}) \ll t^{(j)}_{F_k} - t^{(j+1)}_{F_k}$, where $j_1, j_2 = 1 \ldots, N_{seg}, j_1 \neq j_2$. Let $v$ represent the vector which points from the origin of the local coordinate system of the LiDAR sensor to the center point of the bottom side of the bounding box of the $j$th object (with index 0). Let us denote by $\alpha$ the angle between $v$ and the $x$ axis of the local coordinate system of LiDAR sensor. The range of $\alpha$ is between $0^\circ$ and $180^\circ$ and it is symmetric to the $x$ axis, therefore the evaluation of the signs of the coordinates of the center point is also necessary to calculate $\alpha'$. In range

---

**TABLE 1. Pseudo code for the Trainer Sensors process.**

| READ point cloud frame |
|------------------------|
| MESSAGE point cloud into (segments, channels, point features) |
| CONCATENATE segment timestamps to points |
| ROTATE point cloud |
| DETECT objects with Point Pillars detector |
| FOR each object label |
| DETERMINE bounding box corners |
| FOR each point in point cloud frame |
| FOR each object bounding box |
| IF bounding box contains point |
| ASSIGN point to object |
| FOR each object |
| CALCULATE mean timestamp |
| ASSIGN mean time stamp to object as object scan time |
| FOR each bounding box corner |
| TRANSFORM coordinates from Trainer into Trained local system |
| AVERT object scan time to bounding box corner list |
| SEND bounding box list to Trained sensor |
between 0° and 360°. \( a' \) determines in which segment \( S'_q \) of the Trained Sensor the center point of the object was scanned. Let us denote this segment index by \( q_r \). When \( q_r \) is known, from all frames in the queue, the timestamp corresponding to the segment \( S'_q \) is taken and used to form a list of timestamps as follows: \( t'_{S_q \_list} = \{ t'_{S_q}^{(0)}, \ldots , t'_{S_q}^{(N_{FIFO}-1)} \} \). The timestamps in the composed list have the same order as the frames in the queue. Then the timestamp \( t'_{S_q \_list} \) is selected which is nearest in time with the scan time of the 1st object \( t'_{obj}^{(0)} \). It is now obvious that the list index of the selected timestamp \( t'_{S_q \_list} \) m where \( m = 1..N_{FIFO} - 1 \) is the same as the index of the corresponding frame \( F'_{FIFO} \), which gets paired with the label set \( L_k \). The pseudo code for the above explained process is described in Table 2.

Although the labels \( L_k \) and the associated frame \( F_k \) of the Trained Sensor are close in time, there may still be some time difference, since the LiDAR devices operate independently, i.e. their scanning beams are not aligned. Therefore, a given object in the point cloud and the corresponding label may not fit exactly. To reduce such inaccuracies the pose of the bounding box is refined by a box fitting method described later in this section. According to the time difference between the labels and the point cloud the labels might be ahead or behind in time wrt. the point cloud. Due to the time difference between the labels and the point cloud some of the object points may fall outside the bounding box. In order to consider all the points belonging to the object an increased point searching region is considered which is based on the original bounding box but has increased length and width. Such extended point searching region is used during the mentioned box fitting process. The length increase is determined from the distance which the object can travel during the maximum time difference between the label and the point cloud. The speed limit of Hungarian highway roads is 130 km/h. The LiDAR sensors are operating with 20 frame/s configuration therefore, the maximum time difference between the closest label and the point cloud is 25 ms. With 130 km/h velocity, the movement during that time is 0.9 m, so the bounding box length is increased with this length in both forward and backward directions. The width increase which in this case is 0.2 m for both sides, compensates the inaccuracy of the objects orientation estimation. After the search regions are determined, the enclosed points are collected for each region. To ensure that no road surface points are collected, the points with \( z \) coordinate below the bottom boundary of the searching region plus 0.2 m are filtered out.

**TABLE 2.** Pseudo code for the label-frame assignment process.

| READ label(\( q \)) from label set |
|---|
| CALCULATE bottom center point of bounding box from its corners |
| CALCULATE \( a' \) |
| DETERMINE \( q_r \) |
| FOR each frame in FIFO: |
| APPEND the timestamp \( t'_{S_q} \corresponding to \( q_r \) index to \( T_z \) list |
| FIND nearest timestamp from \( T_z \) list \( t'_z(0,k) \) |
| USE Index \( m \) of assigned timestamp from \( T_z \) list to pair frame with the label |

**TABLE 3.** Rules for label fitting side decision.

| \( a_1 \) | \( a_2 \) | Long side | Short side |
|---|---|---|---|
| + | + | Left side | Back |
| - | + | Right side | Back |
| - | - | Right side | Front |
| - | + | Left side | Front |

To decide which part of the vehicle is represented by the collected points, the orientation information and the position of the bottom front right corner (\( C_3 \)) of the bounding box are used, as shown in Fig. 4. Let vector \( C_3 = (C_3x, C_3y, C_3z) \) represent the scanning laser beam. The scanned side of the target vehicle can be determined based of the object orientation wrt. the laser beam direction (\( C_5 \)) as follows:

First the orientation vector of the label (bounding box) is computed as \( s = C_5 - C_1 \) where \( C_1 \) is the bottom back right corner of the label. Then the unit vector (pointing to the same direction as \( C_3 \)) \( q \) and its normal \( r \) are determined:

\[
q = (q_1, q_2) = (C_3x/||C_3||, C_3y/||C_3||) \quad (4)
\]

\[
r = (-q_2, q_1) \quad (5)
\]

The \( s \) vector can be expressed: \( s = a_1 q + a_2 r \), where \( a \) represents the coordinates of \( s \) in the new orthogonal basis with basis vectors \( q \) and \( r \). In matrix form: \( s = Ba \). The coefficient vector \( a \) can be obtained as: \( a = B^{-1} s \), where \( B = [q \, r] \) represents the matrix of basis vectors. The signs of the coefficients \( a_1, a_2 \) show which side of the vehicle was scanned (visible by the LiDAR). This can be determined by Table 3.

The visible sides of the object determine which pair of perpendicular edges \( e_{long}, e_{short} \) of the bottom side of the corresponding bounding box will be used for fitting. However, it is necessary to determine which object points are going to be used as reference for fitting. Let us denote these points by \( P_{s \_ref} = (P_{s \_ref,1}, P_{s \_ref,2})^T \) and \( P_{l \_ref} = (P_{l \_ref,1}, P_{l \_ref,2})^T \) which represent the reference point of fitting the short and long sides of the bounding box respectively (see Fig. 5). The positions of the objects in the local coordinate system of the Trained Sensor and their orientations relative to the scanning.
laser beam must be considered because the position and relative direction together with a rule base (see later in this section) is used to determine $P_{s\_ref}$, $P_{l\_ref}$. The two axes of the Trained Sensors local coordinate system divides the $x, y$ plane into four quarters. On Fig. 4 the quarters are numbered with blue labels. The quarter index is determined based on which quarter $C_5$ is located in. A rule set has been set up to select $P_{s\_ref}$, and $P_{l\_ref}$ among the object points based on their coordinate values, with the evaluation of the quarter index and the signs of $a_1$, and $a_2$. The rules are described in Table 4. For example, in Fig. 4 $P_{s\_ref}$, and $P_{l\_ref}$ are selected for the object being in the outer lane as follows: The quarter index is 1, and the signs of $a_1$, and $a_2$ are $-$ and $+$ respectively. According to the corresponding rule, the object point which has the least $x$ coordinate value is selected for $P_{s\_ref}$, and the object point which has the least $y$ coordinate value is selected as $P_{l\_ref}$.

Let us denote by $p_1 = (p_{1x}, p_{1y})^T$, and $p_2 = (p_{2x}, p_{2y})^T$ the nodes of $e_{short}$ also $p_1$ is the common node of $e_{short}$ and $e_{long}$. $p_1, p_2$ determine normal vectors $n = (n_x, n_y)^T$, and $m = (m_x, m_y)^T$ of the bottom short edge $e_{short}$ and the bottom long edge $e_{long}$ of the bounding box. The normal vectors are computed with the following expressions:

$$m_x = p_{2x} - p_{1x} \quad n_x = m_y$$
$$m_y = p_{2y} - p_{1y} \quad n_y = -m_x$$

The algorithm then calculates the equations of the lines, $e_s$, $e_l$ which have the normal vectors $n, m$ therefore parallel with the side lines and pass through the corresponding fitting reference point. Finally the crossing point $P$ of lines $e_s$, and $e_l$ is calculated thus for its coordinates, we get:

$$p_s = (n_x c_1 - m_x c_2)/(n_x m_y - m_x n_y)$$
$$p_l = (c_s - n_x p_l)/n_x,$$

where $c_s, c_l$ are the offset coefficients of lines $e_s, e_l$ respectively. This calculated point represents the common node of $e_{short}$ and $e_{long}$ of the fitted label. Subtracting point $p_1$ from the calculated crossing point $p$ result the $\overrightarrow{p_1p}$ translation vector, with which the original label can be fitted to the object points. Fig. 5 shows an example for the above described process.

**B. CONCEPT OF TRAINING DATASET GENERATOR SYSTEM**

The proposed Automatic Label Injection method gives an opportunity for training dataset generation without the need of manual annotation. It can be used first of all in such measurement systems, which consists of multiple trainer sensors (covering a larger area) to provide fully automatic labeling for point clouds recorded by the infrastructure mounted LiDAR sensor. The trainer sensors can be temporarily deployed units, being mounted at similar height as the device used for the creation of the datasets the detector network was trained on. Thus, the detectors (of the trainer sensors) have appropriate detection performance on the recorded point clouds. The conceptual design is illustrated by Fig. 7. The theoretical concept of the system is described in the following paragraph.

Since the performance of the detector degrades as the object distance to the Trainer sensor increases, thus for each sensor a range limit can be defined inside which the detector performance is considered as acceptable for labeling. Let us call this area as Label Injection Region. In this concept, the label with higher confidence value gets assigned to an object which is currently within an overlapping Label Injection Region. With this configuration double labeling of an object can simply be avoided, otherwise an object level fusion solution might be considered. The point cloud acquired by the Trained Sensor is saved by the proposed dataset generator system if injected object labels from any Trainer Sensor have been assigned to it. The Label Injection Region of the Trained Sensors is determined based on the results in Section III-B. The aim is to define an area where the error rate of the detector is low enough, therefore the rarely happening false negative detections (absence of label) have negligible effect to the training process of the Trained Sensor. Based on the evaluation of the measurement results, the Label Injection range of the Trained Sensors was set to 35 m. The distance between each Trainer Sensor must be defined according to

| Quarter Index | $a_1$ | $a_2$ | Selection Rule for Short Side | Selection Rule for Long Side |
|---------------|-------|-------|------------------------------|-----------------------------|
| 1             | +     | +     | min $y$                      | max $y$                     |
| 2             | +     | -     | min $x$                      | min $y$                     |
| 3             | -     | -     | min $x$                      | max $y$                     |
| 4             | +     | +     | max $y$                      | min $y$                     |
| 5             | +     | -     | max $x$                      | min $y$                     |

**TABLE 4. Rules for fitting point selection.**

![FIGURE 5. Side fitting example for one of the two objects visible in the point cloud (left) and the transformed label, the extended search area and the fitted labels represented with white, blue and green color (right).](image-url)
the Label Injection range and the current traffic environment. For example, in a highway section, all traffic lanes must be covered with at least one Label Injection Region in order to avoid blind zones. Obviously, for a two-by-two lane arrangement (when each carriageway consists of two traffic lanes), the gap between the Trainer sensors can be larger than in case of a three-by-three lane arrangement (where there are 3 allocated lanes for each traffic direction). For the highway section which is displayed in Fig. 7, with a lane width of 3.75 m and with 35 m Label Injection range, the optimal distance between Trainer Stations is 67 m. Although, there are some constrains regarding the proposed system. First, it requires a traffic environment where the deployment of Trainer Sensors is supported, i.e. there is enough space for their temporal deployment, and the devices can be protected against theft or abuse. Secondly, sparse traffic is more favorable, because the chance of one object being shadowed by another is reduced. The third constraint is the necessity to include multiple LiDAR devices as trainer sensors (to fully cover the area perceived by the Trained Sensor), although they are costly nowadays, therefore high amount of financial effort is required for the installation of such a system.

C. INCREASING THE DETECTION RANGE OF THE TRAINED SENSOR

If high precision GNSS (Global Navigation Satellite System) data is available for all the objects inside the perception range of the Trained Sensor, the GNSS position, orientation, the GNSS reference point configuration and vehicle dimension data might be used as ground truth [20] and based on that, labels can be injected into the point cloud of the Trained Sensor. This ground truth can be used for two purposes: to validate the described Automatic Label Injection method and to generate the automatically labeled training set, similar to the one which would have been produced by the dataset generator system. This section describes the steps of increasing the object detection range of the detector (PointPillars neural network) operating on the data of a fix mounted infrastructure sensor. This goal was achieved by creating a training dataset based on available GNSS data. An infrastructure LiDAR sensor was deployed on a Highway section at ZalaZone Automotive Proving Ground. The detailed description about the measurement and the test environment is given in Section III-A. Multiple test vehicles were installed with high precision GNSS devices which logged the accurate position and orientation of the test vehicles with 10 ms sampling interval. The installation configuration (i.e. the position of the GNSS device inside the vehicle wrt. a reference point) of the GNSS device was given for all the vehicles as well as the dimensions of the test vehicles. With the help of such data, labels can be assigned to each object. The width, height and length properties for the label is determined by the vehicle dimensions. The position of the center point and the orientation of the label comes from...
the GNSS data. The GNSS-based label set (ground truth) is assigned to the point cloud being closest in time. Because of the time difference between two GNSS-based label sets was 10 ms, the maximum time shift between the assigned ground truth and the point cloud is 5 ms. To reduce this inaccuracy, the test vehicles performed the test scenarios with reduced speed, which meant 50 km/h and 100 km/h velocity. The position information from the GNSS device is logged in UTM (Universal Travel Mercator) global coordinate system. The deployed infrastructure sensor was calibrated wrt. this global reference system, which resulted the local to UTM transformation. By using these extrinsics, the labels given in the UTM system were transformed into the local coordinate system of the infrastructure sensors. Next, the labels were filtered by their position and the number of enclosed object points. The position filter was set to eliminate all the labels which were closer than 22 m or had a distance greater than 100 m. The lower threshold of the filter comes from the features of the LiDAR device and the mounting height, which creates a blind zone on the ground under 25 m from the origin of the LiDAR. The test vehicles had an average height of 1.5 m, thus, point cloud points belonging to vehicles can be observed starting from 22 m distance. The upper limit on the other hand ensures that the labels encapsulate at least one object point. In addition, one the constraints of the training algorithm of the PointPillars detector is that the number of encapsulated object points should be at least five in order for the label to be taken into account during training. Based on the statistical evaluation of the injected labels in section III-C, gaussian white noise \( \mathcal{N}(\mu_0, \Sigma) \) was added to the ground truth labels. This noise models the uncertainties in the position and heading in the overall label injection procedure. A point cloud frame combined with the corresponding label represents a frame in the training set. A training set was also created for benchmark, based on the measurement records. In this case, no noise has been added to the ground truth during the label generation phase.

During the test measurement, multiple test runs were performed. The ground truth data have been paired with the corresponding point cloud frames of the Trained station from all of the test runs. A single test run was selected and reserved for evaluation of the trained detector. The frames from this selected test run obviously were not used during the generation of the training set. After the frame collection was done the obtained dataset (including the point clouds and the labels) was converted into KITTI data format. In the KITTI data format, each point in the LiDAR point cloud is represented by \( x, y, z \) coordinates and an intensity value. The values are encoded in float32 data type. To ensure compatibility with the KITTI label format, the object class name, the position, the yaw angle and the dimension of the bounding box had to be computed for each label. The object class name for all labels is set to “car”. The 3-D dimension information stands for the height, with and length of the object in meters. The location gives the 3-D position of the object in meters. The Rotation_y means the heading of the object in range between \(-\pi\) and \(\pi\). During the KITTI format transformation, the frames of the dataset are shuffled. This step breaks the consecutive frame series with small position difference of objects, ensuring better generalisation capability for the trained network.

From the 14 recorded test runs, 13 were used to make the training set. The data set has 1734 frames and over 2400 car labels, which is considered as a smaller training set. From this set, 1213 frames were used for the training and the rest was reserved for evaluation. The detection range of the network was set from 0 m to 110 m in longitudinal direction and from \(-39.68\) m to \(39.68\) m in lateral direction. The height range was set between \(-3\) and \(-7\) meters, because the mounting height of the sensor was approximately 6m above the road surface. The size of the voxels is set to 0.16 m in both longitudinal and lateral directions and their height is set to 4 m. The training has run through 160 epochs, overall 296950 steps and reached a classification loss of 0.0803, and localisation loss of 0.144.

The detector network was also trained on the GNSS ground truth-based training set. The procedure and the configuration was the same as previously, which means 160 epochs and 296950 steps. At the end of the training, the network reached a classification loss of 0.0555, and localisation loss of 0.12.

The network which were trained with the ground truth and the network which were trained on the ground truth with noise added were compared to the network which was trained on the KITTI dataset. The results are presented in Section III-D.

III. RESULTS

A. MEASUREMENT SETUP

The test measurement for the proposed Automatic Label Injection method was performed at the Motorway module of ZalaZONE Automotive Proving Ground. This module is a real 1.5 km long highway section with an overpass. The bridge provides the possibility to set up the Trained Sensor above the road surface in any chosen point in lateral direction.
FIGURE 9. Layout of the calibration points and the location of the stations.

The trainer station consisted of an Ouster OS2 high range LiDAR unit with 128 channels, a Flir BlackFly 2 MP camera with GigE output, and a Cohda MK5 unit. The camera was used to give visual feedback for the measurements but the captured data was not utilised for the Automatic Label Injection method. The Cohda MK5 unit provided the time synchronisation service. The OS2 unit was mounted at vehicle rooftop level above the ground. The station of the Trainer Sensor was deployed at 50 m distance from the overpass (see Fig. 9).

The Trained Sensor was an Ouster OS2 high range LiDAR unit. In addition to this LiDAR, the station was equipped with a Cohda MK5 unit. The sensors were mounted on the overpass element of the Motorway module. For testing the proposed Automatic Label Injection method, the output data of the OS2 unit and the time synchronisation service of the Cohda MK5 device was used from this sensor station. Fig. 11 shows the station of the Trainer Sensor (left) and the station which holds the Trained Sensor (right) and the location of the two stations can be seen in Fig. 9.

In order to perform the proposed label injection method, the transformation between the two local coordinate systems had to be determined. The local coordinate system of the LiDAR sensors is defined according to Fig. 8. This was performed by calibration of the sensors.

The UTM (Universal Transverse Mercator) system [22] was selected as the reference global coordinate system, the extrinsics of the two sensors have been estimated wrt. this global reference. The transformation from the local coordinate system of the Trainer sensor (\(TR_{local}\)) to UTM system and the transformation from the local coordinate system of the Trained sensor (\(TD_{local}\)) to UTM provided the transformation form \(TR_{local}\) to \(TD_{local}\). The following paragraphs will describe how the calibration was performed.

First, several easily distinguishable points were marked in the environment as calibration points. Then the precise GPS coordinates of each point were measured with a high precision GNSS surveyor. Fig. 9 shows the layout of the points. To determine the location of each calibration point, an indicator box (of size 1 x 1 m) was used to detect the calibration points in LiDAR point clouds. The box was placed at the location of calibration points in such a way that one of its corners indicated the exact location of the given calibration point. At each location, the corresponding point cloud (containing also the indicator box) was recorded. Then the local coordinates of the calibration points were extracted from the point cloud in case of both sensors.

To enable the transformation from the local coordinate system of a given sensor to the global UTM system, a six degrees of freedom problem must be solved. This means, that at least 3 non-collinear points are necessary to estimate transformation. However, due to the measurement noise, thirteen calibration points were marked and surveyed. This generates an over-determined system of linear equations. The two representations (local coordinate system of the sensor and UTM) of the calibration points are normalised to have zero mean and unit variance. To determine the rotation, a system of linear equations must be solved resulting a 3 x 3 matrix \(Q\), which is not yet a rotation matrix. In order to get the closest rotation matrix \(R\) to \(Q\) the following problem has to be solved:

\[
\min_{R} ||R - Q||_F^2 \quad \text{subject to} \quad R^T R = I \quad (6)
\]

By taking the singular value decomposition of \(Q\) i.e. \(Q = USV^T\). \(R\) can be obtained as \(R = UV^T\) [23]. With \(R\) and the center points of the point clouds: \( centroid_A \), and \( centroid_B \), the translation can be computed with the expression below.

\[
\text{translation} = -R \times centroid_A + centroid_B \quad (7)
\]
Because of the measurement noise during the GNSS survey and the local position measurement, the obtained rotation matrix and the translation vector are estimations of the real transformation, resulting in a rough calibration. Let us refer to the calibration of Lidar→UTM extrinsics as UTM calibration. To obtain a more precise calibration, further refinement is needed. Since the HD map from the Motorway section is available, it can be used to refine the calibration as follows: the HD map and the LiDAR point cloud are registered by the ICP (Iterative Closest Point) method [24]. This method yielded acceptable results for our proposed Automatic Label Injection method. For each point in the source point cloud the closest point from the reference point cloud is assigned. Based on such assignment the transformations estimated and applied on the source point cloud. These steps are repeated until the two point clouds get aligned (i.e. the defined cost is minimised). The whole calibration process can be followed in Fig. 10. This process was performed for both the Trained and the Trainer Sensors.

Let us denote the obtained sensor to UTM transformations as follows: $T_{TD\rightarrow UTM}$ and $T_{TR\rightarrow UTM}$. With the help of these transformation matrices the labels can be transformed from $TR_{local}$ to $TD_{local}$. The transformation is calculated with the following expression, where $T_{orig}$ and $C$ represent the transformation of the Trainer Sensors original point cloud (acquired by the Trainer sensor) to such a pose which is
optimal for the detector network and a corner point of a detected label, respectively:

\[ C' = T_{\text{TD} \rightarrow \text{UTM}}^{-1} T_{\text{TR} \rightarrow \text{UTM}} T_{\text{orig}}^{-1} C T \]  

(8)

Each data recorder framework at the trainer and the trained stations had its own system clock which have to be synchronised in order to collect data with synchronised timestamps. The reference clock for the synchronisation was the high precision clock of the GPS system. In every second, a precise clock signal is transmitted by the GPS satellites. The Cohda MK5 devices used at the stations are capable to receive those transmissions and adjust their own system clock accordingly. The clock of the data recording system is kept synchronised with the clock of the corresponding Cohda MK5 device by using the Network Time Protocol [25].

B. TRAINER DETECTOR PERFORMANCE

The performance of the detector processing the data of the Trainer Sensor was evaluated by relying on the IOU metric. For a given frame, the predictions are paired with the ground truth labels based on the center points of the bounding boxes. Then the IOU score for a ground truth-prediction pair is calculated. The IOU scores are calculated as follows:

\[ \text{OverallIOUScore} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m_i} \text{IOU}}{n} \]  

(9)

where \( m_i, i = 1..n \) and \( n \) stand for the number of objects in the \( i \)th frame and the number of frames, respectively. Furthermore, it is obvious that \( 0 \leq \text{OverallIOUScore} \leq 1 \). If there is no detection available for a given object, the corresponding IOU is set to zero. Furthermore, the mean and standard deviation of the distances between the center points of the bounding boxes (detected and the corresponding ground truth) were calculated. In this case, the detector of the Trainer Station was evaluated on a recording which was used during the training dataset (for the detector corresponding to the Trained Station) generation process. To determine the label injection range of the Trainer Sensor, the OverallIOUScore for different distance intervals were computed, see Table 5. Based on the measurement results, the label injection range for the Trainer Sensor is set at 35 m due to the decreasing tendency of the IOU score values after the 35-0 m range, shown in Fig. 12.

C. LABEL VERIFICATION

The injected labels were evaluated just as the original labels yielded by the Trainer sensor. The evaluation sequence was the same as in III-B. The evaluation of the injected labels gives information about the overall performance of the Automatic Label Injection method, especially when the IOU results are compared with the IOU results of the detections of the Training Sensor. The difference between the two results (see Table 6) reflects the effectiveness of the proposed label injection approach (described in II-A) which prevents significant accuracy losses caused by the time difference between the scans of the two sensors, and the measurement system calibration error.

Let \( \mu_{\text{INJ}} \) and \( \sigma_{\text{INJ}} \) denote the mean and standard deviation of the distance between the center points of the bounding boxes of the ground truth and the bounding boxes of the injected labels. Let \( \mu_{\text{TR}} \) and \( \sigma_{\text{TR}} \) denote the mean and standard deviation of the distance between the center points of the bounding boxes of the ground truth and the labels provided by the detector of the Training Station. The OverallIOUScore for the injected labels and for the labels provided by the detector of the Training Station, together with \( \mu_{\text{TR}}, \mu_{\text{INJ}}, \sigma_{\text{TR}} \) and \( \sigma_{\text{INJ}} \) can be followed in Table 6.

The center point distance between the injected labels and the ground truth has been evaluated along each axis. Furthermore the absolute error of the heading was measured as well (see Table 7).

D. COMPARISON OF THE ORIGINAL AND THE RETRAINED DETECTORS

One of the test runs was not involved during the training dataset generation procedure. This recording serves as reference to evaluate the performance of three detectors: the

---

### TABLE 5. Distance intervals and the corresponding overall IOU scores.

| Distance interval [m] | IOU score | Distance interval [m] | IOU score |
|-----------------------|-----------|-----------------------|-----------|
| 10-0                  | 0.6260    | 45-0                  | 0.6486    |
| 15-0                  | 0.6517    | 50-0                  | 0.6011    |
| 20-0                  | 0.6473    | 55-0                  | 0.5876    |
| 25-0                  | 0.6691    | 60-0                  | 0.5458    |
| 30-0                  | 0.6689    | 65-0                  | 0.4989    |
| 35-0                  | 0.6722    | 70-0                  | 0.4725    |
| 40-0                  | 0.6530    | 75-0                  | 0.4323    |

### FIGURE 12. IOU scores of the Trainer Sensors detector in relevance of various distance intervals.

### TABLE 6. Injected labels and Trainer Sensors labels evaluation results: overall IOU score, center points distance mean and standard deviation values.

|                     | Injected labels | Mean value [m] | Std. deviation |
|---------------------|----------------|----------------|----------------|
| Injected labels     | 0.6117         | 0.3196         | 0.1623         |
| Trainer Sensors Labels | 0.6722        | 0.3274         | 0.0751         |
detector which was trained on the KITTI open dataset, the detector which was trained on a generated training dataset based on the ground truth and the detector which was trained on a generated dataset corrupted by Gaussian white noise with mean and std. deviation given by Table 7 (this noise was added to the ground truth labels). The test run realises a scenario where two cars are approaching the sensor stations beside each other and a third one follows them in the outer lane (in an L-shaped formation). Here, the ability of the detectors to recognise objects from different distances was tested. The maximum detection range was determined in case of the mentioned three detector variants. Table 8 contains the measurement results.

In case of the original detector, the OverallIOUScore was calculated for ranges 22–100 m and 22–50 m. The narrower range falls within the maximum detection range of the original detector. The overall IOU score for the 22–100 m range enables the comparison between the detection performance of the three different detector variants. The results obtained for the range 22–50 m show how the predicted labels overlap with the ground truth labels inside the detection range (see Table 8) of the original detector. For the detectors which were trained on the generated training datasets, evaluation in the 22–50 m range is not necessary because the 22–100 m range falls into their object detection range.

The OverallIOUScore for the detectors are contained by Table 9.

The third comparison method was the evaluation of the distances between the center points of the ground truth and the predicted boxes. This metric considers only those frames where the ground truth and the corresponding predicted label are present for an object. The mean distance and standard deviation are calculated (see Table 10).

TABLE 7. The mean and standard deviation of the distance between the center points of the injected labels and the center points of the ground truth labels.

| Direction | Mean value [m] | Std. deviation |
|-----------|---------------|---------------|
| x direction | 0.2363 | 0.1807 |
| y direction | 0.1311 | 0.0975 |
| z direction | 0.3274 | 0.0751 |
| heading | 0.1133 | 0.0683 |

TABLE 8. Maximum detection ranges for the compared detector nets.

| Detector variant | Maximum detection range [m] |
|------------------|-----------------------------|
| Trained on KITTI | 53.22                       |
| Trained on GNSS based training set | 104.87 |
| Trained on noise added training set | 104.87 |

TABLE 9. OverallIOUScore of the original and custom trained detectors.

| Detector variant | IOU at 22-50m | IOU at 22-100m |
|------------------|---------------|---------------|
| Trained on KITTI | 0.222         | 0.106         |
| Trained on GNSS based training set | -     | 0.442         |
| Trained on noise added training set | -     | 0.439         |

TABLE 10. Mean value and standard deviation of distance between ground truth and prediction.

| Detector variant | Mean value [m] | Std. deviation |
|------------------|---------------|---------------|
| Trained on KITTI | 0.3287        | 0.0929        |
| Trained on GNSS based training set | 0.2059 | 0.1364 |
| Trained on noise added training set | 0.2538 | 0.1079 |

TABLE 11. The center point mean distance and standard deviation along each axis and the mean and difference and standard deviation for the heading for the three detector variants.

| Detector variant | Mean value [m] | Std. deviation |
|------------------|---------------|---------------|
| Trained on KITTI | 0.2249        | 0.1264        |
| Trained on GNSS based training set | 0.1577 | 0.0756 |
| Trained on noise added training set | 3.954 | 3.139 |

The distance between the center points of the predicted labels and the ground truth were evaluated along each axis. Furthermore the heading difference was also examined. The details of evaluation results for the three detector variants are reported in Table 11.

IV. DISCUSSION

A. LABEL VERIFICATION

The results in Table 6 suggest that the labels yielded by the Trainer sensor suffered only a slight accuracy loss during the label injection process. However, $\sigma_{\text{INJ}}$ (see III-C) is increased compared to $\sigma_{\text{TR}}$, showing that the distance between the center points of the injected label and the corresponding ground truth label may be larger than in case of the labels coming from the Trainer Station. The reason for this is the rule set of the label fitting process. There are several boundary situations, where the orientation vector is close to a segment separation line. In those cases, the label fitting algorithm can adjust the label into a non-optimal position. Fig. 14 shows an optimal and a non-optimal solution for the label fitting process.

B. COMPARISON OF THE ORIGINAL AND THE RETRAINED DETECTORS

As for the comparison of the original and the retrained detectors (see III-D), the results in Table 8 show that the maximum detection range has been increased by 97% with the use of the ground truth based dataset. This range remained the same when noise was added to the training labels which the detector was trained on.
C. LIMITATIONS
In order to perform a tailored training dataset generation with the proposed system, numerous temporarily deployed trainer sensors are required. These sensors are currently being expensive therefore, large amount of funds is necessary. Furthermore, the installation area has to be safe enough to prevent theft or intentional damage in the equipment.

The method operates on pre-collected data in offline mode. The quality of the generated labels significantly depends on the performance of the detector network which processes the point clouds of the Trainer sensor.

V. CONCLUSION
In this article a novel Automatic Label Injection method has been proposed, which automatically generates the labels for objects in the point cloud acquired by the Trainer Sensor relying on the labels provided by a well performing detector operating on the point cloud of the Trainer Sensor. The proposed technique enables the creation of a training dataset which is tailored for a particular high mounted infrastructure LiDAR unit. The creation process of the dataset does not require manual labour for object labeling. The proposed method was tested under real conditions at a motorway section where a single Trainer and a single Trained station was deployed. The performance of the proposed Automatic Label Injection technique was evaluated on the data recorded at the test site. The case of multiple trainer stations covering a larger area was also considered and its concept was elaborated.

It is shown in the article, that the detector neural networks can be retrained on the created dataset to enhance their detection performance on point clouds which are provided by fix mounted infrastructure LiDAR devices at elevated position. Further development possibility for the proposed method is to change the trainer Sensors from temporarily deployed units to fix mounted sensors assigned with already optimised detectors, which includes not just LiDAR devices but the fusion of different sensor types, as well. Furthermore the implementation of object level fusion which increases the robustness of the labeling is a possible way of improvement.

The preliminary calibration process which provides the transformation between the local and UTM coordinate systems is performed manually. In further development stages, this process can be automated to reduce the necessary manual labour. Also, refinement of the label fitting rule set for eliminating non-optimal bounding box adjustments can be considered as further development direction.

REFERENCES
[1] D. Fernandes, A. Silva, R. Névoa, C. Simões, D. Gonzalez, M. Guevara, P. Novais, J. Monteiro, and P. Melo-Pinto, “Point-cloud based 3D object detection and classification methods for self-driving applications: A survey and taxonomy,” Inf. Fusion, vol. 68, pp. 161–191, Apr. 2021.
[2] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2012, pp. 3354–3361.
[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. Erin Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “NuScenes: A multi-modal dataset for autonomous driving,” 2019, arXiv:1903.11027.
ZOLT VINCZE was born in Budapest, Hungary, in 1986. He received the B.S. degree in electrical engineering from Obuda University, Budapest, in 2012, and the M.S. degree in electrical engineering from the Budapest University of Technology and Economics, Budapest, in 2015, where he is currently pursuing the Ph.D. degree in transportation and vehicle engineering.

From 2015 to 2020, he worked as an Electrical Engineer at Geoelectro Ltd., Nagykárody, Hungary. Since 2020, he has been working as a Research Assistant at the Department of Automotive Technologies, Budapest University of Technology and Economics. His research interest includes the development of intelligent infrastructure system for aiding autonomous traffic, which involves the development of new neural networks based object detectors, also researching new solutions which lessen the required amount of manual labor for supervised learning of detector networks.

ANDRÁS RÖVID was born in Rimaszombat, Slovakia, in 1978. He received the degree in computer engineering from the Faculty of Electrical Engineering and Informatics, Technical University of Kosice, Kosice, Slovakia, in 2001, and the Ph.D. degree in transportation sciences from the Budapest University of Technology and Economics (BUTE), Budapest, Hungary, in 2005. Since 2019, he has been the Leader of the Perception Group, Department of Automotive Technologies, BUTE, where he is currently a Senior Research Fellow. He has been the author or coauthor of over 100 publications. His main research interests include image processing, 3D machine vision, and sensor fusion.

VIKTOR TIHANYI was born in Hungary, in April 1981. He received the degree in electric engineering from the Faculty of Electric Machines and Drives, Budapest University of Technology and Economics, in 2005, the Ph.D. degree, in 2012, and the B.Sc. degree in mechanical engineering from the Vehicle Technology Faculty, University of Obuda, in 2014. He has been working at Hyundai Technology Center Hungary, for since years, since 2008. In 2013, he changed to the Automotive Sector, Knorr-Bremse Fekrendozó Kft., as a Project Leader and as a Team Leader of research and innovation activities. Besides his industrial employment, he has been also working at the Department of Automotive Technologies, Budapest University of Technology and Economics, as a Research Leader of autonomous vehicle related research projects, since 2016, and as an Associate Professor.

[4] P. Sun, “Scalability in perception for autonomous driving: Waymo open dataset,” 2019, arXiv:1912.04838.
[5] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, “One thousand and one hours: Self-driving motion prediction dataset,” 2020, arXiv:2006.14480.
[6] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, “Argoverse: 3D tracking and forecasting with rich maps,” 2019, arXiv:1911.02620.
[7] (2019). Pandaset: Public Large-Scale Dataset for Autonomous Driving, Scale AI. [Online]. Available: https://scale.com/open-datasets/pandaset
[8] M. Pitropov, D. E. Garcia, J. Rebbel, M. Smart, C. Wang, K. Czarnecki, and S. Waslander, “Canadian adverse driving conditions dataset,” Int. J. Robot. Res., vol. 40, nos. 4–5, pp. 681–690, Dec. 2020, doi: 10.1177/0278364920979368.
[9] G. Pandey, J. R. McBride, and R. M. Eustice, “Ford campus vision and lidar data set,” Int. J. Robot. Res., vol. 30, no. 13, pp. 1543–1552, 2011, doi: 10.1177/0278364911400640.
[10] M. De Deuge, A. Quadros, and C. Hung, “Unsupervised feature learning for classification of outdoor 3D scans,” in Proc. Australas. Conf. Robot. Automat. (ACRA), 2013, pp. 1–9.
[11] A. Teichman, J. Levinson, and S. Thrun, “Towards 3D object recognition via classification of arbitrary object tracks,” in Proc. IEEE Int. Conf. Robot. Autom., May 2011, pp. 4034–4041.
[12] B. Gong and T. Funkhouser, “Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2021, pp. 15363–15373.
[13] C. Creß, W. Zimmer, L. Strand, V. Lakshminarasimhan, M. Fortkord, S. Dai, and A. Knoll, “A9-dataset: Multi-sensor infrastructure-based dataset for mobility research,” 2022, arXiv:2204.06527.
[14] F. Wang, Y. Zhuang, H. Gu, and H. Hu, “Automatic generation of synthetic LiDAR point clouds for 3-D data analysis,” IEEE Trans. Instrum. Meas., vol. 68, no. 7, pp. 2671–2673, Jul. 2019, doi: 10.1109/TIM.2019.2906416.
[15] X. Chen, B. Mersch, L. Nunes, R. Marcuzzi, I. Vizzio, J. Behley, and C. Stachniss, “Automatic labeling to generate training data for online LiDAR-based moving object segmentation,” IEEE Robot. Autom. Lett., vol. 7, no. 3, pp. 6107–6114, Jul. 2022.
[16] Y. You, K. Luo, C. P. Phoo, W. L. Chao, W. Sun, B. Harirhan, M. Campbell, and K. Q. Weinberger, “Learning to detect mobile objects from LiDAR scans without labels,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2022, pp. 1130–1140.
[17] G. Wang, J. Wu, T. Xu, and B. Tian, “3D vehicle detection with RSU LiDAR for autonomous mine,” IEEE Trans. Veh. Technol., vol. 70, no. 1, pp. 344–355, Jan. 2021.
[18] M. Obaid and Z. Szalay, “A novel model representation framework for cooperative intelligent transport systems,” Periodica Polytechnica Transp. Eng., vol. 48, no. 1, pp. 39–44, May 2019, doi: 10.3311/PPhr.13759.
[19] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “PointPillars: Fast encoders for object detection from point clouds,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 12697–12705.
[20] V. Tihaný, T. Tettamanti, M. Csonthó, and A. Eichberger, “Motorway measurement campaign to support R&D activities in the field of automated driving technologies,” Sensors, vol. 21, no. 6, p. 2169, 2021.
[21] Sensor Data—Ouster Sensor Docs documentation. Accessed: Aug. 10, 2022. [Online]. Available: https://static.ouster.dev/sensor-docs/image_route1/image_route2/sensor_data/sensor_data.html#sensor-coordinate-frame
[22] R. B. Langley, “The utm grid system,” GPS World, vol. 9, no. 2, pp. 46–50, 1998.
[23] Z. Zhang, “A flexible new technique for camera calibration,” Microsoft Res., Redmond, WA, USA, Tech. Rep. MSRTR-98-71, Dec. 1998. [Online]. Available: http://research.microsoft.com/~zhang/Calib/
[24] S. Bouazizi, A. Tagliasacchi, and M. Pauly, “Sparse iterative closest point,” in Computer Graphics Forum, vol. 32, no. 5, Oxford, U.K.: Blackwell, 2013, pp. 113–123.
[25] D. L. Mills, “Internet time synchronization: The network time protocol,” IEEE Trans. Commun., vol. 39, no. 10, pp. 1482–1493, Oct. 1991, doi: 10.1109/26.103043.