3DHD CityScenes: High-Definition Maps in High-Density Point Clouds

Conference Paper · October 2022

DOI: 10.1109/ITSC55140.2022.9921866

5 authors, including:

Christopher Plachetka
Volkswagen AG
5 PUBLICATIONS **25** CITATIONS
SEE PROFILE

Benjamin Sertolli
Universität Augsburg
8 PUBLICATIONS **81** CITATIONS
SEE PROFILE

Jenny Fricke
Technische Universität Braunschweig
5 PUBLICATIONS **16** CITATIONS
SEE PROFILE

Marvin Klingner
Technische Universität Braunschweig
36 PUBLICATIONS **485** CITATIONS
SEE PROFILE

3DHD CityScenes: High-Definition Maps in High-Density Point Clouds

Christopher Plachetka[1], Benjamin Sertolli[1], Jenny Fricke[1], Marvin Klingner[2] and Tim Fingscheidt[2]

***Abstract*— In this paper, we present 3DHD CityScenes — a new dataset with the most comprehensive, large-scale high-definition (HD) map to date, annotated in the three spatial dimensions of globally referenced, high-density LiDAR point clouds collected in urban domains. The HD map covers a wide variety of map element types, for instance traffic signs and lights, construction site elements such as cones and fences, markings, lanes, and relations between map elements. Our presented dataset is suitable for numerous perception tasks, such as 3D object detection or map deviation detection. Furthermore, we address the example task of detecting traffic signs in LiDAR point clouds, proposing a novel method based on a deep neural network. Our architecture, named `3DHDNet`, specifically allows for the individual detection of vertically stacked signs. `3DHDNet` significantly outperforms two state-of-the-art architectures that we selected for comparison. Our method achieves an $\mathbf{F_1}$ score, recall, and precision, of 0.83, 0.76, and 0.90, respectively, and may serve as a baseline for future approaches. The dataset is available at `https://www.hi-drive.eu/Data` for both commercial and non-commercial use.**

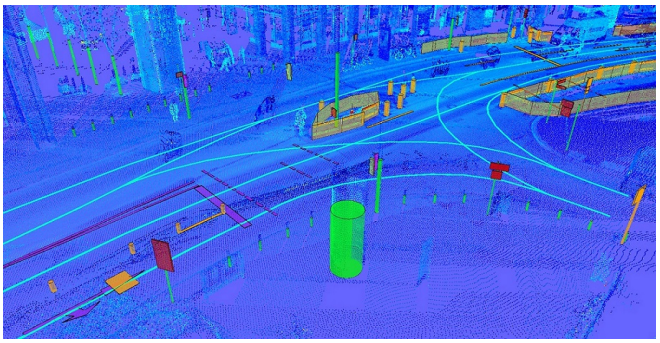


Fig. 1: **High-density point cloud with various HD map elements** contained in our dataset. The point cloud is colored according to the intensity of reflected LiDAR beams. Additionally, annotations for map elements are shown, such as poles (green), signs (red), lights (for vehicles in yellow and for pedestrians in magenta), markings (purple), lane center lines (cyan), and temporary construction sites elements, e.g., fences and delineators (orange).

I. Introduction

High-definition (HD) maps are a vital component for an automated vehicle and applied in numerous tasks. Most importantly, HD maps are used to compensate shortcomings of current environment modeling algorithms, e.g, to obtain the lane topology or traffic regulations, which demands a semantical understanding of the scene. Also, HD maps can be used to facilitate the environment perception task [1, 2], or for a landmark-based localization [3]. To develop such algorithms, only few datasets featuring HD maps exist [4–6], which either feature misalignment between sensor and map data [5], e.g., due to an imperfect localization, or provide only few different map element types [4, 6].

Our HD map, on the other hand, provides the most comprehensive set of different HD map element types to date, which allows for various new 3D object detection tasks. To the best of our knowledge, we are the first to provide 3D annotations regarding all three spatial dimensions of the point clouds for signs, poles, curbs, and construction site elements, such as delineators or fences. Such annotations include 3D positions and bounding polygons for all element types. Moreover, the map elements feature a precise alignment with the high-density point clouds, as these were used to generate the map. Accordingly, the new 3DHD CityScenes dataset with its accurate annotations allows research and development of both 3D object detection and map deviation detection [7]. The user of our dataset can induce known localization errors artificially, e.g., for ablation studies. 3DHD CityScenes is suitable both for geodesy, in which high-density point clouds are commonly used, e.g., for HD map generation [8–10], and for the automotive domain. For the latter, onboard scans can be simulated by taking crops from our high-density point clouds, which is the approach that we follow in this work.

As new example task to be performed on our introduced dataset, we select the 3D detection of traffic signs in LiDAR point clouds, which is a vital topic in the field of automated driving but has received only little attention in academic research, presumably due to a lack of a publicly available LiDAR dataset covering 3D traffic sign annotations. Besides map generation, traffic signs must also be detected on the fly to generate an environment model, if one aims at driving without a map or at detecting map deviations. Existing algorithms rely on handcrafted features [11], and additionally utilize camera images for sign detection [12, 13], while our deep neural network (DNN) achieves a high performance using LiDAR-only by learning an optimal feature representation and by incorporating context from the scene. The detection of traffic signs is particularly challenging as traffic signs are small and can be stacked in the vertical dimension.

To summarize, our contributions are as follows. First, we provide an HD map with a comprehensive set of different map element types. Second, we provide globally referenced, high-density point clouds featuring a highly precise map alignment. Third, we are the first to perform traffic sign detection directly in LiDAR point clouds using a DNN.

[1]Christopher Plachetka, Benjamin Sertolli, and Jenny Fricke are with Volkswagen Group, Commercial Vehicles, Berliner Ring 2, 38440 Wolfsburg, Germany. {`forename.surname`}`@volkswagen.de`

[2]Marvin Klingner and Tim Fingscheidt are with the Institute for Communications Technology, Technische Universität Braunschweig, Schleinitzstr. 22, 38106 Braunschweig, Germany. {`m.klingner`, `t.fingscheidt`}`@tu-bs.de`

II. Related Work

In this section, we review available HD map datasets and traffic sign detection methods in both the automotive field and in geodesy. We conclude with a review of methods for DNN-based object detection in LiDAR point clouds.

A. Datasets with HD Maps

ArgoVerse [4], NuScenes [5], and Level5 [6] provide onboard data, road user annotations, and underlying HD maps. However, the mentioned datasets address road user detection and motion prediction tasks. 3DHD CityScenes, on the other hand, focuses on 3D object detection of map elements, thus providing a more comprehensive set of different map element types as further analyzed in Section III-B. For comparison, ArgoVerse and 3DHD CityScenes contain 300 km and 467 km of lanes, respectively. Level5, on the other hand, provides 15,242 manually annotated map items, while 3DHD CityScenes offers 266,763 of such items. Other datasets exist that provide high-density point clouds for semantic segmentation, without offering HD maps [14, 15].

B. Traffic Sign Detection

Detecting traffic signs in camera images is a common task, while LiDAR-only approaches are less studied in public research [16]. In geodesy, high-density point clouds are a common input to traffic sign detection algorithms, which are developed on unpublished datasets. Existing approaches typically feature two stages: Traffic signs are first detected in the point cloud, and then projected into the camera image for further classification [17–19]. Regarding the LiDAR-based *sign detection* stage, various conventional algorithms exist that generally follow three steps. First, the ground points are removed [8–10]. Additionally, other removal criteria may be applied, e.g., based on the intensity of reflected LiDAR beams [20]. Second, the point cloud is segmented using various clustering algorithms, such as Euclidean clustering [8–10, 19] or DBSCAN [18, 20]. Third, a plane is fitted to point clusters, typically using a principal component analysis (PCA) [9, 18, 20] performed on the spatial distribution of points, or using the cluster's outline [19]. Regarding the *sign classification* stage, machine learning-based algorithms are typically employed, such as convolutional neural networks [18] or deep Boltzmann machines [19, 21].

In the automotive field, approaches generally follow the processing stages found in geodesy with the traffic sign detection performed on LiDAR data with a subsequent projection and classification in camera images [12, 13, 22]. To tackle the sparsity of onboard scans, [11, 22] construct pseudo images from LiDAR data, in which point clusters featuring a high intensity are found. Moreover, [12, 13] augment the point cloud with color features obtained from images to facilitate the clustering process. To obtain traffic sign planes, RANSAC [11–13] or PCA is employed [22]. Regarding the subsequent classification of traffic signs in the image domain, SVM-based methods [13] or template-matching approaches [22] have been presented.

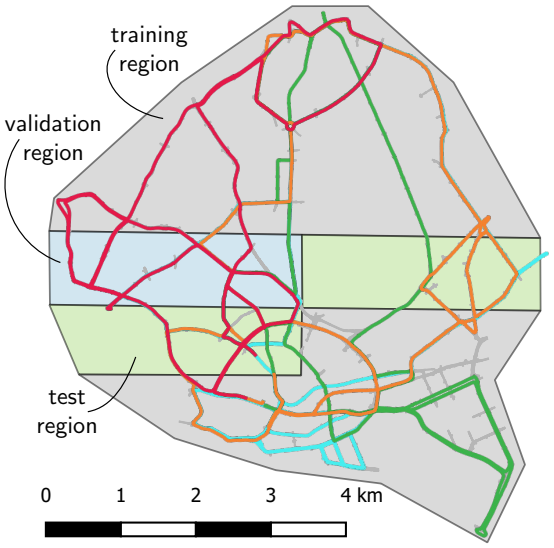


Fig. 2: **HD map and dataset split**. We show lanes (gray lines), 4 out of 14 real-world trajectories (colored lines, with the subset selected for better visibility), and regions of the map assigned to the training, validation, and test set (gray, blue, and green areas).

To the best of our knowledge, we are the first to employ a DNN for sign detection trained in an end-to-end fashion, omitting the need for hand-crafted feature design, clustering algorithms, or plane fitting. Moreover, our approach is suitable for both high-density and sparse point clouds.

C. DNN-Based Object Detection in LiDAR Point Clouds

While early works for DNN-based object detection rely on hand-crafted encodings [23, 24], the paradigm has shifted towards learned feature encodings [25–27], reducing the loss of geometrical information. Recent research examines point-based networks [28, 29] that omit the point cloud discretization. In contrast to known architectures in the field featuring learned encodings [25–27] that are designed for road user detection, our `3DHDNet` architecture allows for the resolution of vertically stacked objects.

III. 3DHD CityScenes Dataset

In this section, we introduce our new 3DHD CityScenes dataset. First, we present details regarding the high-density LiDAR point clouds and items of the HD map. Subsequently, we explain our method for generating samples used to train the DNN of the example task.

Overall, 3DHD CityScenes covers approx. 127 km of road sections[1] of the inner city of Hamburg, Germany (cf. Figure 2), including various different domains such as city streets, urban highways, and residential areas. The road sections incorporate 467 km of individual lanes. In total, the map comprises 266,763 individual items. The point clouds and HD map were created in 2017. During a measurement campaign conducted in 2019 [7], 14 real-world trajectories have been recorded to provide realistic driving routes.

[1] A *road section* groups all parallel lanes with the same driving direction, regardless the number of parallel lanes. A *road* on the other hand comprises a maximum of two road sections with opposing driving directions.

A. High-Density LiDAR Point Clouds

The provided high-density point clouds (HDPCs) visualized in Fig. 1 were recorded using the mobile mapping system `Trimble MX8` and are referenced in global UTM32N coordinates. The point density of each HDPC has been downsampled to $1/1\,\mathrm{dm}^3$. Each point comprises the Cartesian coordinates, the reflected LiDAR beam's intensity, and a ground classification flag. The z-dimension of a point corresponds to the geodetic height relative to the earth's surface. The entire map area is split into 648 tiles, each containing one HDPC with millions of point measurements. Note that sparse point clouds from rotating onboard scanners provided by other datasets [4–6] are naturally misaligned with the map, as single point measurements and localization poses arise at different timestamps. HDPCs from 3DHD CityScenes are post-processed to compensate this effect.

B. Map Elements and Relations

Subsequently, we apply the terminology for HD maps and map deviations defined in our previous work [7], according to which an HD map comprises map elements (e.g., traffic signs) and relations[2] between such elements (e.g., the association of a sign to a lane). We use the term *map item* as generic term for an element or relation. The entire set of various map item types and their features in comparison with other HD map datasets is presented in Table I. All items feature a unique identifier and a deviation annotation. The latter classifies an element as temporary, experimental[3], or negated [7]. According to Table I, 3DHD CityScenes exclusively provides 3D annotations for traffic signs, poles, construction site obstacles, curbs, and polygon markings (e.g., arrows, symbols, zebra markings, etc.). Polygon markings are vital to derive a higher semantical understanding of the scene from physical detections, e.g., a lane restriction to buses. Regarding traffic signs, the L5 dataset [6] only provides stop signs in a 2D representation without z-positions or bounding rectangles. Moreover, also the deviation annotation described above is novel. While other datasets provide relations only for traffic lights, 3DHD CityScenes also incorporates traffic signs and negation relations (e.g., a cross marking negating an arrow marking). The drivable area within a lane is given by the center line and a lane width.

C. Sample Generation

In the following, we describe our procedure to generate training, validation, and test samples (comprising a point cloud and map elements) from the larger HDPCs and the HD map, e.g., required to train a neural network. Such samples simulate scans from an onboard LiDAR sensor as used in the automotive domain. First, all global poses (position and orientation) originating from the recorded real-world trajectories are pooled and subsequently filtered, so that we ensure a minimal distance of $2\,\mathrm{m}$ between poses.

[2]Compared to our previous work [7], we use the term *relation* instead of *connection*, as the latter term is used in the context of lanes.

[3]An *experimental* element (e.g., a road marking) is temporary but appears outside of a construction site.

TABLE I: **Summary of all map items** contained in 3DHD CityScenes compared to ArgoVerse [4] (AV), NuScenes [5] (NS), and Level5 [6] (L5). The variables h, w, φ, refer to an object's height, width, and orientation, while z indicates the geodetic height relative to the earth's surface. The amount of instances regarding a specific item type contained in our dataset is indicated in squared brackets below the respective type. We use the following abbreviations: Lat (latitude), lon (longitude), CS (construction site), and CSO (construction site obstacle). The "face" of a traffic light refers to the front side of the light box. An orange checkmark indicates that a published dataset provides less item features than ours.

Map Item	Features of 3DHD CityScenes	AV	NS	L5
Traffic sign [20,163]	- Rectangle center position (lat, lon, z) - Bounding rectangle (h, w, φ) - Shape: Triangle, rectangle, circle, etc. - Class: See German sign catalog 2017 [30]	✗	✗	✓
Traffic light [5,762]	- Face position (lat, lon, z) - Face rectangle (h, w, φ) - Colors: Has red, has yellow, has green - Class: Vehicle, pedestrian, warning - Signal: Left arrow, pedestrian, bike, etc. - Layout: Vertical, horizontal	✗	✓	✓
Pole [67,540]	- Base point position (lat, lon, z) - Diameter - Class: Lamppost, tree, bollard, pillar, etc.	✗	✗	✗
CS [20]	- Outline of a construction site - Polygon (tuples of lat, lon, z)	✗	✗	✗
CSO (point) [2,515]	- Base point position (lat, lon, z) - Diameter - Class: Cone, delineator, reflector	✗	✗	✗
CSO (line) [1,767]	- Position as polyline (tuples of lat, lon, z) - Width - Class: Fence, guide board, reflectors	✗	✗	✗
Curb [20,054]	- Position as polyline (tuples of lat, lon, z) - Class: High, low	✗	✗	✗
Marking (line) [107,017]	- Position as polyline (tuples of lat, lon, z) - Color: White, yellow - Class: Solid, dashed, bike, pedestrian	✗	✓	✗
Marking (polyg.) [11,508]	- Position as polygon (tuples of lat, lon, z) - Color: White, yellow - Class: Ordinary, arrow, text, symbol - Ordinary: zebra, emergency, stop, negation - Arrow: Straight, right, deflection left, etc. - Text: "BUS", "TAXI", etc. - Symbol: Bicycle, charging, speed limit, etc.	✗	✗	✗
Lane [29,424]	- Centerline as polyline (tuples of lat, lon, z) - Width - Class: Normal, bike, bus, taxi - Connection type: Continuation, split, merge - Successors: List of lanes (per ID)	✓	✓	✓
Relation [993]	- Class: light to lane, sign to lane, negation	✗	✓	✓

Second, under consideration of the ego vehicle's orientation, we take a rectangular crop from the larger point cloud. The crop is then transformed into the local vehicle reference frame, with the x-axes pointing along the vehicle's length axis. We normalize the z-coordinates by subtracting the mean value of all ground-classified points. Last, we collect map items that are within the crop's range and transform them into the vehicle reference frame as well. In this work, we use a crop size of $x_{\min} = -10\,\mathrm{m}$, $x_{\max} = 50.8\,\mathrm{m}$, $y_{\min} = -20\,\mathrm{m}$, $y_{\max} = 20\,\mathrm{m}$, $z_{\min} = -2\,\mathrm{m}$, $z_{\max} = 7.6\,\mathrm{m}$.

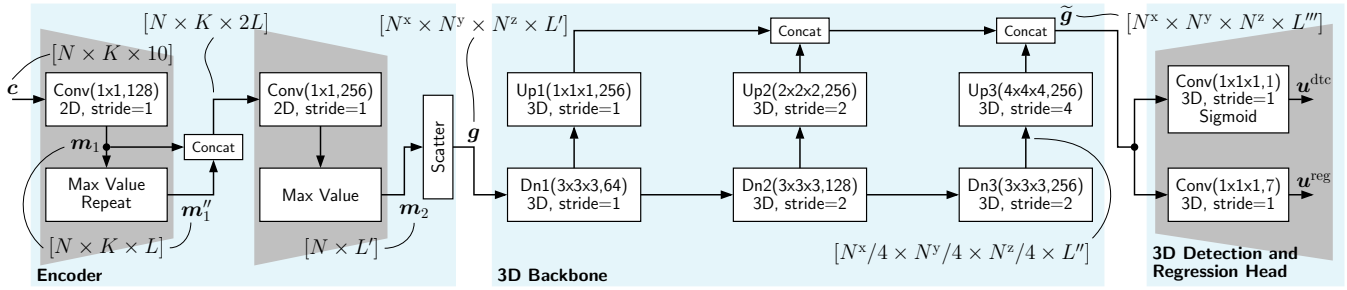


Fig. 3: **3DHDNet**. Blue: Processing stages. Gray: Encoder and decoder stages. A 2D convolution ("2D") Conv(1x1, F) with input dimension $[N \times K \times 10]$ uses F kernels of size $1 \times 1 \times 10$, while a 3D convolution ("3D") Conv(1x1x1, F) with $[N^{\mathrm{x}} \times N^{\mathrm{y}} \times N^{\mathrm{z}} \times L''']$-dim. input uses $1 \times 1 \times 1 \times L'''$-sized kernels. "Concat" indicates a concatenation. Downsampling (Dn) blocks comprise multiple conv. layers.

IV. DNN-Based Sign Detection Method

In this section, we introduce our proposed DNN-based method for detecting traffic signs in LiDAR point clouds. First, we present the network's architecture. Subsequently, we introduce our anchor design and the respective strategy for matching ground truth objects to anchors during training. Finally, we define the loss used to train the network.

A. Network Architecture

Our `3DHDNet` architecture inspired by [26] is visualized in Fig. 3 and comprises three stages: encoder, backbone, and head. A tutorial description regarding learned encodings for point clouds can be found in our previous work [31].

The novel *encoder* stage learns an optimal feature representation for the point cloud. As a first step, the point cloud is discretized to obtain a 3D voxel grid with N^{x}, N^{y}, and N^{z} voxels in the x-, y-, and z-dimension of the grid. Then, only the N occupied voxels with a maximum of $K = 100$ points per voxel are collected into a second grid $\boldsymbol{c} = (\boldsymbol{c}_{n,k})$ of size $N \times K \times 10$ as input to the encoder, with $n \in \mathcal{N} = \{1, ..., N\}$ being the voxel index and $k \in \mathcal{K} = \{1, ..., K\}$ being the point index, respectively. Following the augmentation strategy in [25], each (augmented) point of the grid $\boldsymbol{c}_{n,k} = (v_{n,k}^{\mathrm{int}}, \boldsymbol{v}_{n,k}^{\mathrm{crd}}, \boldsymbol{v}_{n,k}^{\mathrm{crd}} - \overline{\boldsymbol{v}}_n, \boldsymbol{v}_{n,k}^{\mathrm{crd}} - \boldsymbol{w}_n)$ with $v_{n,k}^{\mathrm{int}} \in \mathbb{I} = [0, 1], \boldsymbol{v}_{n,k}^{\mathrm{crd}} \in \mathbb{R}^3, \overline{\boldsymbol{v}}_n \in \mathbb{R}^3, \boldsymbol{w}_n \in \mathbb{R}^3$ provides ten features, with $v_{n,k}^{\mathrm{int}}$ being an intensity measurement, $\boldsymbol{v}_{n,k}^{\mathrm{crd}}$ the Cartesian point coordinate, $\overline{\boldsymbol{v}}_n$ the mean of all Cartesian point measurements contained in voxel n, and $\boldsymbol{w}_n$ the Cartesian center coordinate of the point's assigned voxel n. In the first encoding stage, each point $\boldsymbol{c}_{n,k}$ is first mapped to $L = 128$ features using a 2D convolution with a $1 \times 1 \times 10$ kernel, which yields $\boldsymbol{m}_1 \in \mathbb{R}^{N \times K \times L}$. Subsequently, the maximum for each feature among all K points in a voxel n is taken to obtain $\boldsymbol{m}_1' \in \mathbb{R}^{N \times 1 \times L}$, and then repeated K times to match dimensions with $\boldsymbol{m}_1$, yielding $\boldsymbol{m}_1'' \in \mathbb{R}^{N \times K \times L}$. Each point contained in $\boldsymbol{m}_1$ 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 mapping points to $L' = 256$ features, the second maximum operation yields the final encoding $\boldsymbol{m}_2 \in \mathbb{R}^{N \times 1 \times L'}$ for all points contained in each voxel n. Finally, the N obtained feature vectors are scattered back to their original position in the voxel grid with zero-padding applied to empty voxels, which yields $\boldsymbol{g} \in \mathbb{R}^{N^{\mathrm{x}} \times N^{\mathrm{y}} \times N^{\mathrm{z}} \times L'}$.

The novel *3D backbone* stage processing $\boldsymbol{g}$ comprises a downstream and an upstream network, utilizing 3D (transposed) convolutions in both networks on the 3D voxel grid, allowing for the individual detection of vertically stacked objects. The downstream network is composed of three blocks ("Dn" in Fig. 3), 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 $\widetilde{\boldsymbol{g}} \in \mathbb{R}^{N^{\mathrm{x}} \times N^{\mathrm{y}} \times N^{\mathrm{z}} \times L'''}$, with $L''' = 3L'' = 768$ and $L'' = 256$.

Our novel *3D detection and regression head* operates in a single-shot fashion [32], whereby the network predicts existence likelihoods and regresses bounding rectangles 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. We place one anchor in each voxel of tensor $\widetilde{\boldsymbol{g}}$, with $g \in \mathcal{G} = \{1, 2, ..., G\}$ being the voxel index in the grid, and $G = N^{\mathrm{x}} \cdot N^{\mathrm{y}} \cdot N^{\mathrm{z}}$ being the number of voxels. Tensor $\boldsymbol{u}^{(\cdot)} = \left(\boldsymbol{u}_1^{(\cdot)}, ..., \boldsymbol{u}_G^{(\cdot)}\right)$ comprises predictions $\boldsymbol{u}_g^{(\cdot)}$ for single voxels g. The detection head predicts an existence likelihood for each anchor $\boldsymbol{u}_g^{\mathrm{dtc}} = u_g^{\mathrm{dtc}} \in \mathbb{I}$ with $\mathbb{I} = [0, 1]$, while the regression head adopts the anchor's (predefined) bounding rectangle to match the size and position of a real-world object. For the latter, we employ the bounding rectangle parameters $\{x^{\mathrm{pos}}, y^{\mathrm{pos}}, z^{\mathrm{pos}}, h, w, \varphi\}$, with x^{pos}, y^{pos}, z^{pos} being the Cartesian position of a sign, h and w being the rectangle's height and width, respectively, and φ being the sign's orientation. The regression head outputs the difference between a real-world object's bounding rectangle (superscript O), and the bounding rectangle of the respective anchor (superscript A). With $r_{\mathrm{vox}} \times r_{\mathrm{vox}} \times r_{\mathrm{vox}}$ 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 head's output $\boldsymbol{u}_g^{\mathrm{reg}} = (u_g^{(x^{\mathrm{pos}})}, u_g^{(y^{\mathrm{pos}})}, u_g^{(z^{\mathrm{pos}})}, u_g^{(w)}, u_g^{(h)}, u_g^{(\varphi^{\mathrm{re}})}, u_g^{(\varphi^{\mathrm{im}})})$ as

$$u_g^{(p)} = \frac{(p_g^{\mathrm{O}} - p_g^{\mathrm{A}})}{r_{\mathrm{vox}}}, \quad p \in \{x^{\mathrm{pos}}, y^{\mathrm{pos}}, z^{\mathrm{pos}}\}, \qquad (1)$$

$$u_g^{(\ell)} = \log\Big(\frac{\ell_g^{\mathrm{O}}}{\ell_g^{\mathrm{A}}}\Big), \quad \ell \in \{w, h\}, \tag{2}$$

$$u_g^{(\varphi^{\mathrm{re}})} + j \cdot u_g^{(\varphi^{\mathrm{im}})} = \cos(2\varphi^{\mathrm{O}}) + j \cdot \sin(2\varphi^{\mathrm{O}}), \tag{3}$$

with j being the imaginary unit. Without camera images, the network cannot distinguish between a sign's front and back. 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 2 in (3). During inference, the normalization of predictions is reverted and only anchors with a predicted existence likelihood $\boldsymbol{u}_g^{\mathrm{dtc}} = u_g^{\mathrm{dtc}}$ above a threshold Θ^{score} are considered as valid detection:

$$u_g^{\mathrm{dtc}} \geq \Theta^{\mathrm{score}}. \tag{4}$$

Multiple anchors can make predictions for the same real-world object. Thus, in a post-processing step, respective overlapping bounding rectangles are filtered using non-maximum-suppression [25] to obtain the final object list. Thereby, only the objects with the highest existence likelihood remains. To measure the overlap, we use the same criteria as for matching ground truth objects to anchors described in the following section.

B. Anchor Design and Matching Strategy

We design our anchors based on the ground truth distribution for sign bounding rectangles shown in Fig. 4. The distribution for rectangle height and width has peaks around $\overline{h} \approx 0.65\,\mathrm{m}$ and width $\overline{w} \approx 0.65\,\mathrm{m}$, 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 $0.4\,\mathrm{m}$. Thus, we use $r_{\mathrm{vox}} = 0.4\,\mathrm{m}$ as voxel size for the anchor grid. Also, Fig. 4 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 described in Section III-C.

During training, ground truth (GT) objects are matched to the anchor grid to provide the targets $\overline{\boldsymbol{u}}_g^{\mathrm{dtc}} = \overline{u}_g^{\mathrm{dtc}}$ and $\overline{\boldsymbol{u}}_g^{\mathrm{reg}}$ for the respective network outputs. A matching anchor thereby features a close proximity to the respective GT object and a vertical overlap of respective bounding rectangles. Note that a single GT object can match with multiple anchors. If a GT object matches with an anchor in voxel g, we set the anchor's detection target to $\overline{u}_g^{\mathrm{dtc}} = 1$. The regression target vector $\overline{\boldsymbol{u}}_g^{\mathrm{reg}}$ is set according to (1), (2), and (3). As condition for a match, we use the following two criteria: First, we find a set of matching candidates in the x-y-plane. To this end, we define the "distance to line" (DtL) criterion as the shortest distance between the line segment of the GT rectangle (defined by the edge points), and an anchor's center point. We require $\mathrm{DtL} < \frac{r_{\mathrm{vox}}}{2} = 20\,\mathrm{cm}$ to consider anchors as matching candidates. Second, the final matches are obtained by filtering candidate rectangles according to the "z-overlap" z_{over} with the GT rectangle in the z-dimension. For this purpose, we use the overlap of line segments given by respective $z_{\min}$ and $z_{\max}$ values, which we normalize by the smaller segment's length. The anchor is considered a final match (setting $\overline{u}_g^{\mathrm{dtc}} = 1$), if $z_{\mathrm{over}} \geq 0.2$, and is discarded (setting $\overline{u}_g^{\mathrm{dtc}} = 0$) when $z_{\mathrm{over}} \leq 0.1$. Furthermore, we allow a "don't care" state for $0.1 < z_{\mathrm{over}} < 0.2$, in which an anchor is not considered for loss computation.

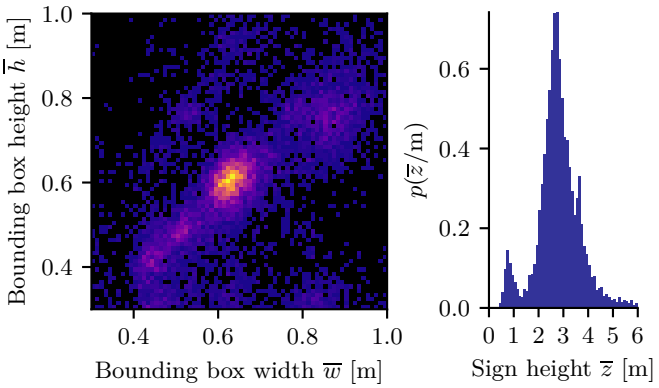


Fig. 4: **Ground truth distributions for sign bounding rectangles.** Left: Distribution of bounding box width $\overline{w}$ and height $\overline{h}$. Brighter colors indicate a higher occurrence. Right: Distribution of a sign's z-position $\overline{z}$, here encoded as height above ground.

C. Loss

We define the loss function optimizing the network similar to `PointPillars` [25] and `SECOND` [27] as

$$\begin{aligned} J = \frac{1}{N^{\mathrm{signs}}} \cdot \sum_{g \in \mathcal{G}} & \lambda \cdot J_g^{\mathrm{dtc}}(\overline{u}_g^{\mathrm{dtc}}, u_g^{\mathrm{dtc}}) \\ & + (1-\lambda) \cdot J_g^{\mathrm{reg}}(\overline{\boldsymbol{u}}_g^{\mathrm{reg}}, \boldsymbol{u}_g^{\mathrm{reg}}), \end{aligned} \tag{5}$$

whereby J_g^{dtc} and J_g^{reg} denote the detection and regression loss, respectively, N^{signs} equals the number of signs contained in a training sample, and $\lambda = 2/3$ is a weight factor.

We formulate the detection loss J_g^{dtc} as focal loss [33], which focuses on particularly hard-to-detect signs using adaptive weights. Thereby, using the distinction of cases depending on $\overline{u}_g^{\mathrm{dtc}}$ in (6), an anchor contained in voxel g is weighted higher when a prediction is further away from the target value. Using the original paper settings [33] for the weight factors $\alpha = 0.25$, $\beta = 2$, and with the mask factor $\lambda_g^{\mathrm{mask}} \in \{1, 0\}$ being zero in the "don't care" state (cf. Section IV-B), we define:

$$J_g^{\mathrm{dtc}} = -\lambda_g^{\mathrm{mask}} \alpha_g \cdot (1-\gamma_g)^{\beta} \cdot \log(\gamma_g), \tag{6}$$

$$\gamma_g = \begin{cases} u_g^{\mathrm{dtc}} & \text{if object in voxel } g \ (\overline{u}_g^{\mathrm{dtc}} = 1) \\ 1 - u_g^{\mathrm{dtc}} & \text{otherwise} \end{cases},$$

$$\alpha_g = \begin{cases} \alpha & \text{if object in voxel } g \ (\overline{u}_g^{\mathrm{dtc}} = 1) \\ 1 - \alpha & \text{otherwise} \end{cases}.$$

The regression loss is formulated using the smooth L1-loss (also known as Huber loss [34]):

$$J_g^{\mathrm{reg}} = \lambda_g^{\mathrm{obj}} \sum_{q \in \mathcal{Q}} \mathrm{smoothL1}(\Delta u_g^{(q)}), \tag{7}$$

$$\text{smoothL1}(\Delta u_g^{(q)}) = \begin{cases} 0.5(\Delta u_g^{(q)})^2 & \text{if } |\Delta u_g^{(q)}| \leq 1 \\ |\Delta u_g^{(q)}| - 0.5 & \text{otherwise} \end{cases}, \quad (8)$$

with $q \in \mathcal{Q} = \{x^{\text{pos}}, y^{\text{pos}}, z^{\text{pos}}, h, w, \varphi^{\text{re}}, \varphi^{\text{im}}\}$ being a regression parameter, $\Delta u_g^{(q)} = u_g^{(q)} - \overline{u}_g^{(q)}$ being the difference between predicted and target regression values, respectively, and with $\lambda_g^{\text{obj}} \in \{1, 0\}$ being 1 only if an object is present in voxel g.

V. Experimental Setup

In this section, we describe our experimental setup. First, we provide the definition of applied evaluation metrics. Subsequently, we describe the applied training procedure, and provide architectural and implementation details regarding the approaches selected for performance comparison.

A. Metrics

We evaluate the detection performance of our approach using the following metrics: Recall $\text{RE} = \frac{\text{TP}}{\text{TP+FN}}$, precision $\text{PR} = \frac{\text{TP}}{\text{TP+FP}}$, and the F_1 score $\text{F}_1 = 2 \cdot \frac{\text{PR} \cdot \text{RE}}{\text{PR+RE}}$, with TP and FP being the amount of true and false positives, respectively, and FN being the amount of false negatives. Signs are only counted as TP if the two matching criteria DtL and z_{over} as defined in Section IV-B regarding sign prediction and GT object are met. To evaluate the regression performance, we define the L1 and L2 error measures as:

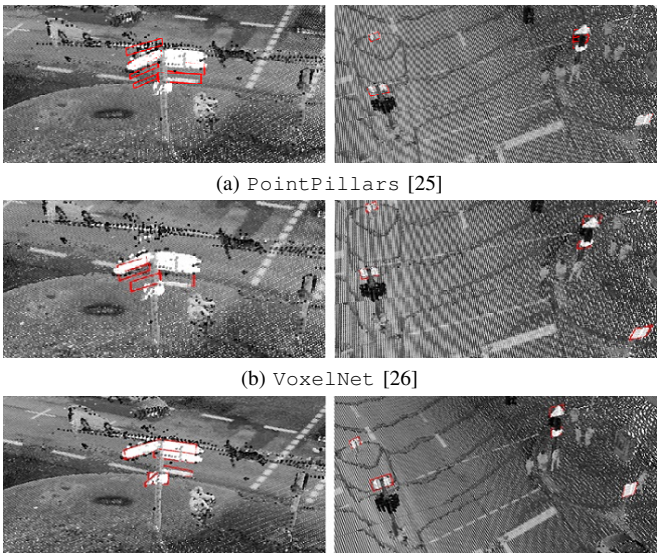
$$\text{L1:} \qquad E(q) = \frac{1}{\text{TP}} \sum_{i=1}^{\text{TP}} ||q_i - \overline{q}_i||_1, \quad (9)$$

$$\text{L2:} \qquad E(\boldsymbol{p}) = \frac{1}{\text{TP}} \sum_{i=1}^{\text{TP}} ||\boldsymbol{p}_i - \overline{\boldsymbol{p}}_i||_2, \quad (10)$$

with $q \in \{w, h, \varphi\}$, and $\boldsymbol{p} = (x^{\text{pos}}, y^{\text{pos}}, z^{\text{pos}})$ being the center of a predicted rectangle, and i being the object's index.

B. Training Procedure

Our experimental pipeline is implemented in `PyTorch`. All networks are trained for 10 epochs using a 2-GPU setup (`Tesla V100`) with a batch size of 2. We utilize the Adam optimizer with a fixed learning rate of $2 \cdot 10^{-4}$ and a momentum of 0.9. Moreover, we use a global augmentation strategy for both point cloud and signs. Thereby, we apply a random translation $\boldsymbol{t} \in \mathcal{N}^2(\mu = 0, \sigma = 1)$, with $\mathcal{N}()$ being the Gaussian distribution, and a random rotation around the z-axis by $\varphi \in \mathcal{U}$, with $\mathcal{U} = [-20°, 20°]$ being a uniform distribution. As multi-occurrences of individual signs in different point cloud samples exist, we randomly remove 10% of all points in each crop to reduce potential overfitting. If vertically-stacked rectangular signs of the same size with no spatial gap between (e.g., indicating directions, typically found on urban highways) appear as a single rectangle, we merge respective signs as the sign boundaries cannot be determined in the point cloud, even for a human. The training, validation, and test set comprise 54140, 7528, and 11881 samples, respectively, whereby each sample contains a subset of the 20094 signs in 3DHD CityScenes (cf. Table I).



(a) `PointPillars` [25]

(b) `VoxelNet` [26]

(c) `3DHDNet` (ours)

Fig. 5: **Example predictions for vertically stacked traffic signs** made by different network architectures. Predictions are highlighted in red, while the gray value indicates LiDAR reflectivity.

C. Architectural and Implementation Details

The `PointPillars` [25] architecture operates on a discretized 2D bird's eye view of the point cloud, with the anchor grid only containing 2D grid cells placed in the x-y-plane. The encoder comprises only one stage with 64 features and augments each point with the two (Cartesian) offsets regarding the mean of all points in a cell, and the point's respective cell center. Both backbone and network heads utilize 2D convolutions only. Thus, the final detection and regression heads have to predict all vertically stacked anchors for a single 2D grid cell simultaneously. We use a square cell size of $r_{\text{cell}} \times r_{\text{cell}}$, with $r_{\text{cell}} = 20\,\text{cm}$.

`VoxelNet` [26] on the other hand discretizes the point cloud into a 3D voxel grid, and utilizes a three-stage encoder with 32, 128, and 128 features, respectively. The encoder augments a point only with the (Cartesian) offset regarding the mean of all points in a respective voxel. Moreover, the network comprises a "middle extractor" located between the encoder's scatter operation and the 2D backbone. The "middle extractor" uses a series of 3D convolutions with a final feature concatenation operation to map the (encoded) voxel grid into a 2D bird's eye view as input to the 2D backbone. We apply the same voxel size $r^{\text{x}}_{\text{vox}} \times r^{\text{y}}_{\text{vox}} \times r^{\text{z}}_{\text{vox}} = (20 \times 20 \times 40)\ \text{cm}$ as used in the original paper.

Our proposed `3DHDNet` architecture omits the "middle extractor" and uses 3D convolutions consequently throughout the backbone and network heads. To compensate the additional computational cost, we use a larger cubic voxel size $r_{\text{vox}} = 40\,\text{cm}$, to reduce the overall number of voxels that are processed in the backbone and the network heads. With the extent of a point cloud crop specified in Section III-C, the resulting voxel grid contains $N^{\text{x}} = 152$, $N^{\text{y}} = 100$, and $N^{\text{z}} = 24$ voxels in the respective dimensions. To compensate for the coarser voxel resolution, we use a more powerful

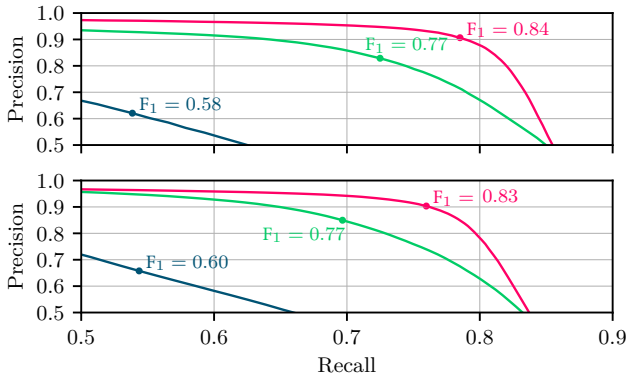


Fig. 6: **Precision-recall curves** on the **validation set** (top) and the **test set** (bottom). We use the color code: `PointPillars` [25] (blue), `VoxelNet` [26] (green), `3DHDNet` (red).

two-stage encoder with 256 and 256 features, respectively, whereby each point is augmented with the (Cartesian) offsets regarding the mean of all points in a voxel, and the voxel's center coordinate (cf. Section IV-A).

VI. Experimental Results and Discussion

In this section, we present and discuss the experimental results for our sign detection method. First, we evaluate the respective detection performances on the validation set. Subsequently, we present the test set results.

A. Validation Set Analysis

The example predictions made by three different architectures depicted in Fig. 5 show the superior capability of `3DHDNet` to detect vertically stacked traffic signs as individual objects, especially for clusters of such signs (see the left half of Fig. 5). Specifically, predictions by `PointPillars` (Fig. 5a) indicate that a single feature vector per 2D grid cell combined with a 2D backbone and head is not sufficient to resolve vertically stacked signs. `VoxelNet` preserves more geometric information during encoding, which allows to resolve simple cases of stacked signs (right of Fig. 5b). However, clusters of stacked signs are still an issue (left of Fig. 5b), which we attribute to the 2D backbone and head as a bottleneck for information related to the vertical dimension. The 3D backbone, head, and anchor grid applied in `3DHDNet` yield significant improvements regarding the detection of vertically stacked signs, as the vertical dimension is also used for convolution. Our architecture is also capable of resolving respective sign clusters (cf. Fig. 5c). Thus, we conclude that the positional information of signs is largely encoded by the spatial position of voxels in the grid, rather than in a voxel's feature vector, which can seemingly only encode small positional offsets. The network's detection performance evaluated on the validation set is shown in Fig. 6 (top). We select the operating point separately for each network by maximizing the F_1 score on the validation set, which yields the respective existence likelihood thresholds for `3DHDNet`, `VoxelNet`, and `PointPillars` of $\Theta^{\text{score}} = 0.34$ ($F_1 = 0.84$), $\Theta^{\text{score}} = 0.40$ ($F_1 = 0.77$), and $\Theta^{\text{score}} = 0.35$ ($F_1 = 0.58$), respectively. The obtained thresholds determine the operating points on the test set.

TABLE II: **Test set results** for the examined network architectures. Errors for position $E(\boldsymbol{p})$, height $E(h)$, and width $E(w)$ are indicated in cm, while the orientation error $E(\varphi)$ is given in degree.

Network Architecture	RE	PR	F_1	$E(\boldsymbol{p})$	$E(h)$	$E(w)$	$E(\varphi)$
`PointPillars` [25]	0.54	0.66	0.60	17.8	17.1	10.5	20.3
`VoxelNet` [26]	0.70	0.85	0.77	12.6	13.8	8.8	19.4
`3DHDNet` (ours)	**0.76**	**0.90**	**0.83**	**10.0**	**11.5**	**8.3**	**13.0**

B. Test Set Analysis

The test set performance is shown in Fig. 6 (bottom) and summarized in Table II. While `VoxelNet` shows an improved detection capability compared to `PointPillars`, our `3DHDNet` architecture outperforms both methods significantly. We achieve the best results among all metrics, with $F_1 = 0.83$, $RE = 0.76$, and $PR = 0.90$. Also, the error metrics (9), (10) regarding bounding rectangle regression parameters are best for `3DHDNet`. Specifically, we achieve mean errors for distance, rectangle height, rectangle width, and orientation of $E(\boldsymbol{p}) = 10.0\,\text{cm}$, $E(h) = 11.5\,\text{cm}$, $E(w) = 8.3\,\text{cm}$, and $E(\varphi) = 13.0°$. The largest improvement is achieved for the distance and rectangle height errors $E(\boldsymbol{p})$ and $E(h)$, when comparing `PointPillars` and `3DHDNet`. Using our pipeline featuring a `Tesla V100` GPU, the mean execution time during test inference without pre- and post-processing for `PointPillars`, `VoxelNet`, and `3DHDNet` is 403 ms, 616 ms, and 548 ms, respectively. Note that the combination of a powerful encoder and larger voxel size used in `3DHDNet` compensates for the additional computation cost due to application of 3D convolutions in the backbone and head, `3DHDNet` being 11% faster and 6% (absolute for F_1) more performant than 2nd-best `VoxelNet`.

VII. Conclusions

In this paper, we introduce 3DHD CityScenes, a novel large-scale dataset featuring high-definition (HD) maps and high-density point clouds obtained from a LiDAR-based mobile mapping system. The dataset comprises 127 km of road sections, 467 km of lanes, and 266,763 individual map items. Moreover, the dataset includes 14 real-world trajectories to provide realistic driving routes. The point clouds are globally referenced and used as basis to annotate the HD map, thus, featuring a highly-precise map alignment. The HD map is annotated in all three spatial dimensions of the point clouds and offers the most comprehensive set of different HD map element types to date, with various element types only provided by 3DHD CityScenes, such as 3D annotations for traffic signs, poles, construction site elements, curbs, and detailed road marking types. 3DHD CityScenes enables various new 3D object detection tasks, of which we selected 3D traffic sign detection as example task for this work.

For this purpose, we propose a novel deep neural network architecture, termed `3DHDNet`, which allows for the individual detection of vertically stacked objects. Our experiments show that well-known architectures in the field have a bottleneck regarding the preservation of information related to the vertical dimension of points when encoding the point cloud. The `3DHDNet` architecture addresses this

issue and significantly outperforms state-of-the-art networks, achieving an F_1 score, recall, and precision, of 0.83, 0.76, and 0.90, respectively, which is an 6% absolute increase of F_1 compared to state of the art, while being 11% faster. The 3DHD CityScenes dataset is available at `https://www.hi-drive.eu/Data`.

REFERENCES

[1] B. Yang, M. Liang, and R. Urtasun, "HDNET: Exploiting HD Maps for 3D Object Detection," in *Proc. of CoRL*, Zürich, Switzerland, Oct. 2018, pp. 146–155.

[2] L. C. Possatti, R. Guidolini, V. B. Cardoso, R. F. Berriel, T. M. Paixão, C. Badue, A. F. D. Souza, and T. Oliveira-Santos, "Traffic Light Recognition Using Deep Learning and Prior Maps for Autonomous Cars," in *Proc. of IJCNN*, Budapest, Hungary, Jul. 2019, pp. 1–8.

[3] D. Wilbers, C. Merfels, and C. Stachniss, "Localization With Sliding Window Factor Graphs on Third-Party Maps for Automated Driving," in *Proc. of ICRA*, Montreal, Canada, May 2019, pp. 5951–5957.

[4] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3D Tracking and Forecasting with Rich Maps," in *Proc. of CVPR*, Long Beach, CA, USA, Jun. 2019, pp. 8740–8749.

[5] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, E. L. Venice, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *Proc. of CVPR*, virtual, Jun. 2020, pp. 11 618–11 628.

[6] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. (2020). One Thousand and One Hours: Self-driving Motion Prediction Dataset, [Online]. Available: `https://level-5.global/level5/data/`.

[7] 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. of IV - Workshops*, virtual, Oct. 2020, pp. 63–70.

[8] P. Huang, M. Cheng, Y. Chen, H. Luo, C. Wang, and J. Li, "Traffic Sign Occlusion Detection Using Mobile Laser Scanning Point Clouds," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2364–2376, 2017.

[9] C. Wen, J. Li, H. Luo, Y. Yu, Z. Cai, H. Wang, and C. Wang, "Spatial-Related Traffic Sign Inspection for Inventory Purposes Using Mobile Laser Scanning Data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 27–37, 2016.

[10] Y. Yu, J. Li, H. Guan, and C. Wang, "Automated Extraction of Urban Road Facilities Using Mobile Laser Scanning Data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2167–2181, 2015.

[11] 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. of IV*, Paris, France, Jun. 2019, pp. 1484–1490.

[12] L. Zhou and Z. Deng, "LiDAR and Vision-Based Real-Time Traffic Sign Detection and Recognition Algorithm for Intelligent Vehicle," in *Proc. of ITSC*, Qingdao, China, Oct. 2014, pp. 578–583.

[13] Z. Deng and L. Zhou, "Detection and Recognition of Traffic Planar Objects Using Colorized Laser Scan and Perspective Distortion Rectification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5, pp. 1485–1495, 2018.

[14] W. Tan, N. Qin, L. Ma, Y. Li, D. Jing, G. Cai, K. Yang, and J. Li, "Toronto-3D: A Large-Scale Mobile LiDAR Dataset for Semantic Segmentation of Urban Roadways," in *Proc. of CVPR - Workshops*, virtual, Jun. 2020, pp. 797–806.

[15] D. Munoz, J. Bagnell, N. Vandapel, and M. Hebert, "Contextual Classification with Functional Max-Margin Markov Networks," in *Proc. of CVPR*, Miami, FL, USA, Jun. 2009, pp. 975–982.

[16] C. Liu, S. Li, F. Chang, and Y. Wang, "Machine Vision Based Traffic Sign Detection Methods: Review, Analyses and Perspectives," *IEEE Access*, vol. 7, pp. 86 578–86 596, 2019.

[17] 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 Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1415–1427, 2016.

[18] 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," *Journal of Expert Systems with Applications*, vol. 89, pp. 286–295, 2017.

[19] 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 Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 5, pp. 1715–1724, 2018.

[20] 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 Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 1, pp. 295–303, 2016.

[21] 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 Journal of Photogrammetry and Remote Sensing*, vol. 113, pp. 106–123, 2016.

[22] A. Vu, Q. Yang, A. Farrell Jay, and M. Barth, "Traffic Sign Detection, State Estimation, and Identification Using Onboard Sensors," in *Proc. of ITSC*, The Hague, Netherlands, Oct. 2013, pp. 875–880.

[23] B. Li, "3D Fully Convolutional Network for Vehicle Detection in Point Cloud," in *Proc. of IROS*, Vancouver, Canada, Sep. 2017, pp. 1513–1518.

[24] 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. of ICRA*, Marina Bay Sands, Singapore, Jun. 2017, pp. 1355–1361.

[25] 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. of CVPR*, Long Beach, CA, USA, Jun. 2019, pp. 12 689–12 697.

[26] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," in *Proc. of CVPR*, Salt Lake City, UT, USA, Jun. 2018, pp. 4490–4499.

[27] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely Embedded Convolutional Detection," *Sensors*, vol. 18, no. 10, pp. 3337–3354, 2018.

[28] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-Based 3D Single Stage Object Detector," in *Proc. of CVPR*, virtual, Jun. 2020, pp. 11 037–11 045.

[29] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: Sparse-to-Dense 3D Object Detector for Point Cloud," in *Proc. of ICCV*, Seoul, Korea, Oct. 2019, pp. 1951–1960.

[30] Federal Highway Research Institute. (2022). Traffic Signs and Symbols, [Online]. Available: `https://www.bast.de/EN/Traffic_Engineering/Subjects/verkehrszeichen/vz-start.html`.

[31] C. Plachetka, J. Fricke, M. Klingner, and T. Fingscheidt, "DNN-Based Recognition of Pole-Like Objects in LiDAR Point Clouds," in *Proc. of ITSC*, Indianapolis, IN, USA, Sep. 2021, pp. 2889–2896.

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg, "SSD: Single Shot Multibox Detector," in *Proc. of ECCV*, Amsterdam, The Netherlands, Oct. 2016, pp. 21–37.

[33] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[34] P. J. Huber, "Robust Estimation of a Location Parameter," *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.