IEEE Open Journal of
Intelligent Transportation
Systems

Received 3 March 2023; revised 3 June 2023; accepted 6 July 2023. Date of publication 11 July 2023; date of current version 16 August 2023.

Digital Object Identifier 10.1109/OJITS.2023.3293911

DNN-Based Map Deviation Detection in LiDAR Point Clouds

CHRISTOPHER PLACHETKA[1], BENJAMIN SERTOLLI[1], JENNY FRICKE[1], MARVIN KLINGNER[2] (Graduate Student Member, IEEE), AND TIM FINGSCHEIDT[2] (Senior Member, IEEE)

[1]Self-Driving System Development, Volkswagen Group, 38440 Wolfsburg, Germany
[2]Institute for Communications Technology, Technische Universität Braunschweig, 38106 Braunschweig, Germany

CORRESPONDING AUTHOR: C. PLACHETKA (e-mail: christopher.plachetka@volkswagen.de)

ABSTRACT In this work we present a novel deep learning-based approach to detect and specify map deviations in erroneous or outdated high-definition (HD) maps using both sensor and map data as input to a deep neural network (DNN). We first present our proposed reference method for map deviation detection (MDD) utilizing a sensor-only DNN detecting traffic signs, traffic lights, and pole-like objects in LiDAR data, with deviations obtained by subsequently comparing detected objects and examined map. Second, we facilitate the object detection task by using the examined map as additional input to the network. Third, we employ a specialized MDD network to directly infer the correctness of the map input. Finally, we demonstrate the robustness of our approach for challenging scenes featuring occlusions and a reduced point density, e.g., due to heavy rain. Our code is available at https://github.com/Volkswagen/3dhd_devkit.

INDEX TERMS 3D object detection, automated driving, convolutional neural network (CNN), deep neural network (DNN), deviation detection, high-definition (HD) map, LiDAR, map verification, map validation.

I. INTRODUCTION

DUE to shortcomings of today's perception and environment modeling algorithms [1], the driving function of automated vehicles relies on prior knowledge regarding the stationary environment in the form of high-definition (HD) maps. Such driving function is prone to failure as map data can deviate from the real world. In our previous work [2], we proposed a system framework to achieve dependable maps to address the issue of such map deviations. Such dependability requires a system that detects, specifies, and corrects map deviations to obtain maps that are safe to use, reliable, and available. Specifically, the safe use of map data refers to the detection of deviations ahead of the vehicle, while reliable maps feature short update cycles. Available maps are obtained by correcting deviations on the fly within the vehicle, e.g., in construction sites. This article proposes a map deviation detection (MDD) method of such a system.

The review of this article was arranged by Associate Editor Abel C. H. Chen.

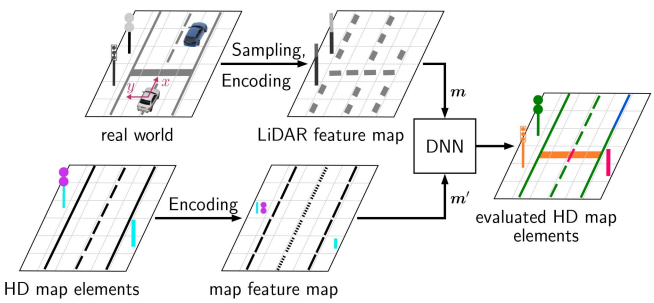


FIGURE 1. General concept of our deep learning-based map deviation detection approach. Top: The real world (grey: LiDAR reflectivity) is sampled and encoded into a first feature map m. Bottom: HD map elements (e.g., poles, signs, and markings) are discretized and encoded into a second feature map m'. Right: A deep neural network (DNN) compares feature maps to verify (green), falsify (red), or detect missing map elements (orange), if not unknown due to occlusion (blue). White car: ego vehicle and its reference frame.

Existing MDD or map verification approaches (e.g., [3], [4]) are not able to detect and specify map deviations on the level of individual HD map elements and, thus, are not suitable for achieving map dependability. Moreover, algorithms for detecting HD map elements are

known (e.g., for road markings [5], traffic lights [6], or lanes [7]) but underperform in bad weather conditions [8] or in the presence of partial occlusions [7]. Therefore, a deviation detection method is needed that simplifies the detection task wherever possible by leveraging the map as source of prior hypotheses regarding possible element locations and features. In this work, we present a novel approach to MDD based on a deep neural network (DNN) that takes both map and sensor data as input (see Fig. 1 for illustration). With this additional map input, fewer measurements are potentially required to either verify or falsify the correctness of an existing hypothesis, as respective measurements can be compared to an expected measurement distribution for a specific type of map element. Using a DNN, respective distributions and the comparison method can be learned, whereas hand-crafting yields unsatisfying results in a related attempt [9], as expected distributions have been proven difficult to be derived manually. Thereby, contextual knowledge from the scene can be incorporated to facilitate the deviation detection task, e.g., locations for pole-like objects (subsequently referred to as poles) increase the likelihood for traffic signs.

As proof of concept, we train and evaluate our DNN using the 3DHD CityScenes dataset [10] as the only publicly available dataset providing a holistic set of HD map elements, i.e., comprising signs, lights, and poles. The dataset features high-density LiDAR point clouds that were used to annotate the corresponding HD map, yielding accurate labels both in terms of completeness and spatial alignment between sensor and map data. Such a high-quality dataset has several key advantages compared to data obtained from onboard sensors. First, nearly all deviations are known, apart from those due to annotation mistakes, as we induce artificial deviations by changing both map and LiDAR data intentionally. Also, no other dataset provides annotated map deviations. Second, the usage of high-density point clouds without occlusions allows for respective ablation studies, where both a reduced point density (simulating onboard scans or bad weather conditions) and occlusions can be exactly controlled. Third, the highly precise spatial data alignment allows for the induction of artificial misalignment errors in a controlled fashion. Our holistic approach extends to various types of HD map elements and can be combined with different DNN architectures for 3D object detection. In this work, we apply `3DHDNet` [10] being designed for predicting vertically stacked elements such as signs. As 3DHD CityScenes does not provide camera images, we only employ LiDAR point clouds as method input. However, also multimodal architectures additionally fusing camera images can be employed (e.g., [11], [12], [13]), given a respective dataset.

Our contributions are fourfold. First, we present our reference method for MDD utilizing a sensor-only DNN for object detection, with deviations being obtained by comparing detected objects with the examined map. It comprises a multitask extension of our earlier single-task `3DHDNet` architecture [10], now capable of predicting traffic signs, traffic lights, and poles simultaneously. Second, we demonstrate a way of facilitating the object detection task by providing the DNN with the encoded, examined map as additional input. Third, we present our specialized MDD network that directly evaluates the correctness of individual map elements as the full expression of our concept shown in Fig. 1. Last, we perform ablation studies simulating bad weather and low-density onboard scans by reducing point density, and simulating partial occlusions of objects, showing the superior performance of our specialized MDD network. Our deviation annotations published along with our code allow for benchmarks in the field of MDD, while our open-source method may serve as a baseline for future research.

II. RELATED WORK

In this section, we review detection methods for HD map elements in both geodesy and in the automotive domain regarding traffic signs, traffic lights, and poles, followed by a review of existing methods for map deviation detection. We conclude with a brief review regarding DNN-based object detection in LiDAR point clouds.

A. MAP ELEMENT DETECTION

The generation of HD maps based on detecting map elements in high-density point clouds is a frequently researched topic in geodesy [14], [15], [16], while in the automotive field, such maps are typically only used while driving [17], [18]. Compared to subsequently presented approaches, we provide a holistic, DNN-based method that extends to various types of map elements and omits the need for hand-crafted, type-specific algorithms. Our novel multitask `3DHDNet` integrates our previous work on DNN-based pole and sign detection [10], [19], and is extended in this work to also detect traffic lights.

1) GEODESY

In geodesy, methods for detecting *traffic signs* commonly feature two stages [20], [21], [22]: First, three-dimensional (3D) traffic sign locations are detected in the point cloud. Second, detections are projected into the 2D camera image for further classification. The LiDAR-based detection stage generally employs three steps: ground point removal [23], [24], [25], segmentation using various clustering algorithms [21], [22], [23], [24], [25], [26], and plane fitting [21], [22], [25], [26]. The classification stage typically utilizes machine learning-based algorithms [21], [22], [27], [28], [29]. Regarding *poles*, detection methods can be grouped into shape-based [30], [31], [32], [33], feature-based [33], [34], [35], [36], [37], and machine learning-based [37], [38], [39], [40] approaches. In particular, shape-based methods rely on prior assumptions regarding the shape of the objects to be detected, e.g., fitting cylindrical [30] or circular [31] shapes to the voxelized point cloud. Feature-based approaches first extract geometrical [34], intensity [34], or density [35] features from the point cloud to provide the input for downstream detection or segmentation algorithms [33], [34],

[36], [37]. *Traffic lights* have been considered in the context of pole detection, i.e., as detected traffic light poles, without being represented as individual instances [33], [34], [37]. Further, traffic lights occur as class in various proposed semantic segmentation algorithms for point clouds relying on conventional algorithms [41], [42], [43] or DNNs [44].

2) AUTOMOTIVE FIELD

In the automotive field, *traffic sign* detection methods generally follow the two processing stages found in geodesy [45], [46], [47]. To address the sparsity of onboard scans, pseudo images are constructed from point clouds [47], [48]. Also, the point cloud is augmented with color features to facilitate the clustering process [45], [46], with RANSAC [45], [46], [48] or PCA [47] being utilized for plane fitting. For the classification stage, SVM-based [46], or template-matching approaches [47] have been proposed. *Poles*, on the other hand, are most commonly detected as part of a landmark-based self-localization using conventional algorithms and various sensor setups (e.g., [18], [49]). Regarding *traffic light* detection, proposed methods are image-based [50], relying on conventional algorithms [50], [51] or single-shot (2D) CNNs [6], [52]. Commonly, the detection is bypassed by projecting known traffic lights obtained from an HD map into the camera image to classify the traffic light state (i.e., red) [53], [54], [55], [56]. To this end, LiDAR data is only used to facilitate the localization of the ego vehicle on the map [55], [56].

B. MAP DEVIATION DETECTION (MDD)

Our conceptual system framework for dependable maps [2] assumes an MDD component capable of evaluating individual HD map elements, for which we propose a holistic and robust solution in this article. In comparison, early approaches to MDD examine ordinary navigation maps, which do not provide the level of detail (e.g., lane geometry) required for map-based driving [4], [57], [58]. More recently, MDD methods for lane markings have been proposed [55], [59], [60], [61], [62], [63], [64], whereby respective methods rely on prior object detection results as input. In contrast, our proposed solution features an additional map input to facilitate object detection in the first place, which allows for a robust map verification in challenging conditions. Only few works consider the additional map input for MDD [3], [65], [66]. Specifically, Hartmann et al. [65] consider the detection of lane geometry deviations using a DNN predicting the probability of the entire map being correct, whereas our method is able to evaluate map elements individually. Also predicting the correctness of the map as a whole, Lambert and Hays [66] fuse image and rendered map data within a DNN, whereby map deviations regarding lane geometry and crosswalks are simulated by modifying the map. They highlight the generalization of simulated deviations to those seen in the real world. In our work, we also simulate map deviations, but we modify both map and sensor data to create various deviation types.

Moreover, probabilistic methods for evaluating map correctness have been proposed [9], [67], [68]. Specifically, Raaijmakers [9] manually derived estimated sensor measurement distributions for roundabouts, which yielded unsatisfying results. Fabris et al. [67], [68] estimate map correctness using Bayesian networks assuming respective conditional probabilities, e.g., to model the influence of bad weather on the map correctness estimation. In contrast, we propose a method that learns expected measurement distributions without the need of prior assumptions.

C. DNN-BASED OBJECT DETECTION IN POINT CLOUDS

While early research for DNN-based object detection relied on hand-crafted encodings [69], [70], the paradigm has shifted towards learned feature encodings [71], [72], [73], reducing the loss of geometrical information. More recent research examines point-based networks [74], [75] that omit the discretization step and create predictions for each point. Our `3DHDNet` architecture draws from network topologies designed for road user detection [71], [72], [73], but specifically allows for the resolution of vertically stacked map elements such as signs.

III. DNN-BASED MAP DEVIATION DETECTION

In this section, we introduce our approach to DNN-based map deviation detection (MDD). To this end, we first provide definitions on maps and types of map deviations in Section III-A. Subsequently, we present an overview on the three MDD method variants examined in Section III-B. We then present in detail our reference method `MDD-SC` in Section III-C, serving as performance reference for later evaluations. Thereby, we present our core concepts on DNN-based map element and deviation detection, which are adopted or modified by the method variants `MDD-MC` and `MDD-M`, subsequently presented in Sections III-D and III-E, respectively. The variant `MDD-M` incorporates our specialized MDD network that directly infers the correctness of the map input, implementing the concept shown in Fig. 1.

A. DEFINITIONS ON MAPS AND MAP DEVIATIONS

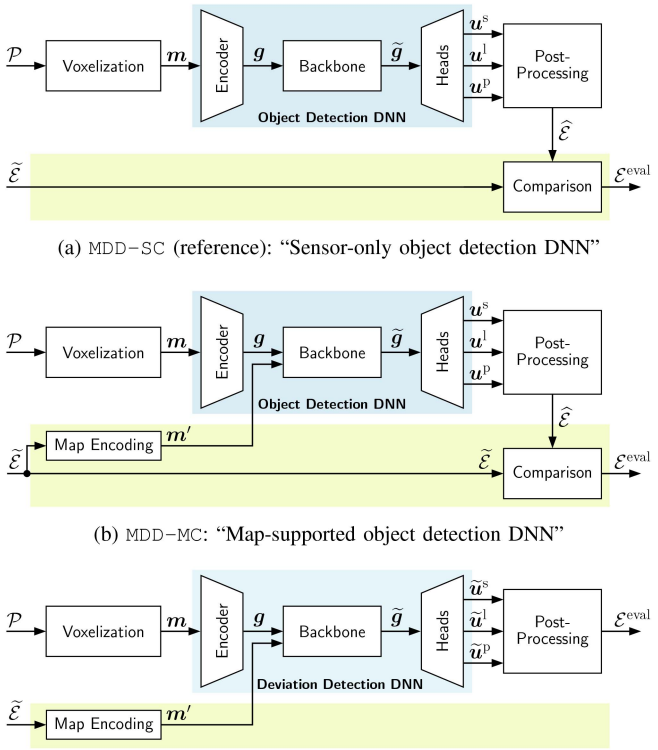
An HD map can be defined as a set $\mathcal{M} = \mathcal{E} \cup \mathcal{R}$, consisting of a set of *map elements* $\mathcal{E}$ (e.g., traffic signs or lane markings) and *relations* $\mathcal{R}$ between elements (*association*, *composition*, or *link* relation) [2]. Map elements can be categorized into physical (real-world objects) and semantical elements (mental models, e.g., lanes or roundabouts), and can be differentiated by a *type* (major class, e.g., `sign`, `light`, or `pole`) and a set of (*mandatory* or *characteristic*) *attributes* (e.g., `subclass` or `orientation`). In general, to provide a complete definition for map deviations, deviating map items comprise both elements and relations: $\mathcal{F} = \mathcal{F}^{\mathcal{E}} \cup \mathcal{F}^{\mathcal{R}}$. However, we only consider deviations regarding map elements $\mathcal{F}^{\mathcal{E}}$ in the following. To obtain map deviations, a set of map elements $\widehat{\mathcal{E}}$ extracted from sensor data is associated with a set of examined, presumably deviating map elements $\widetilde{\mathcal{E}}$, which yields the set of evaluated map

elements $\mathcal{E}^{\text{eval}} = \mathcal{V} \cup \mathcal{U} \cup \mathcal{F}^{\mathcal{E}}$, with the set of *verifications* $\mathcal{V}$ comprising successfully associated ("verified") elements, $\mathcal{U}$ being the set of "unknown" elements that could not be evaluated due to occlusion, and $\mathcal{F}^{\mathcal{E}}$ being the set of "deviating" map elements. To successfully associate elements, their major class and type-dependent overlap criteria must be fulfilled. Moreover, the set $\mathcal{F}^{\mathcal{E}} = \mathcal{F}^{\text{PS}} \cup \mathcal{F}^{\text{A}}$ can be further categorized into the set of deviating physical and semantical map elements $\mathcal{F}^{\text{PS}}$, referring to the existence of respective elements, and the set of elements with attributional deviations $\mathcal{F}^{\text{A}}$. In our approach to MDD, physical and semantical "deviations" $\mathcal{F}^{\text{PS}} = \mathcal{D} \cup \mathcal{I} \cup \mathcal{S}$ are determined first during the association step, comprising elements that are missing in the examined map (*deletions* $\mathcal{D}$), falsely existing elements (*insertions* $\mathcal{I}$), and replaced elements (*substitutions* $\mathcal{S}$). In a second step, map elements with attributional deviations $\mathcal{F}^{\text{A}}$ may be obtained from the initial set of verifications $\mathcal{V}$ by comparing attributes of associated elements on a more detailed level. As the second step is straightforward, we focus only on the more challenging predecessor step of predicting $\mathcal{F}^{\text{PS}} = \mathcal{D} \cup \mathcal{I} \cup \mathcal{S}$ in this work. An examined element $e \in \widetilde{\mathcal{E}}$ for which we detect a deviation ($e \in (\mathcal{I} \cup \mathcal{S}) \subset \widetilde{\mathcal{E}}$) is called "falsified".

B. OVERVIEW ON EXAMINED MDD VARIANTS

The method variants performing MDD that we compare in this work are shown in Fig. 2. Subsequently, we briefly introduce these variants, with respective details provided in Sections III-C to III-E.

Our reference method for MDD (MDD-SC) in Fig. 2 (a) utilizes a sensor-only (-S) object detection DNN and a subsequent comparison (-C), serving as a performance reference for later evaluations. Specifically, map elements are first detected in the sensor data using an object detection algorithm, yielding the set of predicted map elements $\widehat{\mathcal{E}}$, which is then compared to the examined set of (potentially deviating) map elements $\widetilde{\mathcal{E}}$ to obtain the set of evaluated map elements $\mathcal{E}^{\text{eval}}$, which comprises respective map deviations $\mathcal{F}^{\text{PS}}$. To this end, a point cloud $\mathcal{P}$ as unordered set of points is sorted into a spatial voxel grid $\boldsymbol{m}$ of fixed size featuring three spatial dimensions (subsequently referred to as "LiDAR feature map", see Fig. 1). Our DNN uses an encoder stage to learn an optimal feature representation for points contained in each voxel, which yields the encoded LiDAR feature grid $\boldsymbol{g}$, that preserves the spatial dimensions. The network's backbone further extracts abstract features of higher semantics and provides the extracted feature grid $\widetilde{\boldsymbol{g}}$ as input to the network heads. Specifically, we attach three individual network heads (visualized as one "heads" block in Fig. 2 for simplicity) that output bounding shapes for traffic signs $\boldsymbol{u}^{\text{s}}$, traffic lights $\boldsymbol{u}^{\text{l}}$, and poles $\boldsymbol{u}^{\text{p}}$, respectively. In a post-processing step, these respective output feature maps are converted into the set of predicted map elements $\widehat{\mathcal{E}}$ as input to the aforementioned comparison with $\widetilde{\mathcal{E}}$. As we use high-density point clouds that are free of occlusions as sensor data for our proof of concept, the evaluated set of map



(a) MDD-SC (reference): "Sensor-only object detection DNN"

(b) MDD-MC: "Map-supported object detection DNN"

(c) MDD-M: "Map-supported deviation detection DNN"

FIGURE 2. Method variants performing map deviation detection examined in this work. The reference method MDD-SC (a) detects traffic sign, light, and pole objects in sensor data $\mathcal{P}$ (-S) and obtains map deviations based on comparison (-C). The method MDD-MC (b) supports the object detection task using the (deviating) map $\widetilde{\mathcal{E}}$ as additional input (-M). This approach is further advanced using a specialized deviation detection DNN in MDD-M (c) that directly predicts map deviations without explicit comparison. Blue: DNN components. Green: Map processing path.

elements $\mathcal{E}^{\text{eval}} = \mathcal{V} \cup \mathcal{F}^{\text{PS}} = \mathcal{V} \cup \mathcal{D} \cup \mathcal{I} \cup \mathcal{S}$ omits the set of unknown elements $\mathcal{U}$, which yields the four "evaluation states" $\mathcal{S} = \{\text{VER}, \text{DEL}, \text{INS}, \text{SUB}\}$ considered in this work: verification, deletion, insertion, or substitution, respectively.

As shown in Fig. 2 (b), the proposed and more advanced method MDD-MC, featuring a map-supported object detection DNN, uses the encoded, presumably deviating map as additional input $\boldsymbol{m}'$ (-M) to facilitate the detection task, with map deviations still obtained by comparing detected objects in $\widehat{\mathcal{E}}$ with the set of examined map elements $\widetilde{\mathcal{E}}$ (-C). The network can leverage the map as source of initial hypotheses regarding possible element locations and features, and as a source for contextual knowledge. However, as the map contains deviations, the network cannot rely on the map only to detect missing map elements (deletions) or falsify existing map hypotheses (insertions or substitutions) in $\widetilde{\mathcal{E}}$, but has to decide internally when to rely on sensor or map data, respectively. To this end, MDD-MC additionally includes the map encoding step in Fig. 2 (b), whereby map elements in $\widetilde{\mathcal{E}}$ are matched to a voxel grid of the same size as the encoded LiDAR grid $\boldsymbol{g}$, with respective element features (e.g., major class and bounding shape features) being incorporated into matched voxels, yielding the map representation $\boldsymbol{m}'$.

The third method variant includes a map-supported deviation detection DNN (`MDD-M`) that directly evaluates the correctness of the additional map input (`-M`) without an explicit comparison. Specifically, we force the network to classify the evaluation state $s \in \mathcal{S}$ for each map element individually by using a specialized loss function, with a changed network output including the evaluation state classification in $\widetilde{\boldsymbol{u}}^{\mathrm{s}}$, $\widetilde{\boldsymbol{u}}^{\mathrm{l}}$, and $\widetilde{\boldsymbol{u}}^{\mathrm{p}}$. In consequence, the network has to directly evaluate an existing element hypothesis by comparing available sensor data to map data. Thereby, the network needs to model expected sensor data distributions for specific map element types internally to infer an element's evaluation state, i.e., by comparing current sensor data to a learned, typical distribution in order to verify an element. A point density ablation study (cf. Section III-C1) will demonstrate the high performance of such an approach in the presence of degenerated sensor data, where the evaluation state has to be determined using few LiDAR points only.

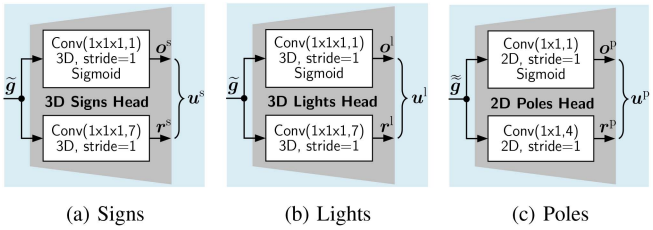
C. REFERENCE VARIANT MDD-SC: SENSOR-ONLY

In this section, we provide details regarding our reference method `MDD-SC`. First, we briefly summarize our DNN architecture for object detection and the loss function used during training. Subsequently, we present the required post-processing and comparison steps.

1) OBJECT DETECTION DNN

For the object detection DNN in Fig. 2 (a) and (b) used for the methods `MDD-SC` and `MDD-MC`, respectively, we apply the `3DHDNet` architecture [10] comprising three stages: encoder, backbone, and heads. Compared to the original architecture, we provide a multitask extension of the network with multiple heads attached to the backbone, simultaneously predicting signs, lights, and poles. The network provides a learned encoding $\boldsymbol{g}$ for the point cloud input $\mathcal{P}$ [19], [71]. For more details regarding the internal operation of the encoder and backbone stage, see Appendix A.

Let $\boldsymbol{p} = (p^{\mathrm{int}}, \boldsymbol{p}^{\mathrm{crd}}) \in \mathcal{P}$ be a single point of the point cloud $\mathcal{P}$ that is input to any of the three methods depicted in Fig. 2, with $p^{\mathrm{int}} \in \mathbb{I} = [0, 1]$ being an intensity measurement of the reflected LiDAR beam, and $\boldsymbol{p}^{\mathrm{crd}} \in \mathbb{R}^3$ being the Cartesian coordinate of a point. As a pre-processing step, the point cloud $\mathcal{P}$ is first *voxelized* into a 3D voxel grid with N^{x}, N^{y}, and N^{z} voxels in the x-, y-, and z-dimension of the grid. Subsequently, to increase network speed, only the N occupied voxels with a maximum of $K = 96$ points per voxel are collected into the augmented LiDAR feature map $\boldsymbol{m} = (\boldsymbol{m}_{n,k})$ of size $N \times K \times 10$ as input to the encoder, with $n \in \mathcal{N} = \{1, \ldots, N\}$ being the voxel index and $k \in \mathcal{K} = \{1, \ldots, K\}$ being the point index, respectively. Each (augmented) point of the grid $\boldsymbol{m}_{n,k} = (p^{\mathrm{int}}_{n,k}, \boldsymbol{p}^{\mathrm{crd}}_{n,k}, \boldsymbol{p}^{\mathrm{crd}}_{n,k} - \bar{\boldsymbol{p}}_n, \boldsymbol{p}^{\mathrm{crd}}_{n,k} - \boldsymbol{v}_n)$ provides ten features, with $\bar{\boldsymbol{p}}_n \in \mathbb{R}^3$ being the mean of all Cartesian point measurements contained in voxel n, and $\boldsymbol{v}_n \in \mathbb{R}^3$ being the Cartesian center coordinate of the point's assigned voxel n.



(a) Signs (b) Lights (c) Poles

FIGURE 3. Map element heads (gray) for signs (a), lights (b), and poles (c). A convolution operation (3D or 2D) is denoted as Conv(C, F), with C being the kernel size and F the number of filters, respectively.

First, the *encoder* stage encodes all points contained in a voxel, providing a single feature vector of length $L = 256$ for each voxel, which yields the encoded LiDAR grid $\boldsymbol{g} \in \mathbb{R}^{N^{\mathrm{x}} \times N^{\mathrm{y}} \times N^{\mathrm{z}} \times L}$. Subsequently, the *3D backbone* processes $\boldsymbol{g}$ using a series of 3D (transposed) convolutions, extracting more abstract features and including context from surrounding voxels, which provides $\widetilde{\boldsymbol{g}} \in \mathbb{R}^{N^{\mathrm{x}} \times N^{\mathrm{y}} \times N^{\mathrm{z}} \times L'}$, with $L' = 384$ features per voxel. As shown in Fig. 3, in a last step, the abstract feature grid $\widetilde{\boldsymbol{g}}$ is decoded by three individual *map element heads* to provide the network outputs $\boldsymbol{u}^{\mathrm{s}}$, $\boldsymbol{u}^{\mathrm{l}}$, $\boldsymbol{u}^{\mathrm{p}}$ for signs, lights, and poles, respectively, which we define subsequently. Note that for poles, $\widetilde{\boldsymbol{g}}$ is reorganized by concatenating all features along the vertical z-dimension, providing the poles head input $\widetilde{\boldsymbol{g}} \in \mathbb{R}^{N^{\mathrm{x}} \times N^{\mathrm{y}} \times (N^{\mathrm{z}} \cdot L')}$.

Each head operates in a single-shot fashion [76] simultaneously predicting existing likelihoods and regression bounding shapes for a set of predefined objects, so-called "anchors". If an anchor is likely to contain (part of) a real-world object, the existence likelihood increases. Let $\mathrm{et} \in \mathcal{T} = \{\mathrm{s}, \mathrm{l}, \mathrm{p}\}$ be the map element type, `sign`, `light`, or `pole`, with $\mathcal{T}$ being the set of considered map element types. For signs and lights, we employ the 3D anchor grids $\mathcal{G}^{\mathrm{s}}, \mathcal{G}^{\mathrm{l}} = \{1, 2, \ldots, G\}$ with $G = N^{\mathrm{x}} \cdot N^{\mathrm{y}} \cdot N^{\mathrm{z}}$ being the number of voxels, allowing for the vertical resolution of individual objects. For poles, due to their ground placement, we employ a 2D anchor grid in the x-y-plane, which reduces runtime and memory requirements, with $\mathcal{G}^{\mathrm{p}} = \{1, 2, \ldots, G'\}$ and $G' = N^{\mathrm{x}} \cdot N^{\mathrm{y}}$ being the number of cells in the 2D grid. A map element head output $\boldsymbol{u}^{\mathrm{et}} = (\boldsymbol{u}^{\mathrm{et}}_g)$ contains predictions $\boldsymbol{u}^{\mathrm{et}}_g$ for each anchor, with $g \in \mathcal{G}^{\mathrm{et}}$ being the anchor index. Each anchor prediction $\boldsymbol{u}^{\mathrm{et}}_g = (o^{\mathrm{et}}_g, \boldsymbol{r}^{\mathrm{et}}_g)$ comprises an *object detection head* output o^{et}_g and a *regression head* output $\boldsymbol{r}^{\mathrm{et}}_g$. According to Fig. 3, both outputs are obtained by decoding $\widetilde{\boldsymbol{g}}$, employing 3D or 2D convolutions, depending on the anchor grid.

The object detection head predicts an existence likelihood for each anchor $o^{\mathrm{et}}_g \in \mathbb{I}$, with $\mathbb{I} = [0, 1]$, while the regression head adopts the anchor's (predefined) bounding shape to match the size and position of a real-world object. The bounding shape parameters differ for each map element head. For *poles*, we employ a bounding cylinder with the set of regression parameters $\mathcal{Q}^{\mathrm{p}} = \{p^{\mathrm{x}}, p^{\mathrm{y}}, p^{\mathrm{z}}, d\}$, with p^{x}, p^{y}, p^{z} being the Cartesian position of a pole's base point, and d being the pole's diameter. For *lights*, we use a bounding box $\mathcal{Q}^{\mathrm{l}} = \{p^{\mathrm{x}}, p^{\mathrm{y}}, p^{\mathrm{z}}, h, w, \varphi\}$, with h being the bounding box

height, w the width of a squared base plate, and φ being the light's orientation. Last, for *signs* modeled as bounding rectangles, the set $\mathcal{Q}^{\mathrm{s}} = \{p^{\mathrm{x}}, p^{\mathrm{y}}, p^{\mathrm{z}}, h, w, \varphi\}$ applies, with h and w indicating the rectangle's height and width, respectively. For signs and lights, $p^{(\cdot)}$ indicates the element's center position.

Instead of directly predicting bounding shape parameters, the regression head predicts the difference between a real-world object's bounding shape (superscript O), and the bounding shape of the respective anchor (superscript A) for normalization purposes. With $s^{\mathrm{x}}_{\mathrm{vox}} \times s^{\mathrm{y}}_{\mathrm{vox}} \times s^{\mathrm{z}}_{\mathrm{vox}} = s^3_{\mathrm{vox}}$ and $s_{\mathrm{vox}} = 40\,\mathrm{cm}$ being the size of a cubic voxel, and the orientation φ being encoded as complex number with real and imaginary components φ^{re} and φ^{im}, we define the regression head outputs $\boldsymbol{r}^{\mathrm{s}}_g = \boldsymbol{r}^{\mathrm{l}}_g = (r_g^{(p^{\mathrm{x}})}, r_g^{(p^{\mathrm{y}})}, r_g^{(p^{\mathrm{z}})}, r_g^{(w)}, r_g^{(h)}, r_g^{(\varphi^{\mathrm{re}})}, r_g^{(\varphi^{\mathrm{im}})})$, and $\boldsymbol{r}^{\mathrm{p}}_g = (r_g^{(p^{\mathrm{x}})}, r_g^{(p^{\mathrm{y}})}, r_g^{(p^{\mathrm{z}})}, r_g^{(d)})$ with:

$$r_g^{(p)} = \frac{\left(p_g^{\mathrm{O}} - p_g^{\mathrm{A}}\right)}{s_{\mathrm{vox}}}, \quad p \in \left\{p^{\mathrm{x}}, p^{\mathrm{y}}, p^{\mathrm{z}}\right\}, \tag{1}$$

$$r_g^{(\ell)} = \log\left(\frac{\ell_g^{\mathrm{O}}}{\ell_g^{\mathrm{A}}}\right), \quad \ell \in \{w, h, d\}, \tag{2}$$

$$r_g^{(\varphi^{\mathrm{re}})} + j \cdot r_g^{(\varphi^{\mathrm{im}})} = \cos\left(\kappa \cdot \varphi^{\mathrm{O}}\right) + j \cdot \sin\left(\kappa \cdot \varphi^{\mathrm{O}}\right), \tag{3}$$

with $j = \sqrt{-1}$ being the imaginary unit, and $\kappa = 1$ for lights. Regarding signs, the network cannot easily distinguish between a sign's front and back without camera images.

Thus, we limit a sign's orientation to $\varphi \in [-90°, 90°]$, instead of encoding a full 360° range. To ensure that signs with the same spatial orientation (difference of 180°) generate no loss, we use the factor $\kappa = 2$ in (3). We derive the default anchor parameters $(\cdot)^{\mathrm{A}}_g$ based on the ground truth (GT) parameter distributions (cf. Appendix B), which yields $w^{\mathrm{A}}_g = h^{\mathrm{A}}_g = 0.65\,\mathrm{m}$ for signs, $w^{\mathrm{A}}_g = 0.3\,\mathrm{m}$ and $h^{\mathrm{A}}_g = 0.9\,\mathrm{m}$ for lights, and $d^{\mathrm{A}}_g = 0.2\,\mathrm{m}$ for poles.

2) OBJECT DETECTION LOSS

To generate the target existence likelihoods $\overline{o}^{\mathrm{et}}_g$ and target regression features $\overline{\boldsymbol{r}}^{\mathrm{et}}_g$ (with $\overline{(\cdot)}$ denoting ground truth (GT)) during training, GT map elements have to be matched to the anchor grid, setting matching anchors $\overline{o}^{\mathrm{et}}_g = 1$ and their regression features $\overline{\boldsymbol{r}}^{\mathrm{et}}_g$ according to (1), (2), and (3), whereby a single element can match with multiple anchors. To this end, we employ type-specific matching strategies, association metrics, and criteria. Anchors with only a small overlap with a GT element are ignored during training using a "don't care" state. For more details, see Appendix C.

We define the loss function optimizing the object detection DNN applied in `MDD-SC` and `MDD-MC` (cf. Fig. 2) similar to `PointPillars` [71] and `SECOND` [73] as

$$J = \sum_{\mathrm{et} \in \mathcal{T}} \frac{1}{N^{\mathrm{et}}} \cdot \sum_{g \in \mathcal{G}^{\mathrm{et}}} \lambda \cdot J\left(\overline{o}^{\mathrm{et}}_g, o^{\mathrm{et}}_g\right) + (1 - \lambda) \cdot J\left(\overline{\boldsymbol{r}}^{\mathrm{et}}_g, \boldsymbol{r}^{\mathrm{et}}_g\right), \tag{4}$$

with $\mathrm{et} \in \mathcal{T} = \{\mathrm{s}, \mathrm{l}, \mathrm{p}\}$ denoting the element type (`sign`, `light`, or `pole`), N^{et} being the type-specific number of elements contained in a training sample, $J(\overline{o}^{\mathrm{et}}_g, o^{\mathrm{et}}_g)$ and $J(\overline{\boldsymbol{r}}^{\mathrm{et}}_g, \boldsymbol{r}^{\mathrm{et}}_g)$ denoting the object detection and the regression loss obtained for a specific element type, respectively, and $\lambda = 2/3$ being a weight factor. We formulate the object detection loss as focal loss [77], which focuses on particularly hard-to-detect elements using adaptive weights. Using the original paper settings [77] for the weight factors $\alpha = 0.25$, $\beta = 2$, and with the mask factor $\lambda^{\mathrm{mask}}_g \in \{1, 0\}$ being zero in the "don't care" state, we define:

$$J\left(\overline{o}^{\mathrm{et}}_g, o^{\mathrm{et}}_g\right) = -\lambda^{\mathrm{mask}}_g \alpha_g \cdot \left(1 - \gamma_g\right)^{\beta} \cdot \log\left(\gamma_g\right),$$

$$\gamma_g = \begin{cases} o^{\mathrm{et}}_g & \text{if element in voxel } g \ \left(\overline{o}^{\mathrm{et}}_g = 1\right) \\ 1 - o^{\mathrm{et}}_g & \text{otherwise} \end{cases},$$

$$\alpha_g = \begin{cases} \alpha & \text{if element in voxel } g \ \left(\overline{o}^{\mathrm{et}}_g = 1\right) \\ 1 - \alpha & \text{otherwise} \end{cases}, \tag{5}$$

whereby an anchor contained in voxel g is weighted higher when a prediction is further away from the target value (see the distinction of cases depending on $\overline{o}^{\mathrm{et}}_g$ in (5)).

The type-specific regression loss is formulated using the smooth L1-loss (also known as Huber loss [78]):

$$J\left(\overline{\boldsymbol{r}}^{\mathrm{et}}_g, \boldsymbol{r}^{\mathrm{et}}_g\right) = \lambda^{\mathrm{obj}}_g \sum_{q \in \mathcal{Q}^{\mathrm{et}}} \mathrm{smoothL1}\left(\Delta r_g^{(q)}\right), \tag{6}$$

$$\mathrm{smoothL1}\left(\Delta r_g^{(q)}\right) = \begin{cases} 0.5\left(\Delta r_g^{(q)}\right)^2 & \text{if } |\Delta r_g^{(q)}| \leq 1 \\ |\Delta r_g^{(q)}| - 0.5 & \text{otherwise} \end{cases}, \tag{7}$$

with $q \in \mathcal{Q}^{\mathrm{et}}$ being a type-specific regression parameter defined in Section III-C1, $\Delta r_g^{(q)} = r_g^{(q)} - \overline{r}_g^{(q)}$ being the difference between predicted and target regression values, respectively, and with $\lambda^{\mathrm{obj}}_g \in \{1, 0\}$ being 1 only if an element is present in voxel g.

3) POST-PROCESSING AND COMPARISON

To obtain the list of predicted elements $\widehat{\mathcal{E}}$ during inference for both `MDD-SC` and `-MC`, the map element head outputs $\boldsymbol{u}^{\mathrm{et}}$ are post-processed (cf. Fig. 2 (a) and (b)). First, only map element predictions for single voxels g with a predicted existence likelihood o^{et}_g exceeding an element-type-specific threshold $\Theta^{\mathrm{et}}_{\mathrm{score}}$ are considered as a valid detection:

$$o^{\mathrm{et}}_g \geq \Theta^{\mathrm{et}}_{\mathrm{score}}, \tag{8}$$

with the selection of thresholds being discussed in Section V-A. For valid detections, the normalization of bounding shape predictions is reverted. As multiple anchors can make predictions for the same real-world object, respective overlapping bounding shapes are filtered using non-maximum-suppression [71] to obtain the predicted map element set $\widehat{\mathcal{E}}$ comprising signs, lights, and poles, respectively. Thereby, only the element with the highest existence

likelihood among overlapping predictions remains. To measure the overlap, we apply the association metrics detailed in Appendix C.

To obtain verifications $\mathcal{V}$ and map deviations $\mathcal{D} \cup \mathcal{I} \cup \mathcal{S}$, predicted map elements $\widehat{\mathcal{E}}$ are now compared with elements in the potentially deviating map $\widetilde{\mathcal{E}}$. In the first association step, map elements in $\widehat{\mathcal{E}}$ and $\widetilde{\mathcal{E}}$ with the same major type (`sign`, `light`, `pole`) are compared using again the association metrics. Successfully associated elements are appended to $\mathcal{V}$, while predicted elements in $\widehat{\mathcal{E}}$ that cannot be found in the examined map $\widetilde{\mathcal{E}}$ are considered as deletion and appended to $\mathcal{D}$. Vice versa, elements in $\widetilde{\mathcal{E}}$ for which no prediction from $\widehat{\mathcal{E}}$ can be associated, are considered as falsely existing and appended to the set of insertions $\mathcal{I}$. Substitutions $\mathcal{S}$ are a special case, where one element of a specific major type (e.g., `light`) is interchanged with another element of a different major type (e.g., `sign`). In our case, only signs and lights are physically interchangeable with one another. Specifically, a sign cannot be interchanged with a pole, as the pole must already exist to serve as a mounting location. If the sign is mounted on a wall, there is no space available for a pole. Similar considerations apply regarding lights. For instance, after the first association step, let's assume a light falsely exists in $\widetilde{\mathcal{E}}$ (insertion), with a sign missing in $\widehat{\mathcal{E}}$ (deletion) at the same location. In this case, to identify such substitutions in a second step, deletions and insertions are associated among signs and lights. As the aforementioned association criteria are defined for elements of the same major type, we simply require the Euclidean distance d_{E} between elements to be $d_{\mathrm{E}} \leq \Theta_{\mathrm{E}}^{\mathrm{SUB}}$ for successful association of deletions and insertions, with $\Theta_{\mathrm{E}}^{\mathrm{SUB}} = 0.3\,\mathrm{m}$ being a threshold value. If an association is successful, both deviations are removed from respective sets $\mathcal{D}$ and $\mathcal{I}$, and replaced by a single substitution appended to $\mathcal{S}$.

D. MDD-MC: MAP-SUPPORTED OBJECT DETECTION

The method `MDD-MC` follows the same concepts as presented in the previous section for `MDD-SC`, while additionally applying the map representation $\boldsymbol{m}'$ as input to the object detection DNN (cf. Fig. 2 (b)), providing the network with initial hypotheses for possible map elements locations and features. In this section, we first define the map representation and subsequently present the applied *map encoding* procedure to obtain $\boldsymbol{m}'$, with a respective example result depicted in Fig. 7 (a). Last, we describe the integration of the generated map representation into the DNN.

Let the tensor $\boldsymbol{m}' = (\boldsymbol{m}'_g)$ be of size $N^{\mathrm{x}} \times N^{\mathrm{y}} \times N^{\mathrm{z}} \times 10$, whereby g denotes the voxel index of the spatial grid (cf. Section III-C1). Each map feature vector $\boldsymbol{m}'_g = (P^{\mathrm{s}}_g, P^{\mathrm{l}}_g, P^{\mathrm{p}}_g, r^{(p^{\mathrm{x}})}_g, r^{(p^{\mathrm{y}})}_g, r^{(p^{\mathrm{z}})}_g, r^{(\ell)}_g, r^{(h)}_g, r^{(\varphi^{\mathrm{im}})}_g, r^{(\varphi^{\mathrm{re}})}_g)$ provides ten features, with $P^{(\cdot)} \in \mathbb{P}$ and $\mathbb{P} = \{0, 1\}$ being a prior existence hypothesis that a sign s, light l, or pole p is present in voxel g, whereby we allow only one hypothesis per voxel, resulting in a one-hot encoding for $P^{\mathrm{s}}, P^{\mathrm{l}}, P^{\mathrm{p}}$. If an element of a specific type is present in voxel g, the respective regression features $r^{(\cdot)}$ are set in a normalized fashion according to (1), (2), and (3). If a pole is present, we provide the respective pole diameter $r^{(\ell)}_g = r^{(d)}_g$ with the other regression features set to 0. For signs and lights, we provide the width parameter $r^{(\ell)}_g = r^{(w)}_g$, respectively.

To obtain map feature vectors, map elements $\widetilde{\mathcal{E}}$ have to be matched to voxels in the spatial grid of $\boldsymbol{m}'$, whereby a single element can match with multiple voxels. For such matching, we approximate map elements using cuboid shapes. Such an approximation is less precise than the matching strategies applied during target generation (cf. Appendix C), but provides a greatly reduced runtime during training and inference. For a successful match of a map element with voxel g, we require

$$\bigwedge_{\mathrm{dim} \in \{\mathrm{x},\mathrm{y},\mathrm{z}\}} |p^{\mathrm{dim}} - v^{\mathrm{dim}}_g| < \Theta^{\mathrm{dim}}_{\mathrm{dist}}, \tag{9}$$

with $\mathrm{dim} \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$ being a spatial dimension, $\Theta^{\mathrm{dim}}_{\mathrm{dist}}$ being a dimension-specific threshold value, and p^{dim} and v^{dim} being the element's and the voxel's center positions in that dimension, respectively. Furthermore, we define

$$\Theta^{\mathrm{dim}}_{\mathrm{dist}} = \nu \frac{\max\left(\ell, s^{\mathrm{dim}}_{\mathrm{vox}}\right)}{2}, \tag{10}$$

with $\nu = 1.1$ being an empirically determined factor enlarging the element's shape to include voxels surrounding the element as matches. Regarding poles, we apply the diameter $\ell = d$ for both the x- and y-dimension, and h being a pole-class-specific default height value for the z-dimension. Otherwise, for signs and lights, we use the width parameter $\ell = w$ for the x- and y-dimension. The max-operation ensures a minimal element size to obtain a match also for small elements. For poles and (squared-shaped) lights, this approximation yields sufficient matching results. For signs, however, matches are further refined to account for their rotated shape using the matching strategies applied during target generation. For unmatched voxels, we set $\boldsymbol{m}' = \boldsymbol{0}$.

Finally, to integrate the obtained map representation $\boldsymbol{m}'$ into the network, $\boldsymbol{m}'$ (providing 10 features per voxel) is concatenated along the feature dimension to the encoded LiDAR grid $\boldsymbol{g}$ providing L' features, yielding a spatial grid of size $N^{\mathrm{x}} \times N^{\mathrm{y}} \times N^{\mathrm{z}} \times (10 + L')$ as input to the backbone.

Note that the camera modality can be integrated in the same way as $\boldsymbol{m}'$, following the concept from [11]. For instance, a bounding box for each individual voxel can be projected into the camera image to crop and resize [79] respective image features (i.e., RGB or abstract features), which subsequently can be used as a voxel's feature vector.

E. MDD-M: MAP-SUPPORTED DEVIATION DETECTION

In this section, we first present our specialized MDD network and the according loss function used for training. We close with a presentation of the post-processing specific to `MDD-M`.

1) DEVIATION DETECTION DNN

The deviation detection DNN applied in method MDD-M as depicted in Fig. 2 (c) directly predicts map deviations instead of performing a mere object detection and obtaining deviations by comparison with the (deviating) map, omitting the need for such comparison. To this end, we adopt the network heads, now providing the outputs $\widetilde{\boldsymbol{u}}^{\mathrm{s}}$, $\widetilde{\boldsymbol{u}}^{\mathrm{l}}$, and $\widetilde{\boldsymbol{u}}^{\mathrm{p}}$, while leaving the rest of the network architecture (encoder and backbone) and the pre-processing steps (voxelization and map encoding) unchanged.

To predict map deviations, we substitute the output of the object detection head $\boldsymbol{o}^{\mathrm{et}}$ (cf. Section III-C) in the respective map element head output $\boldsymbol{u}^{\mathrm{et}} = (\boldsymbol{o}^{\mathrm{et}}, \boldsymbol{r}^{\mathrm{et}})$, as used for MDD-SC and MDD-MC, with the *deviation detection head* output $\boldsymbol{d}^{\mathrm{et}}$, which yields the changed map element head output $\widetilde{\boldsymbol{u}}^{\mathrm{et}} = (\boldsymbol{d}^{\mathrm{et}}, \boldsymbol{r}^{\mathrm{et}})$. Thereby, the spatial resolution of the anchor grids remains unchanged, comprising N^{x}, N^{y}, and N^{z} voxels in the respective dimensions, with predictions being made for each anchor (or voxel, respectively). The deviation detection head output $\boldsymbol{d}_g$ for an anchor g comprises four score values, representing the evaluation states $\mathcal{S} = \{\mathtt{VER}, \mathtt{DEL}, \mathtt{INS}, \mathtt{SUB}\}$: $\boldsymbol{d}_g = (s^{\mathtt{VER}}, s^{\mathtt{DEL}}, s^{\mathtt{INS}}, s^{\mathtt{SUB}})$, with $s^{(\cdot)} \in \mathbb{I} = [0, 1]$ being a state score value. To provide the target $\overline{\boldsymbol{d}}_g$ during training, we set the respective score value to 1 according to the element's evaluation state. Otherwise, if no element is present in a voxel, all score values are set to 0. In case of substitutions, the network is trained to predict the map element type present in the sensor data. For instance, in case of the map falsely hypothesizing a light, which is substituted by a sign in the real world, we set $\overline{\boldsymbol{d}}_g^{\mathrm{s}} = (0, 0, 0, 1)$ to predict a substitution and $\overline{\boldsymbol{r}}_g^{\mathrm{s}}$ according to (1), (2), and (3) to predict the sign's bounding shape.

2) DEVIATION DETECTION LOSS

Accordingly, we change the loss function (4) optimizing the network to comprise the deviation detection loss $J(\overline{\boldsymbol{d}}_g^{\mathrm{et}}, \boldsymbol{d}_g^{\mathrm{et}})$:

$$\widetilde{J} = \sum_{\mathrm{et} \in \mathcal{T}} \frac{1}{N^{\mathrm{et}}} \cdot \sum_{g \in \mathcal{G}^{\mathrm{et}}} \lambda \cdot J\left(\overline{\boldsymbol{d}}_g^{\mathrm{et}}, \boldsymbol{d}_g^{\mathrm{et}}\right) + (1 - \lambda) \cdot J\left(\overline{\boldsymbol{r}}_g^{\mathrm{et}}, \boldsymbol{r}_g^{\mathrm{et}}\right). \tag{11}$$

The deviation detection loss $J(\overline{\boldsymbol{d}}_g^{\mathrm{et}}, \boldsymbol{d}_g^{\mathrm{et}})$ incorporates the classification of evaluation states $\mathcal{S}$ formulated as focal loss:

$$J\left(\overline{\boldsymbol{d}}_g^{\mathrm{et}}, \boldsymbol{d}_g^{\mathrm{et}}\right) = -\lambda_g^{\mathrm{mask}} \sum_{s \in \mathcal{S}} \alpha_{g,s} \cdot \left(1 - \gamma_{g,s}\right)^{\beta} \cdot \log\left(\gamma_{g,s}\right),$$

$$\gamma_{g,s} = \begin{cases} d_{g,s}^{\mathrm{et}} & \text{if state in voxel } g \left(\overline{d}_{g,s}^{\mathrm{et}} = 1\right) \\ 1 - d_{g,s}^{\mathrm{et}} & \text{otherwise} \end{cases},$$

$$\alpha_{g,s} = \begin{cases} \alpha & \text{if state in voxel } g \left(\overline{d}_{g,s}^{\mathrm{et}} = 1\right) \\ 1 - \alpha & \text{otherwise} \end{cases}. \tag{12}$$

Further, we apply the same regression loss as previously formulated in (6). In case of insertions (elements not existing in the sensor data), we train the network to reproduce the element's bounding shape as provided by $\boldsymbol{m}'$.

3) POST-PROCESSING

As the network directly classifies the evaluation states for elements of the examined set $\widetilde{\mathcal{E}}$, which are encoded as $\boldsymbol{m}'$ and provided as input to the network, verifications $\mathcal{V}$, deletions $\mathcal{D}$, insertions $\mathcal{I}$, and substitutions $\mathcal{S}$ can be directly obtained from the network output. Only a prediction for voxel g with a state score $s^{(\cdot)}$ exceeding a specific threshold is considered as valid detection. As opposed to the object detection DNN used in MDD-SC and MDD-MC, where three element-type-specific thresholds are applied, the deviation detection DNN in MDD-M allows for the selection and optimization of in total $3 \times 4 = 12$ type- and state-specific thresholds Θ_s^{et}, with $\mathrm{et} \in \mathcal{T}$ being the element type and $s \in \mathcal{S} = \{\mathtt{VER}, \mathtt{DEL}, \mathtt{INS}, \mathtt{SUB}\}$ being the evaluation state:

$$\max_{s \in S}\left(d_{g,s}^{\mathrm{et}}\right) \geq \Theta_s^{\mathrm{et}}. \tag{13}$$

According to (13), an element is appended to respective sets $\mathcal{V}$, $\mathcal{D}$, $\mathcal{I}$, and $\mathcal{S}$ only if the respective threshold is exceeded.

The post-processing procedure for MDD-M is different for elements providing an existing hypothesis in $\widetilde{\mathcal{E}}$ (predicted as verifications $\mathcal{V}$, insertions $\mathcal{I}$, or substitutions $\mathcal{S}$), and elements missing in $\widetilde{\mathcal{E}}$ (deletions $\mathcal{D}$). First, we consider elements in the examined set $\widetilde{\mathcal{E}}$. Recall that each element in $\widetilde{\mathcal{E}}$ is matched to a unique set of voxels in $\boldsymbol{m}'$ during map encoding (cf. Section III-D). As the spatial grid size of both $\boldsymbol{m}'$ and deviation detection head output $\boldsymbol{d}^{\mathrm{et}}$ given by N^{x}, N^{y}, and N^{z} are equal, the set of matched voxels in $\boldsymbol{m}'$ corresponds to a unique set of matched anchors $\mathcal{A} \subset \mathcal{G}^{\mathrm{et}}$ in $\boldsymbol{d}^{\mathrm{et}}$, predicting the evaluation state of an examined element. Thus, for each element in $\widetilde{\mathcal{E}}$, we query only the specific set of matched anchors $\mathcal{A}$ instead of the entire anchor grid $\mathcal{G}^{\mathrm{et}}$, selecting the evaluation state (VER, INS, or SUB) of the anchor $a \in \mathcal{A}$ providing the highest score $d_a^{\max} = \max_{s \in S}(d_{g,s}^{\mathrm{et}})$ among the set $\mathcal{A}$, with a being the matched anchor index, and averaging the regression features of all anchors in $\mathcal{A}$ predicting the selected evaluation state. Such procedure ensures that each element in $\widetilde{\mathcal{E}}$ is evaluated and reduces computational cost. Second, to regard deletions, we apply the ordinary object detection post-processing (cf. Section III-C3). Specifically, we consider all anchors for which the deletion score $d_{g,\mathtt{DEL}}^{\mathrm{et}}$ is highest. Obtained deletion predictions are subsequently filtered using non-maximum-suppression.

IV. EXPERIMENTAL SETUP

In this section, we describe our experimental setup. First, we present the utilized dataset and applied metrics. Last, we refer to the training strategy.

A. DATASET

We employ the 3DHD CityScenes dataset [10] depicted in Fig. 4, which provides a large-scale HD map with an underlying high-density point cloud. The map comprises 20,163 signs, 5,762 lights, and 67,540 poles. However, we exclude bollards being the least significant but most varying pole class, which leaves 32,283 considered poles in the dataset. If vertically-stacked rectangular signs are of similar size with no spatial gap between, we merge respective signs as the individual boundaries cannot be determined.

3DHD CityScenes offers poses on the HD map comprising geo-location and orientation of the vehicle based on recorded real-world trajectories, providing 57,510, 8,087, and 13,061 poses for the training, validation, and test set, respectively. To generate samples comprising a point cloud $\mathcal{P}$ and examined map elements $\widetilde{\mathcal{E}}$ for training, inference, and evaluation, we take crops from the larger point cloud, only considering the subset of map elements within this crop as part of the sample.

For the MDD task of this work, in particular the detection of verifications $\mathcal{V}$ (map elements correctly representing the sensor data), deletions $\mathcal{D}$ (elements missing in the map), insertions $\mathcal{I}$ (elements falsely existing in the map), and substitutions $\mathcal{S}$ (false element types present in the map data), 3DHD CityScenes is not providing annotations straightaway. Thus, map deviations have to be induced into the map artificially, which we term *map deviation simulation*.

Note that we assume the initially provided map as deviation-free, i.e., all elements represent GT verifications $\overline{\mathcal{V}}$, respectively. To simulate deviations, we use a probabilistic approach, whereby each map element is examined and either left unchanged, remaining a verification with the evaluation state `VER`, or turned into a deviation: deletion `DEL`, insertion `INS`, or substitution `SUB`. Specifically, let $P^{(s)}$ be the probability of a map element having assigned a certain evaluation state $s \in \mathcal{S} = \{\texttt{VER}, \texttt{DEL}, \texttt{INS}, \texttt{SUB}\}$. For validation and test, we apply the probabilities $(P^{\texttt{VER}}, P^{\texttt{DEL}}, P^{\texttt{INS}}, P^{\texttt{SUB}}) = (0.75, 0.1, 0.1, 0.05)$, which we regard as an intermediate setting between a mostly correct map (e.g., with $P^{\texttt{VER}} = 0.95$), and a construction site scene (e.g., with $P^{\texttt{VER}} = 0.25$). For reproducibility purposes, we use a fixed assignment of elements to evaluation states that is generated in an offline-fashion and stored prior to training. The assignment is published along with our code. During training, however, we dynamically simulate deviations for each sample, using higher deviation probabilities $(0.5, 0.2, 0.2, 0.1)$ to speed up the learning process for the method `MDD-M`.

If an element receives the `VER` evaluation state, both element and sensor data are left unchanged, with respective elements appended to the set of GT *verifications* $\overline{\mathcal{V}}$. On the other hand, if an element is assigned the `DEL`, `INS`, or `SUB` state, we turn it into a deviation by modifying map $\widetilde{\mathcal{E}}$ or sensor data $\mathcal{P}$, as depicted in Fig. 5. Subsequently, the simulated deviation is appended to the respective GT set $\overline{\mathcal{D}}$, $\overline{\mathcal{I}}$, or $\overline{\mathcal{S}}$. In particular, *deletions* $\overline{\mathcal{D}}$ (Fig. 5 (a)) are simulated by deleting elements from the examined map $\widetilde{\mathcal{E}}$. To create *insertions* $\overline{\mathcal{I}}$ (Fig. 5 (b)), we remove the parts of the point cloud being contained in respective bounding shapes. Typically, such procedure leaves residual points from other map elements (e.g., an underlying pole) in close proximity — rendering the detection of insertions a valid task. Thereby, we consider the removal of semantic groups as a whole. Specifically, the removal of a pole from the point cloud leads to the removal of adjacent signs and lights as well. Further, we generate *substitutions* $\overline{\mathcal{S}}$ (Fig. 5 (c)) by interchanging element types in $\widetilde{\mathcal{E}}$ with realistic bounding shapes being created for the artificial element, which is appended to the set of artificial elements $\widetilde{\mathcal{S}}$ (e.g., an element now being a light instead of a sign), while the respective correct element as indicated by the sensor data is appended to the set of GT substitutions $\overline{\mathcal{S}}$ (remaining a sign in the example). Hence, the examined map $\widetilde{\mathcal{E}}$ as input to all methods comprises verifications, insertions, and artificial elements: $\widetilde{\mathcal{E}} = \overline{\mathcal{V}} \cup \overline{\mathcal{I}} \cup \widetilde{\mathcal{S}}$.

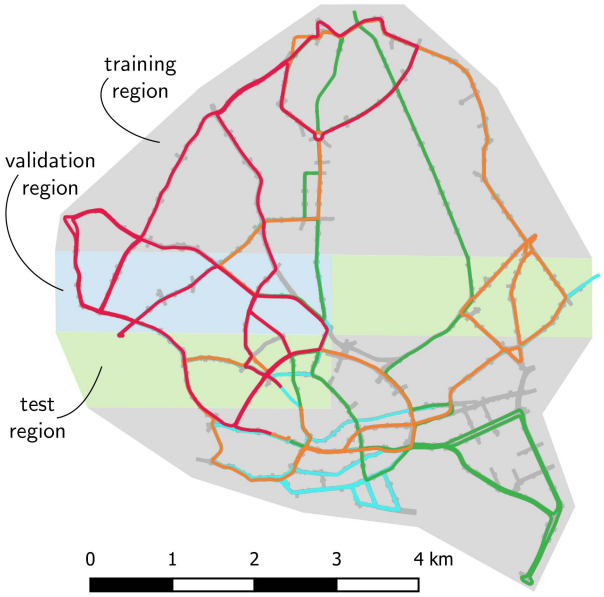


FIGURE 4. HD map and dataset split. We show lanes (gray lines), 4 out of 14 real-world trajectories (colored lines, with the subset of visualized trajectories selected for better visibility), and regions of the map assigned to the training, validation, and test set (gray, blue, and green areas).

B. EVALUATION STRATEGY AND METRICS

To evaluate the performance of the proposed method variants, we associate the predicted set of evaluated map elements $\mathcal{E}^{\text{eval}} = \mathcal{V} \cup \mathcal{D} \cup \mathcal{I} \cup \mathcal{S}$ (cf. Fig. 2) with the respective GT set $\overline{\mathcal{E}}^{\text{eval}} = \overline{\mathcal{V}} \cup \overline{\mathcal{D}} \cup \overline{\mathcal{I}} \cup \overline{\mathcal{S}}$. Thereby, only elements of sets with the same evaluation state (considered as "class") $s = \bar{s} \in \mathcal{S} = \{\texttt{VER}, \texttt{DEL}, \texttt{INS}, \texttt{SUB}\}$ can be associated, e.g., $\mathcal{V}$ to $\overline{\mathcal{V}}$, which is the common approach in the field of object detection (i.e., cars and pedestrians not being associated).

To establish an association between predicted and ground truth elements among sets for which existing hypotheses in the examined set $\widetilde{\mathcal{E}}$ are available (verifications $\mathcal{V}$, insertions $\mathcal{I}$, and substitutions $\mathcal{S}$), an evaluated element's known identifier is utilized to find the element in the respective GT

set. For deletions (elements missing in $\widetilde{\mathcal{E}}$ without identifier), we apply the association criteria as provided in Appendix C. If a predicted element is successfully associated with an element in the respective GT set, the element is considered as *true positive* (TP). If a prediction cannot be associated with a GT element, the detection is considered as *false positive* (FP). Further, a GT element without an associated prediction is counted as *false negative* (FN). We indicate the deviation detection performance for each set (or evaluation state, respectively) individually using the following metrics, with TP and FP being the amount of true and false positives, respectively, and FN being the amount of false negatives:

$$\text{Recall:} \quad \text{RE} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{14}$$

$$\text{Precision:} \quad \text{PR} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{15}$$

$$\text{F}_1\text{ score:} \quad \text{F}_1 = 2 \cdot \frac{\text{PR} \cdot \text{RE}}{\text{PR} + \text{RE}}. \tag{16}$$

Note that false predictions exist for each evaluation state. For instance, residual points in proximity of a GT insertion may lead a method to predict a FP verification instead, causing a (not associated or undetected) FN insertion. To evaluate the regression errors, we employ the L1 and L2 metrics:

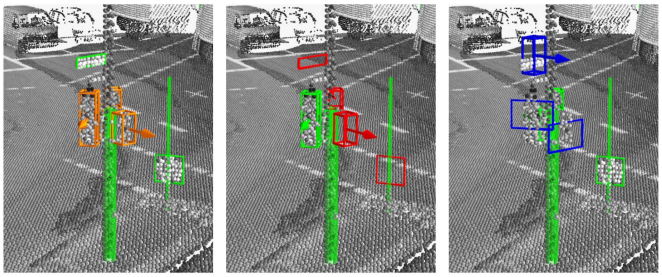
$$\text{L1:} \quad E(q) = \frac{1}{\text{TP}} \sum_{i=1}^{\text{TP}} ||q_i - \overline{q}_i||_1, \tag{17}$$

$$\text{L2:} \quad E(\boldsymbol{p}^{\text{pos}}) = \frac{1}{\text{TP}} \sum_{i=1}^{\text{TP}} \left|\left|\boldsymbol{p}_i^{\text{pos}} - \overline{\boldsymbol{p}}_i^{\text{pos}}\right|\right|_2, \tag{18}$$

with $q \in \{d, w, h, \varphi\}$, and $\boldsymbol{p}^{\text{pos}} = (x^{\text{pos}}, y^{\text{pos}}, z^{\text{pos}})$ being the center position of a predicted bounding shape, and i being the index of the predicted element. Note that the regression errors can only be computed from TP detections, excluding insertion predictions, as no map element in the sensor data is available for regression in their case.

C. TRAINING STRATEGY

Our experimental pipeline is implemented in `PyTorch`. All networks are trained from scratch for 8 epochs using a 2-GPU setup (`Tesla V100`) with a batch size of 2, whereby the network weights are initialized from a normal distribution following [80]. For the point density and partial occlusion ablation studies, we use a batch size of 4 to allow for a reduced training time. We utilize the Adam optimizer with a fixed learning rate of $2 \cdot 10^{-4}$. Furthermore, we use a global augmentation strategy for both point cloud and map elements, applying a random translation $\boldsymbol{t} \in \mathcal{U}^3$, with $\mathcal{U} = [-t^{\text{dim}}, t^{\text{dim}}]$, $\text{dim} \in \{\text{x}, \text{y}, \text{z}\}$, and $(t^{\text{x}}, t^{\text{y}}, t^{\text{z}}) = (1\,\text{m}, 0.2\,\text{m}, 0.2\,\text{m})$, and a random rotation around the z-axis by $\varphi \in \mathcal{U}' = [-20°, 20°]$, with $\mathcal{U}$ denoting a uniform distribution. Additionally, we randomly mirror both point cloud and map along the x-axis with a probability of 0.5. As multi-occurrences of individual map elements in different point cloud samples exist, we randomly remove 20% of all points in each crop to reduce potential overfitting. Further, we observed performance gains for `MDD-MC` and `MDD-M` through increasing deviation probabilities for deviation simulation during training, which saturate at the optimal choice of $(P^{\text{VER}}, P^{\text{DEL}}, P^{\text{INS}}, P^{\text{SUB}}) = (0.5, 0.2, 0.2, 0.1)$ as applied in this work. Moreover, the utilized size of the point cloud crop varies: For our default experiment in Section V-B, we consider a maximum crop extent of 60.8 m, 40 m, and 9.6 m in the x-, y-, and z-dimension, respectively. For the ablation studies, we consider an x-extent of 28.8 m and exclude samples without any map element to allow for a reduced training time, taking into account the large number of experiments conducted.



(a) Deletions (b) Insertions (c) Substitutions

FIGURE 5. Map deviation simulation of (a) deletions (yellow), (b) insertions (red), and (c) substitutions (blue), with verifications depicted in green. Simulated deviations are assumed as ground truth.

V. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we first discuss the threshold optimization on the validation set and then the results on the test set. Last, we conduct ablation studies, simulating degenerated sensor data and inducing sensor and map data misalignments.

A. VALIDATION SET THRESHOLD OPTIMIZATION

To obtain predictions from the anchor grids, all method variants require the selection of thresholds regarding predicted existence likelihoods (methods `MDD-SC` and `MDD-MC`) or evaluation state scores (method `MDD-M`) on the validation set, above which an anchor is considered to provide a valid prediction (see (8) and (13), respectively). Such selection requires an optimization procedure, which we describe in the following. The resulting optimized thresholds are applied later to obtain the test set results. For the method variants `MDD-SC` and `MDD-MC` that are based on object detection, three thresholds $\Theta^{\text{et}}_{\text{score}}$ in (8) with $\text{et} \in \mathcal{T} = \{\text{s}, \text{l}, \text{p}\}$ for signs, lights, and poles have to be selected, which we optimize by maximizing the F_1 score on the validation set for GT elements with a `DEV` evaluation state summarizing all deviating states $\mathcal{S}^{\text{DEV}} = \{\text{DEL}, \text{INS}, \text{SUB}\} \subset \mathcal{S}$. The method variant `MDD-M`, on the other hand, predicts four individual scores for the evaluation states in set $\mathcal{S} = \{\text{VER}, \text{DEL}, \text{INS}, \text{SUB}\}$. This allows for an individual threshold optimization for each evaluation state, which requires the optimization of $3 \times 4 = 12$ thresholds Θ^{et}_{s} in (13) by maximizing the F_1 score on the validation set for each state separately.

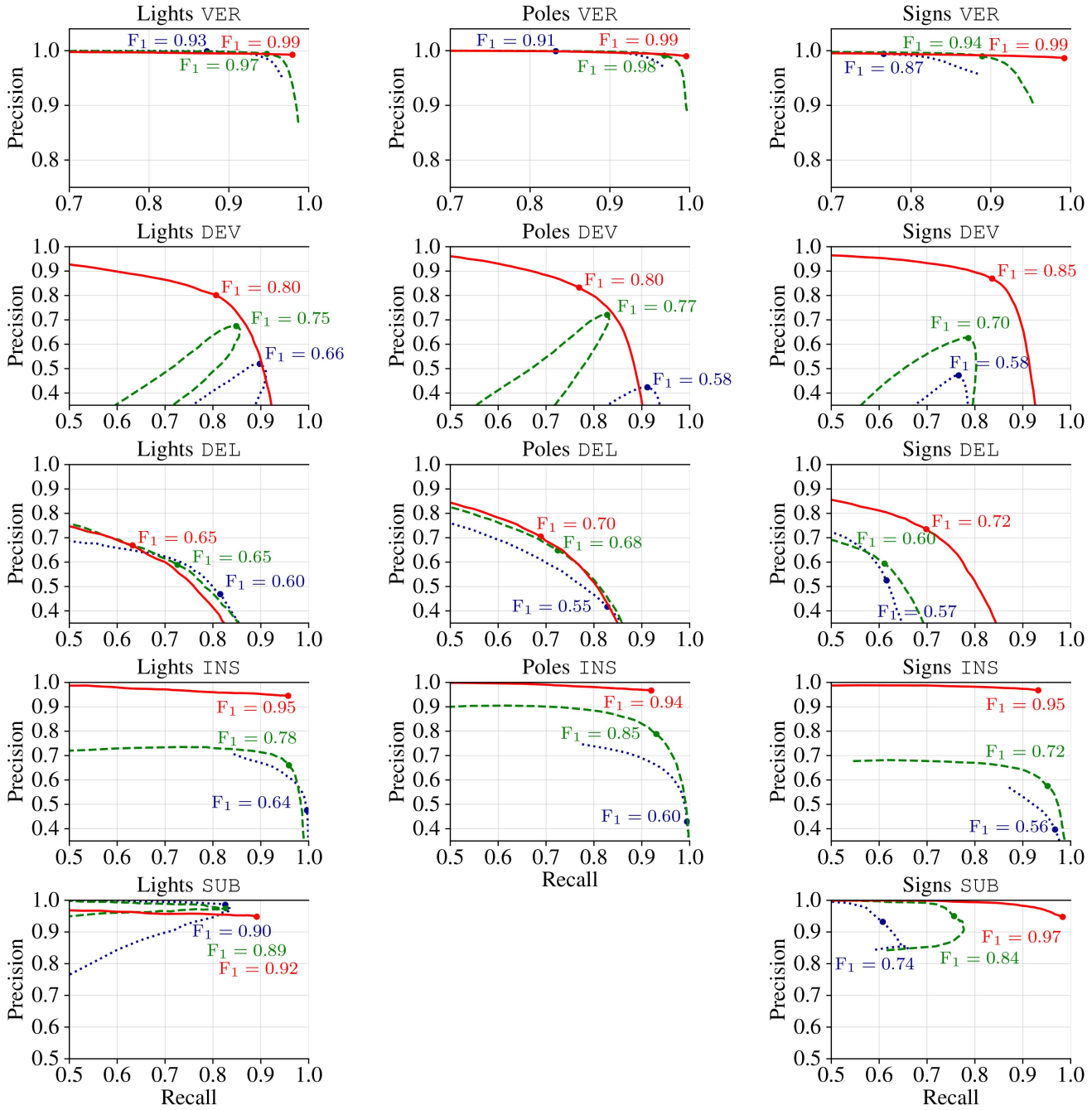


FIGURE 6. Validation set precision-recall curves for lights, poles, and signs comparing MDD-SC (blue dotted), MDD-MC (green dashed), and MDD-M (red solid). The "verified" evaluation state (elements reported as correct) is abbreviated as VER, while DEV denotes the "deviating" state, comprising all elements classified as deletions (DEL), insertions (INS), and substitutions (SUB). For poles, no substitution by other elements exists.

To this end, we create the precision-recall curves for all method variants depicted in Fig. 6 by varying respective threshold values between 0.01 and 0.9 in steps of 0.01. Specifically, Fig. 6 indicates the validation set performance for lights, poles, and signs from left to right, reported separately for each evaluation state (VER, DEV, DEL, INS, and SUB) from top to bottom, with the DEV state summarizing all deviating elements. The performances of MDD-SC, MDD-MC, and MDD-M are depicted as blue dotted, green dashed, and red solid curves, respectively.

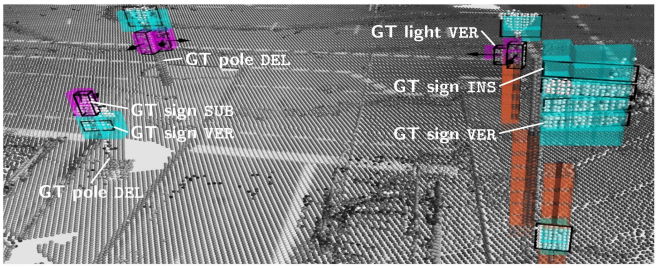
First, the VER performance is quite similar over all three method variants and element types, with MDD-SC showing poorest performance and MDD-M providing the best F_1 score, reaching 0.99 for all element types. However, all methods perform around or quite above an F_1 score of 0.9 at very high precision levels close to 1, showing that FP verifications only occur rarely (towards very low thresholds).

Second, the DEV evaluation state summarizing DEL, INS, and SUB elements shows similar characteristics over

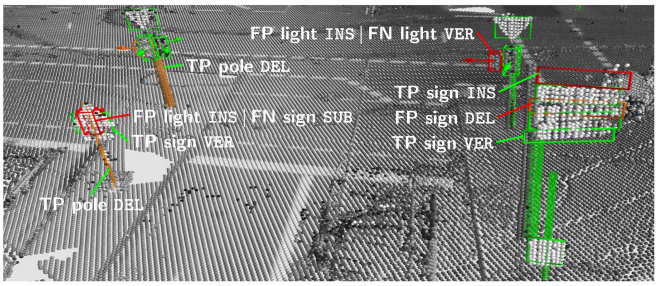
all element types, with MDD-M again performing best with F_1 scores above 0.80, and MDD-SC showing poorest performance ($F_1 \approx 0.60$). The degenerated curves for the object detection-based variants MDD-SC and MDD-MC are due to inverse threshold characteristics of the DEL and INS states. Specifically, higher thresholds lead to fewer object detections, increasing the DEL precision and the INS recall, while decreasing the DEL recall and the INS precision (i.e., causing more FP insertion predictions, also see Fig. 7 for respective examples), and vice versa. Thus, the opposed characteristics diminish the DEV precision towards both low and high thresholds, causing the loop-like precision-recall-curves. In contrast, MDD-M optimizes an individual threshold for each evaluation state, leading to regular DEV curves.

Third, the DEL performance depends on element types. For lights, the advanced methods MDD-M and MDD-MC perform best ($F_1 = 0.65$), with MDD-M operating at higher precision but lower recall than MDD-MC. A similar tendency can be observed for poles, with MDD-M slightly outperforming MDD-MC in this case. Regarding signs, however, MDD-M clearly shows the best performance with an 0.12 absolute F_1 increase compared to MDD-MC. The significant performance gain can be traced to clusters of vertically stacked signs. For the comparison-based method variants MDD-SC and MDD-MC, detected objects are first associated to existing map hypotheses (e.g., yielding verification predictions). Only the remaining unassociated objects are considered as deletions, leaving many deletions undetected, causing a low recall just above 0.6. In contrast, MDD-M separates the evaluation of existing map hypotheses and the prediction of deletions (enabled by the evaluation state classification and exploited by the specialized post-processing), with existing hypotheses being evaluated directly and deletion predictions generated independently. Thus, more (not otherwise associated) deletion predictions are available, boosting the DEL recall. The optimized post-processing also improves the precision in sign clusters: Existing hypotheses cannot be deletions, preventing FP deletion predictions in close proximity (cf. the sign cluster in Fig. 7 (c) and (d)). Note that even without the optimized post-processing, MDD-M still performs best, mirroring the performance characteristics of lights in this case (increased precision, lower recall, and best F_1 score).

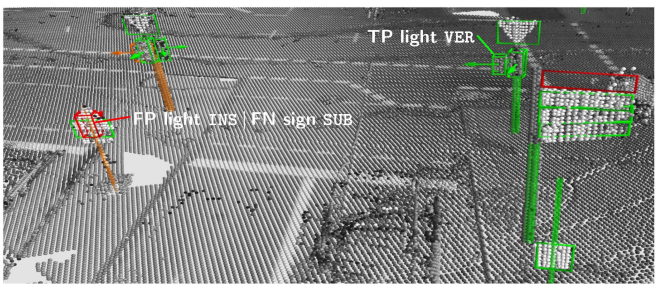
Fourth, the INS performance is similar over all map element types, with MDD-M showing best performance at $F_1 \approx 0.95$, MDD-MC being second place with F_1 scores around 0.80, and MDD-SC showing poorest performance with F_1 scores around 0.60. The performance gains are mostly achieved by an increased precision, or fewer predicted FP insertions, respectively. For the comparison-based variants MDD-SC and MDD-MC, FP insertions are caused by (falsely) undetected objects correctly represented by the map. The additional map input applied in MDD-MC and MDD-M provides the network with initial hypotheses regarding possible element locations, causing both methods to detect objects reliably, if sensor and map data match.



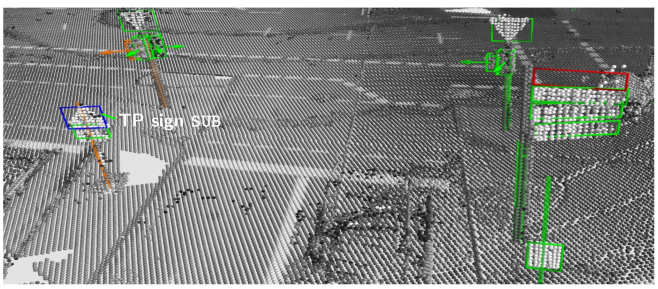
(a) Examined map elements $\widetilde{\mathcal{E}}$, map representation $\boldsymbol{m}'$, and point cloud $\mathcal{P}$



(b) MDD-SC (reference): "Sensor-only object detection DNN"



(c) MDD-MC: "Map-supported object detection DNN"



(d) MDD-M: "Map-supported deviation detection DNN"

FIGURE 7. Example predictions on the test set made by our three method variants in (b)-(d) with used inputs in (a). The colored voxels depict the map representation m' (poles: orange, lights: purple, signs: cyan). The black shapes in (a) depict examined elements $\widetilde{\mathcal{E}}$. For an exemplary subset of elements selected for discussion, we provide ground truth (GT) labels in white. Obtained predictions in (b)-(d) are colored according to their evaluation state (verifications VER: green, insertions INS: red, deletions DEL: yellow, blue: substitutions SUB). Text labels connected in red indicate false predictions regarding the evaluation state (false positive (FP) or false negative (FN)), while the exemplary green connections indicate true positive (TP) predictions. All shapes without text label are TP predictions.

Interestingly, the classification of evaluation states in MDD-M further improves the INS precision, indicating an increased tendency of the network to trust and confirm a map hypothesis, which is also reflected by the slightly decreased INS recall compared to MDD-SC missing the additional map input: In some cases, MDD-M falsely confirms an existing

map hypothesis, e.g., due to residual points or other objects in close proximity.

Last, regarding the SUB performance for lights, all method variants perform at a similar level with F_1 scores around 0.90. For signs, however, MDD-M clearly performs best, achieving $F_1 = 0.97$, which is an absolute improvement of 0.23 compared to MDD-SC. The curves for MDD-SC and MDD-MC appear degenerated in a loop-like fashion as observed for the DEV evaluation state. Also in this case, the inverse threshold characteristics of the DEL and INS states are the root cause, recalling that for both comparison-based methods, substitutions are created during post-processing from deletions and insertions (cf. Section III-C3).

In total, a clear rank order is visible, with MDD-M comprising our specialized MDD network performing best, and the object detection-based variants MDD-MC and MDD-SC being on second and third rank, respectively. *Note the strong verification performance of MDD-M, indicating a greater trust of the network on the map data*, leaving only few elements with an existing hypothesis undetected (high insertion precision) at the cost of generally fewer objects (e.g., lights) predicted if no such hypothesis is available (lower DEL recall but higher DEL precision).

B. TEST SET RESULTS

In the following, we present and discuss the test set results, which are obtained by applying the thresholds optimized on the validation set. Specifically, we compare the method variants MDD-SC, MDD-MC, and MDD-M based on unmodified, high-density point cloud inputs with highly precise spatial alignment with the HD map. First, we present the deviation detection and regression error results. Last, we provide example predictions for each method variant.

1) DEVIATION DETECTION PERFORMANCE

The test set results regarding the deviation detection performance obtained by comparing the predicted set of map elements $\mathcal{E}^{\text{eval}}$ with the respective GT set $\overline{\mathcal{E}}^{\text{eval}}$ are summarized in Table 1, with obtained F_1 score (16), precision (15), and recall (14) results for the method variants MDD-SC, MDD-MC, and MDD-M indicated individually for all element types and possible evaluation states $\mathcal{S} = \{\text{VER}, \text{DEL}, \text{INS}, \text{SUB}\}$ of respective elements in $\mathcal{E}^{\text{eval}}$ (TPs and FPs) or $\overline{\mathcal{E}}^{\text{eval}}$ (FN).

Overall, the test set results in Table 1 mirror the characteristics observed on the validation set discussed in the previous section, with the rank order of MDD-M, MDD-MC, and MDD-SC only changing on a detail level. Specifically, the F_1 score for light deletions is now best for MDD-MC, with MDD-M on second rank. Further, regarding light substitutions, MDD-SC slightly outperforms the other method variants, which, however, all show similar performance with $F_1 = 0.92$ or $F_1 = 0.91$, respectively.

Also on the test set, a clear rank order of methods appears regarding the F_1 *score: The purely object detection-based method MDD-SC is overall the poorest method, with MDD-MC following on second rank, and MDD-M being the best for almost all element types and evaluation states.* On average over all three map element types (lights, poles, signs), MDD-M achieves a strong $F_1 = 0.99$ for map verification, and a good $F_1 = 0.85$ for map deviation detection.

TABLE 1. Test set results for the three method variants MDD-SC, MDD-MC, and MDD-M. The performance is measured using the F_1 score (16), recall (14), and precision (15), reported separately for lights, poles, and signs. Results are indicated individually for possible evaluation (eval.) states (verification VER, deletion DEL, insertion INS, substitution SUB, with the DEV state summarizing elements with DEL, INS, or SUB evaluation states). Best F_1 score is bold, second best is underlined.

Eval. State	F_1			Recall			Precision		
	SC	MC	M	SC	MC	M	SC	MC	M
Lights									
VER	0.95	0.98	**0.99**	0.90	0.96	0.99	1.00	0.99	0.99
DEV	0.71	0.80	**0.87**	0.92	0.89	0.85	0.57	0.73	0.90
├ DEL	0.70	**0.77**	0.75	0.87	0.80	0.71	0.58	0.75	0.80
├ INS	0.66	0.78	**0.96**	1.00	0.97	0.96	0.50	0.65	0.97
└ SUB	**0.92**	0.91	0.91	0.86	0.86	0.90	0.98	0.97	0.91
Poles									
VER	0.93	0.98	**0.99**	0.87	0.98	1.00	1.00	0.99	0.99
DEV	0.62	0.78	**0.81**	0.93	0.85	0.83	0.46	0.72	0.80
├ DEL	0.59	0.69	**0.70**	0.87	0.78	0.74	0.44	0.62	0.66
└ INS	0.65	0.87	**0.94**	0.99	0.92	0.91	0.49	0.83	0.97
Signs									
VER	0.86	0.93	**0.99**	0.75	0.87	0.99	0.99	0.99	0.99
DEV	0.56	0.68	**0.87**	0.75	0.77	0.85	0.45	0.60	0.89
├ DEL	0.53	0.59	**0.71**	0.58	0.59	0.68	0.48	0.60	0.74
├ INS	0.55	0.69	**0.95**	0.97	0.95	0.94	0.38	0.54	0.97
└ SUB	0.72	0.83	**0.97**	0.57	0.73	0.98	0.95	0.96	0.96

TABLE 2. Test set regression error results for the three method variants MDD-SC, MDD-MC, and MDD-M, reported separately for lights, poles, and signs. Errors $E(\cdot)$ for position p^{pos} (18), width w, diameter d, and height h (all (17)) are indicated in centimeters, while the orientation error $E(\varphi)$ (17) is provided in degrees. Best results in bold face, second best underlined.

	Variant	$E(\boldsymbol{p}^{\text{pos}})$	$E(w)$	$E(d)$	$E(h)$	$E(\varphi)$
Lights	MDD-SC	8.0	3.6	-	7.9	14.0
	MDD-MC	6.7	2.8	-	6.3	7.7
	MDD-M	**5.8**	**2.3**	-	**5.3**	**4.4**
Poles	MDD-SC	10.7	-	5.8	-	-
	MDD-MC	6.6	-	2.8	-	-
	MDD-M	**5.0**	-	**2.7**	-	-
Signs	MDD-SC	11.3	9.0	-	11.5	12.3
	MDD-MC	9.8	6.9	-	5.7	5.5
	MDD-M	**7.3**	**5.6**	-	**4.8**	**4.2**

2) REGRESSION PERFORMANCE

The L1 (17) or L2 (18) errors given in centimeter or degree regarding the regression of bounding shapes obtained on the test set are provided in Table 2, indicating the performance

of the method variants MDD-SC, MDD-MC, and MDD-M separately for each element type. As the object detection DNN in MDD-SC operates without additional map input, the reported errors indicate the network's capability to deduce respective bounding shapes from sensor data alone.

Regarding the L2 error of the predicted bounding shape center $E(\boldsymbol{p}^{\text{pos}})$ compared to MDD-SC, best-performing MDD-M provides an absolute improvement of 2.2 cm, 5.7 cm, and 4 cm for lights, poles, and signs, respectively. Further, regarding width and height errors $E(w)$ and $E(h)$, MDD-M w.r.t. MDD-SC overall achieves greater improvements for signs than for lights, i.e., 3.4 cm and 6.7 cm for signs compared to 1.3 cm and 2.6 cm for lights, respectively. The higher error ranges for signs can be attributed to the greater variance of bounding shapes in size. Similar to signs, the diameter error $E(d)$ for poles is reduced almost to half, comparing MDD-M and MDD-SC, with an absolute improvement of 3.1 cm. Regarding the orientation error $E(\varphi)$, MDD-M reduces the obtained error almost to a third compared to MDD-SC, providing an absolute improvement of 9.6° and 8.1° for lights and signs, respectively.

First, the achieved error improvements of advanced method variants (e.g., comparing MDD-MC to MDD-SC) are due to the provision of regression hypotheses by the map representation $\boldsymbol{m}'$. Second, the specialized post-processing of MDD-M averages the regression features of multiple predictions for elements with an existing hypothesis, which yields a refined bounding shape. *Overall, regarding the obtained regression errors, methods rank in the same order as before: MDD-SC, MDD-MC, and MDD-M, with MDD-M consistently achieving lowest errors.*

3) EXAMPLE PREDICTIONS

Fig. 7 provides obtained example predictions $\mathcal{E}^{\text{eval}}$ for all method variants with visualized inputs $\widetilde{\mathcal{E}}$, $\boldsymbol{m}'$ and $\mathcal{P}$ (cf. Fig. 2). Subsequently, we discuss an exemplary subset of predictions to illustrate the performance characteristics discussed in the previous sections.

First, Fig. 7 (a) shows the examined set of elements $\widetilde{\mathcal{E}}$ (black shapes) used to generate the map representation $\boldsymbol{m}'$ (colored voxels). The scene comprises several verifications (map hypotheses matching the sensor data), two deletions (missing poles on the left without map hypothesis), a single sign substitution (with the map falsely indicating a light instead of a sign on the left), and a single insertion (top sign hypothesis on the right without sensor data), with the white labels indicating the ground truth (GT) evaluation states of mentioned examples. If an example element (be it VER, DEL, INS, or SUB) first appears as true positive (TP) (e.g., in Fig. 7 (b)), the respective label is omitted for better clarity in subsequent figures (e.g., in Fig. 7 (c) and (d)), if the example element remains correctly predicted (TP).

Regarding the GT pole deletions, both are correctly detected (TP) by each of the three method variants (indicated by the respective text labels connected in green in Fig. 7 (b), which are omitted in (c) and (d)), reflecting the high DEL recall for poles achieved by all variants (cf. Table 1). However, MDD-SC also provides a deletion prediction not associated with a map hypothesis during comparison, which yields a pole FP deletion (right of Fig. 7 (b)). The FP deletion vanishes for more advanced method variants, corresponding to the increased DEL precision in Table 1.

Further, the GT sign insertion is correctly detected by all variants (green connected "TP sign insertion" label in Fig. 7 (b), omitted in (c) and (d)). However, MDD-SC fails to detect the light object corresponding to the GT light verification (top of Fig. 7 (b)). Thus, during comparison, no object detection is associated to the light hypothesis in $\widetilde{\mathcal{E}}$ (black shape in Fig. 7 (a)), leading to one FP light insertion prediction, and one missed light verification, counted as FN. However, the advanced methods MDD-MC and MDD-M succeed in this case, reflecting the increased VER recall in Table 1. Similar considerations apply regarding the sign substitution: Both MDD-SC and MDD-MC fail to detect the upper of both signs on the left (Fig. 7 (b) and (c)), predicting only a single sign in between both elements, which is associated to the bottom sign hypothesis, leading to a (slightly misplaced) TP sign verification. Therefore, as the sign prediction is not associated to the light hypothesis during comparison, a FP light insertion is predicted, with the sign substitution left undetected (FN). Only MDD-M (Fig. 7 (d)) succeeds in predicting both elements correctly (bottom sign verification and top sign substitution).

In total, we can observe in Fig. 7 that the method variant MDD-M clearly outperforms the comparison-based method variants MDD-SC and MDD-MC, with no false predictions (FPs or FNs) being made in the scene by MDD-M.

C. ABLATION STUDIES

Subsequently, we present results for ablation studies. First, we degenerate the point cloud input by reducing the point density. Second, we induce partial occlusions. Last, to demonstrate the suitability of our approach for onboard use-cases, we induce an artificial misalignment between point cloud and map, which is caused in practice by localization errors of the ego-vehicle on the HD map. Note that driving functions of today's automated vehicles rely on available HD maps as the onboard perception is unable to reliably deduce the stationary environment, especially in bad weather conditions or occlusion scenes. Thus, an approach to MDD is required to at least verify existing map elements reliably.

1) REDUCING POINT DENSITY

We conduct a total of 12 experiments including training and threshold optimization, randomly keeping 100%, 50%, 25%, or 10% of all points for the method variants MDD-SC, MDD-MC, and MDD-M, with the original point density of point clouds in the 3DHD CityScenes dataset [10] being $1/1\,\text{dm}^3$. As the F_1 score (16) results show similar trends for each element type, we depict averaged results in Fig. 8

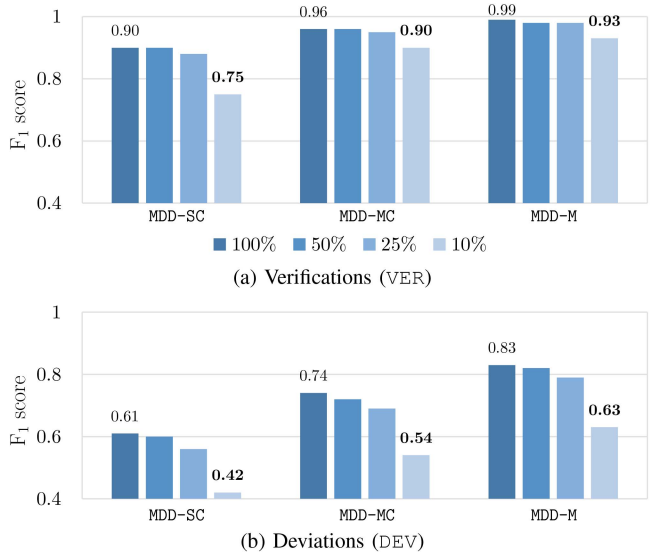


FIGURE 8. Test set results for the point density ablation study obtained by averaging individual results in terms of F_1 score (16) for signs, lights, and poles, reported for verifications (VER) and deviations (DEV). We compare the F_1 performance of all method variants for different point densities, where 100% indicates the full resolution of the point cloud. For the lowest resolution, we keep only 10% of all points.

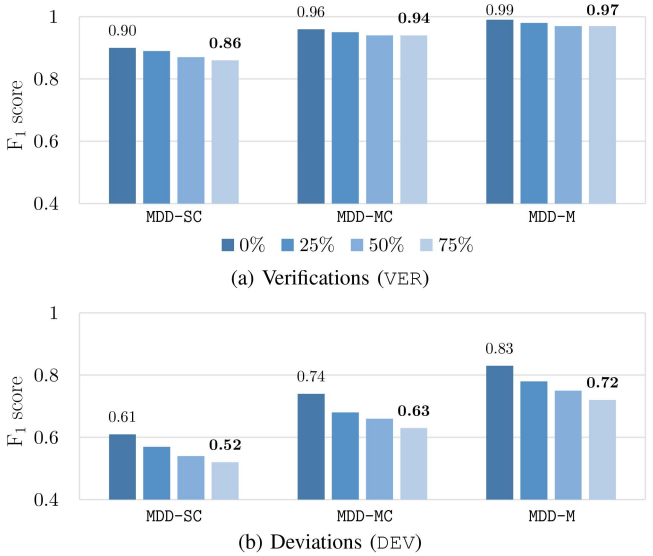


FIGURE 9. Test set results for the partial occlusion ablation study obtained by averaging individual results in terms of F_1 score (16) for signs, lights, and poles, reported for verifications (VER) and deviations (DEV). We compare the F_1 performance of all method variants for different percentages of occluded elements, whereby 0% indicates that no occlusions are present. We (partly) occlude up to 75% of all elements.

over signs, lights, and poles. We differentiate between verifications (VER) and deviations (DEV), the latter summarizing deletions, insertions, and substitutions.

Regarding deviations (DEV), all method variants show only a slight degradation of performance down to 25% point density; only when omitting 90% of the points, significant degradations are observed. Over all reported point densities, MDD-M remains about 21% absolute (F_1) ahead of MDD-SC. Similar observations hold for verifications (VER). Down to 25%, all method variants degrade only slightly in performance. However, *MDD-M turns out to perform very robustly with an* $F_1 > 0.90$ *even if only 10% of the point cloud is available.* The high verification performance at 10% point density of MDD-M demonstrates the network's capability to learn expected sensor data distributions for different map hypotheses internally, comparing (degenerated) sensor data to map hypotheses, while sensor-only MDD-SC frequently fails to detect and verify elements. Hereby, the explicit classification of evaluation states in MDD-M further facilitates distribution learning, shown by the performance gain w.r.t. MDD-MC. Further, the stable performance down to 25% point density for deviations and even to 10% for verifications highlights the *suitability of MDD-M also for low-density onboard LiDAR or even RADAR data.*

2) INDUCING PARTIAL OCCLUSIONS

Also in the case of induced partial occlusions, we conduct a total of 12 experiments including training and threshold optimization, while randomly selecting and occluding 0%, 25%, 50%, or 75% of all map elements that are still present in the sensor data after simulating map deviations, whereby we apply each occlusion setting to MDD-SC, MDD-MC, and MDD-M, respectively. To generate occlusions, we always remove half of the element's points, randomly selecting between left, right, top, or bottom. We average and report the obtained test set results in terms of F_1 score (16) in Fig. 9 in the same fashion as we did in Fig. 8.

Again starting with the deviations (DEV), we observe an almost linear decrease of all methods in the F_1 measure w.r.t. the occlusion ratio. Again, MDD-M turns out to be consistently about 20% absolute (F_1) better than MDD-SC for all occlusion ratios. With respect to verifications (VER), MDD-SC gradually degrades up to −4% absolute, *while MDD-MC and MDD-M show only −2% absolute degradation for up to 75% occlusion*, highlighting the robustness of both variants against scenes with dense occlusions.

3) INDUCING MISALIGNMENT

In the previously presented studies, point cloud and map data feature a highly precise data alignment. In practice, however, the ego-poses on the map obtained from an onboard localization are erroneous, causing a shift between sensor and map data. Also, laser scanner and ego-localization (providing the geo-location and orientation to register sensor and map data) produce their respective measurements at different timestamps, while the scanner typically rotates, which further reduces timely synchronization. While the last two effects can be compensated using odometry data, the imperfect ego-localization remains. As respective algorithms are well developed, however, localization errors seen in practical urban scenarios are indeed very small, e.g., being approx. 10 cm in translation and 0.1° in rotation. For our study, we consider the default experiment for MDD-M from Section V-B1. During test inference, we induce a misalignment in x-, y-, and z-direction between 0 cm and 60 cm in

steps of 10 cm. Rotational errors only cause a translation for larger distances (e.g., approx. 10 cm given a 60 m distance and 0.1° rotational error), which we therefore omit.

To induce misalignment, we shift all map elements in $\widetilde{\mathcal{E}}$ by applying a translation of constant magnitude with the direction randomly selected in spherical coordinates using a uniform distribution, while leaving the GT elements unchanged. The obtained results in terms of F_1 score (16) are averaged over all element types and reported in Fig. 10 separately for each evaluation state, with the gray curves depicting the performance without applying any countermeasures against misalignment. As expected, all MDD-M F_1 curves monotonically decrease with increasing map-sensor misalignment, whereby we observe steepest performance drop for INS starting at 20 cm misalignment. This performance drop is mainly caused by an increasing number of insertion FPs as the network misclassifies actual verifications due the sensor data shifted out of scope. To further increase misalignment robustness, we retrain MDD-M for another 4 epochs while inducing a constant misalignment error of 30 cm during training. The obtained performance during test inference is indicated by the colored curves in Fig. 10. It is clearly visible that the network learns to compensate misalignment errors. The performance drops for all states are less pronounced, i.e., achieving a +0.15 absolute F_1 increase for INS at 30 cm in Fig. 10. *Thus, for all evaluation states, our proposed MDD-M shows a robust F_1 performance even up to 20 cm map-sensor misalignment* when employing the proposed countermeasure.

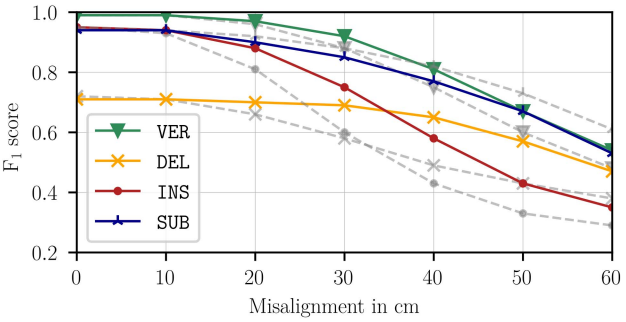


FIGURE 10. Test set results of MDD-M for the misalignment ablation study inducing a translation between sensor and map. The depicted F_1 score is averaged for signs, lights, and poles. We provide F_1 scores for verifications (VER, green), deletions (DEL, yellow), insertions (INS, red), and substitutions (SUB, blue). Each colored curve is accompanied by a gray counterpart depicting the F_1 performance without countermeasures.

D. BENCHMARKS

As previously stated in Section I, our deviation annotations for 3DHD CityScenes [10] published along with our entire MDD pipeline comprising code for training, inference, and evaluation allow for benchmarking MDD methods. To the best of our knowledge, no other MDD methods for signs, lights, and poles comparable to ours are publicly available, preventing direct 1:1 comparisons. However, our MDD method, be it MDD-SC, MDD-MC, or MDD-M as variants, can be combined with other DNN architectures for 3D object detection in the field. Especially MDD-SC is easily combinable with such architectures, e.g., PointPillars [71] or VoxelNet [72], requiring only an adaptation of the network heads to predict map elements instead of road users. Hence, we compare PointPillars [71], VoxelNet [72], and our 3DHDNet [10] as employed architectures for MDD-SC to our specialized MDD network in MDD-M, incorporating our proposed MDD novelties, i.e., the additional map input and the classification of evaluation states. Note that both MDD-MC and MDD-M principally generalize to other architectures as well, e.g., point-based networks [74], [75], given a modified map injection procedure.

All compared methods are trained[1] using the strategies described in Section IV-C with the larger 60.8 m crop extent in the x-dimension, and the threshold optimization procedure from Section V-A. For PointPillars, we use a smaller grid cell size of $s^{x}_{vox} \times s^{y}_{vox} = s^{2}_{vox}$ with $s_{vox} = 20$ cm for discretizing the point cloud into the 2D bird's eye view (BEV) used as network input, while using $s^{x}_{vox} \times s^{y}_{vox} \times s^{z}_{vox} = 20\,\text{cm} \times 20\,\text{cm} \times 40\,\text{cm}$ for the 3D voxel grid input to VoxelNet, exactly as in the original paper. Recall that 3DHDNet employs large cubic voxels of size s^{3}_{vox} with $s_{vox} = 40$ cm to compensate for the computationally expensive 3D convolutions applied in the 3D backbone [10].

Table 3 summarizes the F_1 score performance (16) for the detection of verifications VER and deviations DEV, and the L1 (17) or L2 (18) errors measuring the regression of bounding shapes, all obtained on the test set, with the DEV state summarizing deletions DEL, insertions INS, and substitutions SUB. Comparing PointPillars, VoxelNet, and 3DHDNet as employed architectures for MDD-SC, 3DHDNet clearly performs best regarding all measures, whereby the performance boost relative to VoxelNet is most significant for signs, being +5% and +7% absolute F_1 for VER and DEV, respectively. In comparison, the F_1 performance gains for VER and DEV w.r.t. lights and poles are smaller, reaching between +1% and +4%, which is expected as only signs appear as vertically stacked objects, being the design focus of 3DHDNet. Similar observations hold in terms of regression performance. For instance, 3DHDNet reduces the orientation error by −7.8° and −2.2° for signs and lights, respectively. PointPillars achieves an unacceptable performance for all measures. *Here, the 2D BEV applied for discretization poses an information bottleneck regarding the vertical dimension. Further, our specialized MDD network in MDD-M proposed in this work outperforms all other MDD methods based on object detection, showing the effectiveness of both map input and evaluation state classification.* Relative to 3DHDNet, MDD-M increases VER and DEV performance by +13% and +31% absolute F_1, respectively, and reduces regression errors significantly.

Considering the average runtime during test inference, PointPillars requires 206 ms + 54 ms = 260 ms, adding

[1]The results for 3DHDNet [10] (MDD-SC) and MDD-M in Table 3 are taken from Table 1 and Table 2 to provide an easy comparison at a glance.

TABLE 3. Test set results for benchmarking different MDD methods. We compare `PointPillars` [71] (PointP), `VoxelNet` [72], and `3DHDNet` [10] employed for `MDD-SC` to our specialized MDD network in `MDD-M`. Results are reported separately for lights, poles, and signs. The detection performance regarding verifications `VER` and deviations `DEV` is measured using the F_1 score (16), with `DEV` summarizing deletions `DEL`, insertions `INS`, and substitutions `SUB`. Regression errors $E(\cdot)$ for position p^{pos} (18), width w, diameter d, and height h (all (17)) are indicated in centimeters, while the orientation error $E(\varphi)$ (17) is provided in degrees. Best results in bold face, second best underlined.

	Architecture	VER	DEV	$E(\boldsymbol{p}^{\text{pos}})$	$E(w)$	$E(d)$	$E(h)$	$E(\varphi)$
Lights	PointP [71]	0.85	0.46	15.3	3.9	-	13.4	22.0
	VoxelNet [72]	0.94	0.69	9.2	3.6	-	10.5	16.2
	3DHDNet [10]	<u>0.95</u>	<u>0.71</u>	<u>8.0</u>	<u>3.6</u>	-	<u>7.9</u>	<u>14.0</u>
	MDD-M	**0.99**	**0.87**	**5.8**	**2.3**	-	**5.3**	**4.4**
Poles	PointP [71]	0.76	0.38	14.2	-	6.7	-	-
	VoxelNet [72]	0.91	0.58	14.1	-	5.6	-	-
	3DHDNet [10]	<u>0.93</u>	<u>0.62</u>	<u>10.7</u>	-	<u>5.8</u>	-	-
	MDD-M	**0.99**	**0.81**	**5.0**	-	**2.7**	-	-
Signs	PointP [71]	0.62	0.26	19.1	11.4	-	19.2	19.8
	VoxelNet [72]	0.81	0.49	13.1	9.5	-	13.4	20.1
	3DHDNet [10]	<u>0.86</u>	<u>0.56</u>	<u>11.3</u>	<u>9.0</u>	-	<u>11.5</u>	<u>12.3</u>
	MDD-M	**0.99**	**0.87**	**7.3**	**5.6**	-	**4.8**	**4.2**

the time for GPU data transfer and network execution, compared to `VoxelNet` with 210 ms + 203 ms = 413 ms, `3DHDNet` with 30 ms+295 ms = 325 ms, and `MDD-M` incorporating the map with 43 ms + 313 ms = 356 ms. Regarding network execution time, `MDD-M` is a factor 6 slower than fastest `PointPillars`, while the application of larger voxels reduces data transfer time drastically from 206 ms to 43 ms. Using an optimized implementation with `TensorRT` and low-density onboard LiDAR, `PointPillars` achieves a network execution time of 16 ms in [71], translating to a runtime estimation for `MDD-M` of 16 ms · 6 = 96 ms regarding onboard use cases, being at the edge of real-time capability, considering the typical onboard scan duration of 100 ms. However, sparse convolutions can reduce the required runtime for 3D convolution required for `3DHDNet` by factor 3 [73], further supporting real-time capability.

VI. CONCLUSION

In this article, we introduce a novel deep learning-based approach to map deviation detection (MDD) in high-definition (HD) maps and LiDAR data, utilizing the map as additional input to a neural network. To this end, we propose a specialized MDD network that verifies individual map elements (i.e., signs, lights, and poles) or detects and specifies respective map deviations. We compare our method to variants relying on ordinary object detection, showing the superior performance of our MDD network. Specifically, our approach achieves a strong verification and a good deviation detection performance of $F_1 = 0.99$ and $F_1 = 0.85$, averaged over all element types. Furthermore, our ablation studies degenerating the sensor input to simulate bad weather and partial occlusions show that our network maintains its verification performance with $F_1 = 0.93$ on average, despite 90% of all measurements being removed, which may allow for a continued and safe operation of the driving function even in challenging conditions. Also, our approach is robust against misalignment of sensor and map data as seen in practice, highlighting the suitability for onboard applications.

APPENDIX

In this section, we provide additional in-depth information for the interested reader.

A. NETWORK ARCHITECTURE

Subsequently, we detail the inner operation of the encoder and 3D backbone stages applied in the multitask extension of our earlier `3DHDNet` [10] as mentioned in Section III-C1. Note that encoder and backbone are unchanged w.r.t. [10], while the concatenation of the LiDAR feature map $\boldsymbol{m}$ and map representation $\boldsymbol{m}'$ is novel. The network architecture visualized in Fig. 11 implements the object detection DNN in Fig. 2 used for `MDD-SC` and `MDD-MC`.

The *encoder* stage learns an optimal feature representation for the point cloud. In the first encoding stage, each point $\boldsymbol{m}_{n,k}$ of the LiDAR feature map $\boldsymbol{m}$ is initially mapped to $L = 128$ features using a 2D convolution with a $1 \times 1 \times 10$ kernel, which yields $\boldsymbol{c}_1 \in \mathbb{R}^{N \times K \times L}$. Subsequently, the maximum for each feature among all K points in a voxel n is taken to obtain $\boldsymbol{c}_1' \in \mathbb{R}^{N \times 1 \times L}$. This maximum feature vector of length L is then repeated K times to match dimensions with $\boldsymbol{c}_1$, yielding $\boldsymbol{c}_1'' \in \mathbb{R}^{N \times K \times L}$. Each point contained in $\boldsymbol{c}_1$ is concatenated with the maximum feature vector previously obtained for each voxel n, which provides the input with $2L = 256$ features to the second encoder stage. After first mapping points to $L' = 256$ features, the subsequent maximum operation yields the final encoding $\boldsymbol{c}_2 \in \mathbb{R}^{N \times 1 \times L'}$ for all points contained in each voxel n. In a last step, the N obtained feature vectors for each voxel are scattered back to their original position in the 3D voxel grid with zero-padding applied to empty voxels, which yields the encoded LiDAR grid $\boldsymbol{g} \in \mathbb{R}^{N^x \times N^y \times N^z \times L'}$.

The *3D backbone* stage processes $\boldsymbol{g}$ and comprises a downstream and an upstream network, utilizing 3D (transposed) convolutions in both networks on the 3D voxel grid, which allows for the individual detection of vertically stacked traffic signs, as we showed in our previous work [10]. The downstream network is composed of three blocks ("Dn" in Fig. 11), comprising 3, 5, and 5 layers, respectively. Each layer is composed of one convolution, one batch normalization, and one ReLU activation. For instance, Dn1 applies a 3D convolution with a $3 \times 3 \times 3 \times L'$ kernel on $\boldsymbol{g}$. Upsampling blocks are each composed of one transposed convolution, followed by one batch normalization and one ReLU activation. Using the upstream network, feature maps obtained at different scales by the downstream network are upsampled to the original grid size, and are subsequently concatenated to obtain $\widetilde{\boldsymbol{g}} \in \mathbb{R}^{N^x \times N^y \times N^z \times L'''}$ as input to the multitask heads in Fig. 3 from Section III-C1, with $L''' = 3L'' = 768$ and $L'' = 256$. Note that the input concatenation of the additional map feature map input $\boldsymbol{m}'$ with $\boldsymbol{g}$ is only active for `MDD-MC` and `MDD-M`.

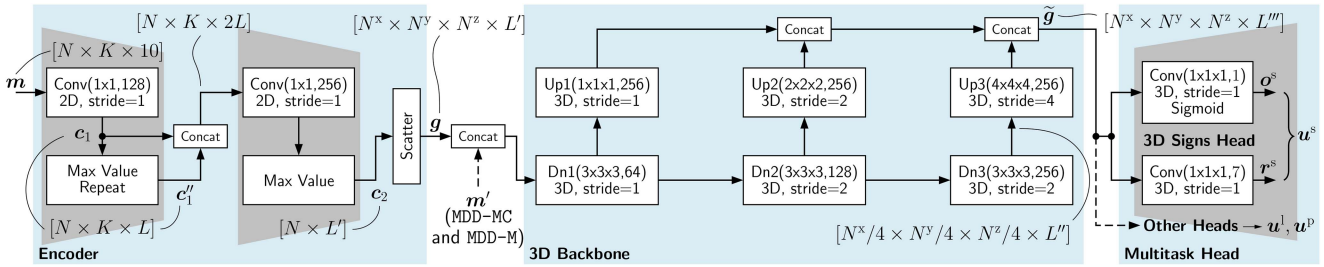
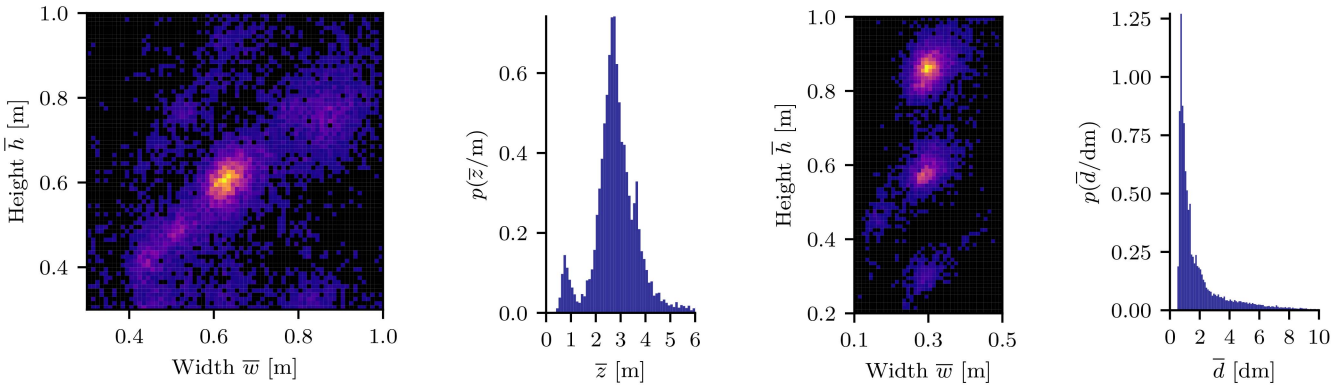


FIGURE 11. Details of the object detection DNN (Fig. 2 (a) and (b)) as a `3DHDNet` multitask architecture. Blue: Processing stages. Gray: Encoder and decoder stages. A 2D convolution ("2D") Conv(1x1, F) with input dimension $[N \times K \times 10]$ uses F kernels of size $1 \times 1 \times 10$, while a 3D convolution ("3D") Conv(1x1x1, F) with $[N^x \times N^y \times N^z \times L''']$-dim. input uses $1 \times 1 \times 1 \times L'''$-sized kernels. "Concat" indicates a concatenation. Downsampling (Dn) blocks comprise multiple convolutional layers.



(a) Sign bounding rectangle (b) Sign $\overline{z}$ position (c) Light bounding box (d) Pole diameter

FIGURE 12. Ground truth distributions for bounding shape parameters $\overline{(\cdot)}$ for signs, lights, and poles, with the overline denoting ground truth. Brighter colors indicate higher occurrence. The $\overline{z}$-position of signs is encoded as height above ground level.

B. ANCHOR DESIGNS

In the following, we describe the anchor design for signs, lights, and poles used to normalize the regression head outputs $\boldsymbol{r}_g^{\mathrm{s}}$, $\boldsymbol{r}_g^{\mathrm{l}}$, and $\boldsymbol{r}_g^{\mathrm{p}}$, respectively. We design our anchors featuring positions p_g^{A} (1) and sizes ℓ_g^{A} (2) based on the ground truth (GT) distributions for respective bounding shape parameters shown in Fig. 12 (a)-(d).

Regarding signs, the ground truth distribution for bounding rectangle height and width in Fig. 12 (a) features peaks around $\overline{h}, \overline{w} \approx 0.65\,\mathrm{m}$ (with the overline indicating GT), which we use as default anchor size $\ell_g^{\mathrm{A}} \in \{w_g^{\mathrm{A}}, h_g^{\mathrm{A}}\}$ in (2). Moreover, most signs are sized above 0.4 m. Hence, we use $s_{\mathrm{vox}} = 0.4\,\mathrm{m}$ as voxel size for the anchor grid. Also, the histogram in Fig. 12 (b) shows that most signs feature a height above ground $\overline{z} < 6\,\mathrm{m}$, which is covered by the z-extent of point cloud crops used as network input (see Section IV-A). For lights, the distribution for bounding box height $\overline{h}$ and width $\overline{w}$ of the (square-sized) base plate in Fig. 12 (c) peaks around $\overline{w} \approx 0.3\,\mathrm{m}$, with three peaks occurring at $\overline{h} \approx 0.3\,\mathrm{m}$, $\overline{h} \approx 0.6\,\mathrm{m}$, and $\overline{h} \approx 0.9\,\mathrm{m}$, due to the varying number of stacked lights in traffic light boxes. As we only apply single-sized anchors, we select the highest peak $\overline{h} \approx 0.9\,\mathrm{m}$, which yields $w_g^{\mathrm{A}} = 0.3\,\mathrm{m}$ and $h_g^{\mathrm{A}} = 0.9\,\mathrm{m}$ used as anchor size in (2). Regarding the pole diameter distribution in Fig. 12 (d), no clear peak can be identified. Thus, we use the mean of all diameters as anchor size, which yields $d_g^{\mathrm{A}} = 0.2\,\mathrm{m}$ for (2).

C. MATCHING STRATEGIES, ASSOCIATION METRICS, AND CRITERIA

For target generation during training, comparison of map elements, and performance evaluation, measuring the overlap of map elements is required. To this end, we employ element-type-specific matching strategies (being algorithmic procedures) visualized in Fig. 13, as well as association metrics that measure the overlap of anchors and GT objects. Also, we define association criteria as thresholds for respective metrics that have to be met for a successful association.

During training, GT objects are matched to the anchor grid to provide the targets $\overline{o}_g$ and $\overline{\boldsymbol{r}}_g$ for the respective network outputs. Such strategies, metrics, and criteria also apply during the comparison (cf. Fig. 2) of predicted elements $\widehat{\mathcal{E}}$ with the examined map $\widetilde{\mathcal{E}}$ to identify deviations, and during performance evaluation when comparing the (predicted) set of evaluated elements $\mathcal{E}^{\mathrm{eval}} = \mathcal{V} \cup \mathcal{D} \cup \mathcal{I} \cup \mathcal{S}$ with the respective GT set $\overline{\mathcal{E}}^{\mathrm{eval}}$.

In general, a single GT element can potentially match with multiple anchors. If an anchor contained in voxel g matches with the GT element, we set the anchor's object detection target $\overline{o}_g^{(\cdot)} = 1$ and the regression target $\overline{\boldsymbol{r}}_g^{(\cdot)}$ according to (1), (2), and (3) using respective ground truth shape parameters. On the other hand, in case of a mismatch, we set $\overline{o}_g^{(\cdot)} = 0$ with $\overline{\boldsymbol{r}}_g^{(\cdot)}$ being "don't care", as regression features

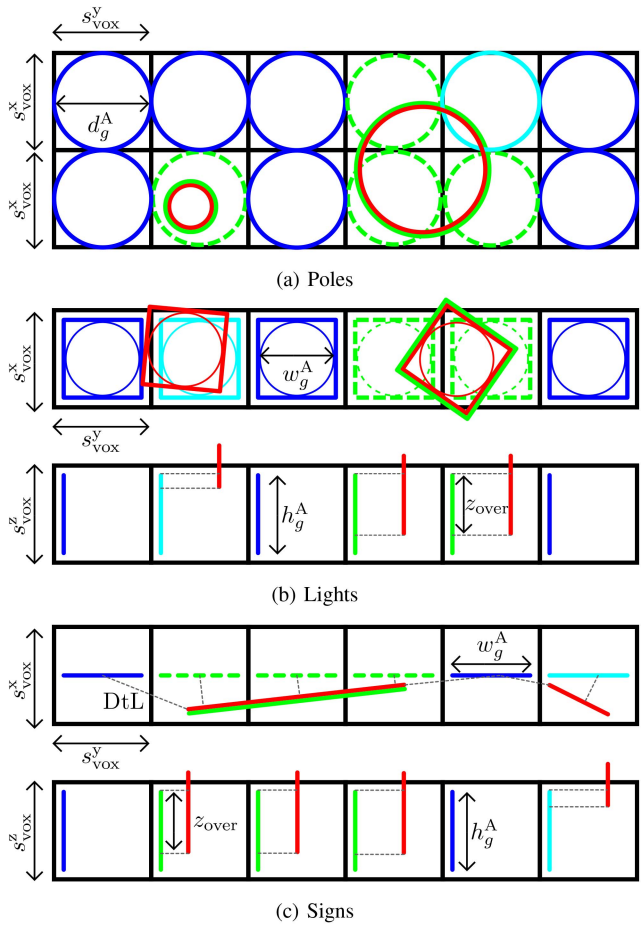


FIGURE 13. Strategies for matching pole, light, and sign ground truth objects (red) to object anchors (blue). For poles, only the x-y-plane is visualized, while for lights and signs the second row visualizes the vertical z-dimension. Matching anchors (dashed) have their target anchor offsets r_s set to match the ground truth object's bounding shape parameters (visualized as overlapping solid green shapes). Anchors with the "don't care" state (cyan) feature an insufficient overlap regarding at least one association metric.

are not considered during loss computation for mismatches (cf. (6)). For all elements, to ensure that a GT element is matched with at least one anchor, we consider the anchor with the closest Euclidean distance d_{E} to the GT element as match, independently from further association criteria.

Regarding *poles*, we employ the matching strategy visualized in Fig. 13 (a). To measure the overlap with anchors surrounding the ground truth element, we define the "Intersection over Smaller Area" (IoSA) metric as the intersecting area of the two circles in the x-y-plane (given by a respective anchor and the ground truth cylinder diameter), indicated relative to the smaller of both circle areas.[2] We require IoSA > 0.2 as association criterion to consider surrounding anchors as matches (see the three green dashed matching anchors in Fig. 13 (a)). Otherwise, if IoSA $= 0$, the respective anchor is considered as mismatch (blue). Moreover, for anchors having an insufficient overlap with the GT element with $0 <$ IoSA ≤ 0.2, we allow a "don't care" state (cyan) during training, in which an anchor is not considered for loss computation. Note that for element comparison (cf. Fig. 2) and performance evaluation, we rely solely on the Euclidean distance d_{E} as association metric, which has proven to be more robust than the additional usage of IoSA. For both comparison and evaluation, we require $d_{\mathrm{E}}/s_{\mathrm{vox}} \leq 0.75$ as criterion for a successful association.

For *lights*, we apply a two-stage matching strategy visualized in Fig. 13 (b). First, we only consider the x-y-plane to find possible matching candidates. To this end, we approximate the base plate of a light's bounding box as a circle with diameter $\overline{w}$ to reuse the IoSA metric defined previously, whereby we require IoSA > 0.05 as association criterion for a match.

Second, we consider the z-dimension for obtained matching candidates, visualized in the second row of Fig. 13 (b). To measure the overlap with anchors in the vertical z-dimension, we define the "Vertical Overlap" (VO) metric using the overlap of line segments z_{over} given by respective z_{min} and z_{max} values of anchor and GT element, normalized by the smaller of both segment lengths Δz as VO $= z_{\mathrm{over}}/\Delta z$. An anchor is considered as final match if VO ≥ 0.2, and is discarded as mismatch when VO ≤ 0.1. Furthermore, the "don't care" state applies for loss computation if $0.1 <$ VO < 0.2.

A similar two-stage matching strategy is employed for *signs* as shown in Fig. 13 (c). First, to identify matching candidates in the x-y-plane, we define the "Distance to Line" (DtL) metric, which indicates the shortest Euclidean distance between the line segment of the GT rectangle (defined by the edge points), and an anchor's center point. Here, we require $\mathrm{DtL}/s_{\mathrm{vox}} < 0.5$ as criterion for a match. Second, we apply the VO thresholds in the same way as described for lights.

[2]The IoSA measure provides a maximum value of 1.0 if the anchor completely encloses the ground truth element or vice versa.

REFERENCES

[1] T. Fingscheidt, H. Gottschalk, and S. Houben, *Deep Neural Networks and Data for Automated Driving: Robustness, Uncertainty Quantification, and Insights Towards Safety*. Cham, Switzerland: Springer, 2022. [Online]. Available: https://library.oapen.org/handle/20.500.12657/57375

[2] C. Plachetka, N. Maier, J. Fricke, J.-A. Termöhlen, and T. Fingscheidt, "Terminology and analysis of map deviations in urban domains: Towards dependability for HD maps in automated vehicles," in *Proc. Intell. Veh. Symp. Workshops*, Oct. 2020, pp. 63–70.

[3] O. Hartmann, M. Gabb, F. Schüle, R. Schweiger, and K. Dietmayer, "Robust and real-time multi-cue map verification for the road ahead," in *Proc. ITSC*, Qingdao, China, Oct. 2014, pp. 894–899.

[4] C. Zinoune, P. Bonnifait, and J. Ibañez-Guzmán, "Detection of missing roundabouts in maps for driving assistance systems," in *Proc. Intell. Veh. Symp.*, Madrid, Spain, Aug. 2012, pp. 123–128.

[5] W. Zhang, Z. Mi, Y. Zheng, Q. Gao, and W. Li, "Road marking segmentation based on Siamese attention module and maximum stable external region," *IEEE Access*, vol. 7, pp. 143710–143720, 2019.

[6] J. Müller and K. Dietmayer, "Detecting traffic lights by single shot detection," in *Proc. ITSC*, Maui, HI, USA, Nov. 2018, pp. 266–273.

[7] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang, "Robust lane detection from continuous driving scenes using deep neural networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 41–54, Jan. 2020.

[8] D. Temel, M.-H. Chen, and G. AlRegib, "Traffic sign detection under challenging conditions: A deeper look into performance variations and spectral characteristics," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3663–3673, Sep. 2020.

[9] M. Raaijmakers, "Towards environment perception for highly automated driving: With a case study on roundabouts," Ph.D. dissertation, Dept. Comput. Sci., Tech. Univ. Eindhoven, Eindhoven, The Netherlands, Jun. 2017.

[10] C. Plachetka, B. Sertolli, J. Fricke, M. Klingner, and T. Fingscheidt, "3DHD CityScenes: High-definition maps in high-density point clouds," in *Proc. ITSC*, Macau, China, Oct. 2022, pp. 627–634.

[11] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *Proc. IROS*, Madrid, Spain, Oct. 2018, pp. 1–8.

[12] Y. H. Khalil and H. T. Mouftah, "LiCaNet: Further enhancement of joint perception and motion prediction based on multi-modal fusion," *IEEE Open J. Intell. Transp. Syst.*, vol. 3, pp. 222–235, 2022.

[13] A. Prakash, K. Chitta, and A. Geiger, "Multi-modal fusion transformer for end-to-end autonomous driving," in *Proc. CVPR*, Nashville, TN, USA, Jun. 2021, pp. 7073–7083.

[14] Z. Bao, S. Hossain, H. Lang, and X. Lin, "High-definition map generation technologies for autonomous driving," 2022, *arxiv.2206.05400*.

[15] V. Ilci and C. Toth, "High definition 3D map creation using GNSS/IMU/LiDAR sensor integration to support autonomous vehicle navigation," *Sensors*, vol. 20, no. 3, p. 899, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/3/899

[16] S. Chen, B. Liu, C. Feng, C. Vallespi-Gonzalez, and C. Wellington, "3D point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception," *IEEE Signal Process. Mag.*, vol. 38, no. 1, pp. 68–86, Jan. 2021.

[17] Z. Jian, S. Zhang, S. Chen, X. Lv, and N. Zheng, "High-definition map combined local motion planning and obstacle avoidance for autonomous driving," in *Proc. Intell. Veh. Symp.*, Paris, France, Jun. 2019, pp. 2180–2186.

[18] D. Wilbers, C. Merfels, and C. Stachniss, "Localization with sliding window factor graphs on third-party maps for automated driving," in *Proc. ICRA*, Montreal, QC, Canada, May 2019, pp. 5951–5957.

[19] C. Plachetka, J. Fricke, M. Klingner, and T. Fingscheidt, "DNN-based recognition of pole-like objects in LiDAR point clouds," in *Proc. ITSC*, Indianapolis, IN, USA, Sep. 2021, pp. 2889–2896.

[20] M. Tan, B. Wang, Z. Wu, J. Wang, and G. Pan, "Weakly supervised metric learning for traffic sign recognition in a LiDAR-equipped vehicle," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 5, pp. 1415–1427, May 2016.

[21] Á. Arcos-García, M. Soilán, J. A. Álvarez-García, and B. Riveiro, "Exploiting synergies of mobile mapping sensors and deep learning for traffic sign recognition systems," *J. Exp. Syst. Appl.*, vol. 89, pp. 286–295, Dec. 2017.

[22] H. Guan, W. Yan, Y. Yu, L. Zhong, and D. Li, "Robust traffic-sign detection and classification using mobile LiDAR data with digital images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 5, pp. 1715–1724, May 2018.

[23] Y. Yu, J. Li, H. Guan, and C. Wang, "Automated extraction of urban road facilities using mobile laser scanning data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 2167–2181, Aug. 2015.

[24] P. Huang, M. Cheng, Y. Chen, H. Luo, C. Wang, and J. Li, "Traffic sign occlusion detection using mobile laser scanning point clouds," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2364–2376, Sep. 2017.

[25] C. Wen et al., "Spatial-related traffic sign inspection for inventory purposes using mobile laser scanning data," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 1, pp. 27–37, Jan. 2016.

[26] B. Riveiro, L. Díaz-Vilariño, B. Conde-Carnero, M. Soilán, and P. Arias, "Automatic segmentation and shape-based classification of retro-reflective traffic signs from mobile LiDAR data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 1, pp. 295–303, Jan. 2016.

[27] Y. Yu, J. Li, C. Wen, H. Guan, H. Luo, and C. Wang, "Bag-of-visual-phrases and hierarchical deep models for traffic sign detection and recognition in mobile laser scanning data," *ISPRS J. Photogrammetry Remote Sens.*, vol. 113, pp. 106–123, Mar. 2016.

[28] M. Atif, A. Ceccarelli, T. Zoppi, M. Gharib, and A. Bondavalli, "Robust traffic sign recognition against camera failures," *IEEE Open J. Intell. Transp. Syst.*, vol. 3, pp. 709–722, 2022.

[29] L. Hacker and J. Seewig, "Insufficiency-driven DNN error detection in the context of SOTIF on traffic sign recognition use case," *IEEE Open J. Intell. Transp. Syst.*, vol. 4, pp. 58–70, 2023.

[30] L. Li, Y. Li, and D. Li, "A method based on an adaptive radius cylinder model for detecting pole-like objects in mobile laser scanning data," *Remote Sens. Lett.*, vol. 7, no. 3, pp. 249–258, 2016.

[31] Z. Kang, J. Yang, R. Zhong, Y. Wu, Z. Shi, and R. Lindenbergh, "Voxel-based extraction and classification of 3-D pole-like objects from mobile LiDAR point cloud data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 11, pp. 4287–4298, Nov. 2018.

[32] Z. Shi, Z. Kang, Y. Lin, Y. Liu, and W. Chen, "Automatic recognition of pole-like objects from mobile laser scanning point clouds," *Remote Sens.*, vol. 10, no. 12, pp. 1891–1913, 2018.

[33] Y. Yu, J. Li, H. Guan, C. Wang, and J. Yu, "Semiautomated extraction of street light poles from mobile LiDAR point-clouds," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 3, pp. 1374–1386, Mar. 2015.

[34] H. Zheng, R. Wang, and S. Xu, "Recognizing street lighting poles from mobile LiDAR data," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 1, pp. 407–420, Jan. 2017.

[35] Y. Li et al., "Pole-like street furniture segmentation and classification in mobile LiDAR data by integrating multiple shape-descriptor constraints," *Remote Sens.*, vol. 11, no. 24, pp. 2920–2942, 2019.

[36] J. Tu, J. Yao, L. Li, W. Zhao, and B. Xiang, "Extraction of street pole-like objects based on plane filtering from mobile LiDAR data," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 1, pp. 749–768, Jan. 2021.

[37] C. Ordóñez, C. Cabo, and E. Sanz-Ablanedo, "Automatic detection and classification of pole-like objects for urban cartography using mobile laser scanning data," *Sensors*, vol. 17, no. 7, pp. 1465–1474, 2017.

[38] A. Golovinskiy, V. G. Kim, and T. Funkhouser, "Shape-based recognition of 3D point clouds in urban environments," in *Proc. ICCV*, Kyoto, Japan, Sep. 2009, pp. 2154–2161.

[39] J. Huang and S. You, "Pole-like object detection and classification from urban point clouds," in *Proc. ICRA*, Seattle, WA, USA, May 2015, pp. 3032–3038.

[40] J. Huang and S. You, "Point cloud Labeling using 3D convolutional neural network," in *Proc. ICPR*, Cancun, Mexico, Dec. 2016, pp. 2670–2675.

[41] F. Li et al., "Semantic segmentation of road furniture in mobile laser scanning data," *ISPRS J. Photogrammetry Remote Sens.*, vol. 154, pp. 98–113, Aug. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092427161930142X

[42] M. Lehtomäki et al., "Object classification and recognition from mobile laser scanning point clouds in a road environment," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 2, pp. 1226–1239, Feb. 2016.

[43] Y. Zhou et al., "Street-view imagery guided street furniture inventory from mobile laser scanning point clouds," *ISPRS J. Photogrammetry Remote Sens.*, vol. 189, pp. 63–77, Jul. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271622001265

[44] J. Grandio, B. Riveiro, M. Soilán, and P. Arias, "Point cloud semantic segmentation of complex railway environments using deep learning," *Autom. Construction*, vol. 141, Sep. 2022, Art. no. 104425. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580522002989

[45] L. Zhou and Z. Deng, "LiDAR and vision-based real-time traffic sign detection and recognition algorithm for intelligent vehicle," in *Proc. ITSC*, Qingdao, China, Oct. 2014, pp. 578–583.

[46] Z. Deng and L. Zhou, "Detection and recognition of traffic planar objects using colorized laser scan and perspective distortion rectification," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1485–1495, May 2018.

[47] A. Vu, Q. Yang, A. F. Jay, and M. Barth, "Traffic sign detection, state estimation, and identification using onboard sensors," in *Proc. ITSC*, The Hague, The Netherlands, Oct. 2013, pp. 875–880.

[48] F. Ghallabi, G. El-Haj-Shhade, M. Mittet, and F. Nashashibi, "LiDAR-based road signs detection for vehicle Localization in an HD map," in *Proc. Intell. Veh. Symp.*, Paris, France, Jun. 2019, pp. 1484–1490.

[49] A. Schaefer, D. Büscher, J. Vertens, L. Luft, and W. Burgard, "Long-term urban vehicle localization using pole landmarks extracted from 3-D LiDAR scans," in *Proc. EMCR*, Prague, Czech Republic, Sep. 2019, pp. 1–7.

[50] M. B. Jensen, M. P. Philipsen, A. Møgelmose, T. B. Moeslund, and M. M. Trivedi, "Vision for looking at traffic lights: Issues, survey, and perspectives," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1800–1815, Jul. 2016.

[51] Z. Ouyang, J. Niu, Y. Liu, and M. Guizani, "Deep CNN-based real-time traffic light detector for self-driving vehicles," *IEEE Trans. Mobile Comput.*, vol. 19, no. 2, pp. 300–313, Feb. 2020.

[52] M. Weber, P. Wolf, and J. M. Zöllner, "DeepTLR: A single deep convolutional network for detection and classification of traffic lights," in *Proc. Intell. Veh. Symp.*, Las Vegas, NV, USA, Jun. 2016, pp. 342–348.

[53] N. Fairfield and C. Urmson, "Traffic light mapping and detection," in *Proc. ICRA*, Shanghai, China, May 2011, pp. 5421–5426.

[54] L. C. Possatti et al., "Traffic light recognition using deep learning and prior maps for autonomous cars," in *Proc. IJCNN*, Budapest, Hungary, Jul. 2019, pp. 1–8.

[55] Z. Li, Q. Zeng, Y. Liu, J. Liu, and L. Li, "An improved traffic lights recognition algorithm for autonomous driving in complex scenarios," *Int. J. Distrib. Sensor Netw.*, vol. 17, no. 5, p. 19, 2021. [Online]. Available: https://doi.org/10.1177/15501477211018374

[56] M. Hirabayashi, A. Sujiwo, A. Monrroy, S. Kato, and M. Edahiro, "Traffic light recognition using high-definition map features," *Robot. Auton. Syst.*, vol. 111, pp. 62–72, Jan. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889018301234

[57] B. Thomas, J. Lowenau, S. Durekovic, and H. Otto, "The ActMAP—FeedMAP framework: A basis for in-vehicle ADAS application improvement," in *Proc. Intell. Veh. Symp.*, Eindhoven, The Netherlands, Jun. 2008, pp. 263–268.

[58] C. Zinoune, P. Bonnifait, and J. Ibañez-Guzmán, "A sequential test for autonomous localization of map errors for driving assistance systems," in *Proc. ITSC*, Anchorage, AK, USA, Mar. 2012, pp. 1377–1382.

[59] A. Welte, P. Xu, P. Bonnifait, and C. Zinoune, "Estimating the reliability of geo-referenced lane markings for map-aided localization," in *Proc. Intell. Veh. Symp.*, Paris, France, Jun. 2019, pp. 1225–1231.

[60] D. Pannen, M. Liebner, and W. Burgard, "HD map change detection with a boosted particle filter," in *Proc. ICRA*, Montreal, QC, Canada, May 2019, pp. 2561–2567.

[61] P. Zhang, M. Zhang, and J. Liu, "Real-time HD map change detection for crowd-sourcing update based on mid-to-high-end sensors," *Sensors*, vol. 21, no. 7, p. 2477, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/7/2477

[62] J.-H. Pauls, T. Strauss, C. Hasberg, M. Lauer, and C. Stiller, "HD map verification without accurate Localization prior using spatio-semantic 1D signals," in *Proc. Intell. Veh. Symp. Workshops*, Oct. 2020, pp. 680–686.

[63] J.-H. Pauls, T. Strauss, C. Hasberg, and C. Stiller, "Boosted classifiers on 1D signals and mutual evaluation of independently aligned spatio-semantic feature groups for HD map change detection," in *Proc. Intell. Veh. Symp.*, Jul. 2021, pp. 961–966.

[64] S. R. Bhavsar, A. Vatavu, T. Rehfeld, and G. Krehl, "Sensor fusion-based online map validation for autonomous driving," in *Proc. Intell. Veh. Symp. Workshops*, virtual, Oct. 2020, pp. 77–82.

[65] O. Hartmann, M. Gabb, R. Schweiger, and K. Dietmayer, "Towards autonomous self-assessment of digital maps," in *Proc. Intell. Veh. Symp.*, Ypsilanti, MI, USA, Jun. 2014, pp. 89–95.

[66] J. Lambert and J. Hays, "Trust, but verify: Cross-modality fusion for HD map change detection," in *Proc. NeurIPS*, Dec. 2021, pp. 1–8. [Online]. Available: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/6f4922f45568161a8cdf4ad2299f6d23-Paper-round2.pdf

[67] A. Fabris, L. Parolini, S. Schneider, and A. Cenedese, "Correlation-based approach to online map validation," in *Proc. Intell. Veh. Symp. Workshops*, Oct. 2020, pp. 51–56.

[68] A. Fabris, L. Parolini, S. Schneider, and A. Cenedese, "Use of probabilistic graphical methods for online map validation," in *Proc. Intell. Veh. Symp.*, Jul. 2021, pp. 43–48.

[69] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *Proc. IROS*, Sep. 2017, pp. 1513–1518.

[70] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *Proc. ICRA*, Singapore, Jun. 2017, pp. 1355–1361.

[71] 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. CVPR*, Long Beach, CA, USA, Jun. 2019, pp. 12689–12697.

[72] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. CVPR*, Salt Lake City, UT, USA, Jun. 2018, pp. 4490–4499.

[73] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, pp. 3337–3354, 2018.

[74] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-based 3D single stage object detector," in *Proc. CVPR*, Jun. 2020, pp. 11037–11045.

[75] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: Sparse-to-dense 3D object detector for point cloud," in *Proc. ICCV*, Seoul, South Korea, Oct. 2019, pp. 1951–1960.

[76] W. Liu et al., "SSD: Single shot multibox detector," in *Proc. ECCV*, Amsterdam, The Netherlands, Oct. 2016, pp. 21–37.

[77] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020.

[78] P. J. Huber, "Robust estimation of a location parameter," *Ann. Math. Stat.*, vol. 35, no. 1, pp. 73–101, 1964.

[79] J. Huang et al., "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proc. CVPR*, Honolulu, HI, USA, Jul. 2017, pp. 3296–3297.

[80] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. ICCV*, Araucano Park, Chile, Dec. 2015, pp. 1026–1034.

CHRISTOPHER PLACHETKA received the B.Sc. degree in electrical engineering from Ostfalia Hochschule Wolfenbüttel, Germany, in 2015, and the M.Sc. degree in electronic systems from Technische Universität Braunschweig, Germany, in 2017, where he is currently pursuing the Ph.D. degree with the Faculty of Electrical Engineering, Information Technology, and Physics, in collaboration with Volkswagen. His research interests include neural networks for computer vision, high-definition (HD) maps, and automated labeling. He received a scholarship from Volkswagen financing the master degree and is the creator of the large-scale HD map dataset 3DHD CityScenes.

BENJAMIN SERTOLLI received the B.Sc. and M.Sc. degrees in computer science from Universität Augsburg, Germany, in 2019 and 2022, respectively. During his internship and subsequent master's thesis with Volkswagen, he worked on deep learning-based object detection in 3-D point clouds and deviation detection between sensor and map data. He is a co-creator of the 3DHD CityScenes dataset. His research interests include computer vision, automated driving, and speech processing.

JENNY FRICKE received the B.Sc. degree in automotive engineering from Ostfalia Hochschule Wolfsburg, Germany, in 2016, and the M.Sc. degree in industrial and mechanical engineering from Technische Universität Braunschweig, Germany, in 2019, where she is currently pursuing the Ph.D. degree in knowledge-based road modeling with the Faculty of Mechanical Engineering. Since 2019, she has been working as a Doctoral Researcher and an Engineer with Volkswagen. Her research interests include automated driving, road modeling, and knowledge-based systems.

MARVIN KLINGNER (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in physics from Georg-August-Universität Göttingen, Germany, in 2016 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Faculty of Electrical Engineering, Information Technology, and Physics, Technische Universität Braunschweig, Germany. His research interests lie in self-supervised 3D geometry perception, multi-task learning, and domain adaptation approaches for neural networks with focus on computer vision tasks. He was the recipient of the Dr. Berliner–Dr. Ungewitter Award of the Faculty of Physics at Georg-August-Universität Göttingen in 2018 and was given the CVPR SAIAD Workshop Best Paper Award in 2020 and 2021.

TIM FINGSCHEIDT (Senior Member, IEEE) received the Dipl.-Ing. degree in electrical engineering and the Ph.D. degree from RWTH Aachen University, Germany, in 1993 and 1998, respectively. He joined AT&T Labs, Florham Park, NJ, USA, in 1998; and Siemens AG (Mobile Devices), Munich, Germany, in 1999. With Siemens Corporate Technology, Munich, he was leading the speech technology development activities from 2005 to 2006. Since 2006, he has been a Full Professor with the Institute for Communications Technology, Technische Universität Braunschweig, Germany. His research interests include speech technology and vision for autonomous driving. He was the recipient of several awards, e.g., the Vodafone Mobile Communications Foundation Prize in 1999 and the 2002 ITG Prize of the Association of German Electrical Engineers (VDE ITG). In 2017 and 2020, he coauthored the ITG Award-Winning Publication. He was given the Best Paper Award of a CVPR Workshop from 2019 to 2021. He has been the Speaker of the Speech Acoustics Committee ITG AT3 since 2015. He was an Associate Editor of the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING from 2008 to 2010. He was a member of the IEEE Speech and Language Processing Technical Committee from 2011 to 2018.