Efficient Object Detection Using Quasi-2D LiDAR Data for Autonomous Vehicles

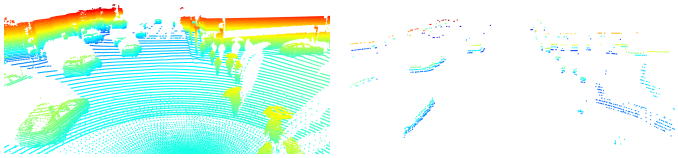
Samanti Das*, Joachim Clemens*, Darshan Ghugare* and Kerstin Schill*
*Cognitive Neuroinformatics Group, University of Bremen, Bremen, Germany
Email: {samanti,clemens,ghugare,kschill}@uni-bremen.de

***Abstract*—Accurate object detection is important for the safe navigation of autonomous vehicles (AVs) in dynamic environments. Conventional AVs rely on high-resolution but bulky, top-mounted 3D Light Detection And Ranging (LiDAR) sensors, which are impractical for scalable production and computationally expensive to process. To address this, we utilize quasi-2D LiDAR sensors, which offer a compact and scalable solution for AVs. However, point clouds generated by these sensors have a narrow vertical field of view and contain a limited number of points compared to 3D LiDARs, which makes object detection significantly more difficult. To overcome this challenge, we transform the quasi-2D point clouds into Bird's-Eye-View (BEV) images and apply four different input encoding strategies. These encodings enhance the spatial and temporal context of the data, making them suitable for efficient deep-learning-based 2D object detection models to perform real-time detection with minimal computational overhead. In contrast to other approaches for 2D LiDARs, we are able to detect not only vehicles but also pedestrians and cyclists. We use a large real-world dataset to compare our proposed input encodings combined with three different state-of-the-art 2D object detection models. Our results demonstrate that these input encodings enhance object detection performance across all models, significantly reducing false positives, improving heading angle prediction, and boosting the detection accuracy of smaller traffic participants.**

I. Introduction

Autonomous vehicles (AVs) are gaining popularity because they have the potential to improve driving comfort and reduce accidents by eliminating human error, which is a major cause of road mishaps [1]. To navigate safely in dynamic and complex environments, AVs should be equipped with advanced perception systems for accurate localization and classification of traffic participants, such as vehicles, pedestrians, and cyclists. This is important for autonomous driving to perform higher-level tasks like object tracking, decision-making, path planning, and motion control.

While cameras offer a lot of semantic information, light detection and ranging (LiDAR) sensors have the advantage of high spatial resolution and accuracy, as well as providing precise distance information. Most popular driverless car systems, manufactured by companies such as Waymo, Cruise, and Argo AI, rely on bulky top-mounted LiDAR sensors that can limit their scalability for mass production. Additionally, these LiDAR sensors have at least 32 layers and generate very dense and rich point clouds that require a considerable amount of computational resources for real-time processing. In contrast, our research vehicle has a compact setup with six LiDARs built into its bumpers to provide comprehensive perception of the surrounding environment, while maintaining a more conventional appearance on the road than existing driverless vehicles. These LiDAR sensors generate quasi-2D point clouds with only 3 layers per scan and a very narrow vertical field of view (FOV) of 3.2°. Consequently, these point clouds contain far fewer points and less information compared to a point cloud from the Waymo system (see Fig. 1).



(a) A point cloud from the Waymo Open Dataset generated by a 128-layer LiDAR mounted on the top of the vehicle.

(b) A point cloud of the same scene from a 4-layer LiDAR mounted in the front bumper.

Fig. 1. Comparison of data obtained by a 128-layer LiDAR with data from a 4-layer LiDAR.

In recent years, deep learning has made significant progress in the field of 2D and 3D object detection [2–7]. 3D LiDAR-based networks rely on high-resolution point clouds, with rich spatial information. Additionally, generating feature maps from 3D point clouds using deep learning models is computationally intensive and requires high processing power and memory.

Our quasi-2D point clouds greatly reduce the amount of information concerning the geometry of the perceived environment, limiting the applicability of sophisticated 3D object detection models. Instead, we propose transforming the point clouds into Bird's-Eye View (BEV) representations and apply existing deep learning-based 2D object detection models. These models are computationally efficient, optimized for grid-structured data, and show good generalization capability in large-scale computer vision tasks. However, our point clouds have a limited number of points, and projecting them onto a 2D plane results in further information loss. Hence, it becomes important to adopt effective input encoding techniques that maximize the information captured from our data, improving detection and localization accuracy. In contrast to existing 2D LiDAR-based methods, this allows us to detect not only vehicle but also smaller objects like pedestrians and cyclists.

In this paper, we develop four different input encoding

strategies that add spatial and temporal information to the BEV images: *Binary Encoding*, *Height Encoding*, *Temporal Encoding*, and *Temporal+Height Encoding*. We apply three state-of-the-art 2D object detection models to the different input encodings: Faster-RCNN [3], RetinaNet [5], and YOLO [4]. A real-world dataset is used to compare the detection performance and runtime of all combinations of input encodings and object detection models. Additionally, we compare our approach against a 3D object detection model, PV-RCNN [6], to provide a comprehensive performance comparison. Finally, we investigate the integration of the proposed approach into an autonomous driving framework.

II. Related Work

Object detection is a fundamental perception task in AVs, which consists of two sub-tasks: localization, which involves estimating the location of traffic participants on the road and classification of those traffic participants (e.g., *Vehicle*, *Pedestrian*, *Cyclist*). In this section, we review the state-of-the-art in LiDAR-based object detection, 2D object detection, and previously proposed input encodings.

A. LiDAR-based Object Detection

Object detection in 3D point clouds is more complex than in 2D images due to their unordered structure. 3D convolutional networks [8, 9] have been used on LiDAR data, but they suffer from slow inference and high computational cost. Voxel-based methods, like VoxelNet [7] and SECOND [10], convert 3D point clouds into dense or sparse voxel grids for CNN processing. Direct processing of raw point clouds and hybrid point-voxel methods for 3D feature learning and region proposal refinement in object detection have been explored in [6, 11, 12].

All the above methods rely on multi-layer 3D LiDAR sensors. In contrast, 2D LiDARs (1 layer) have been used in AVs [13, 14] by converting raw 2D point clouds into pseudo-images for 2D object detection. In [13], Cascade Pyramid RCNN (learning-based) and Hybrid ResNet (classical + learning) for oriented bounding box prediction are introduced. In [14], geometric features using L-shaped keypoints are extracted and a two-stage process for bounding box generation is employed. These methods demonstrate that reliable detection of vehicles can be achieved by using cost-effective 2D LiDARs. However, those approaches are limited to vehicles only, neglecting small traffic participants like pedestrians and cyclists.

B. 2D Object Detection

Regions with Convolutional Neural Networks (R-CNN) [15] revolutionized deep-learning-based object detection, achieving a large improvement over previous methods. It uses a two-stage approach: Selective Search for region proposals, followed by a CNN for classification. However, it is computationally expensive as it performed redundant computations for overlapping proposals. Fast-RCNN [2] improves efficiency by introducing RoI pooling, allowing the CNN to process the entire image instead of individual proposals. Both rely on Selective Search, limiting speed. Faster-RCNN [3] addresses this by replacing it with a Region Proposal Network (RPN), significantly reducing processing time. While accurate, these two-stage detectors have slower inference speeds compared to single-stage methods. To overcome the speed limitations of two-stage detectors, single-stage models like YOLO [4] and SSD [16] were introduced, enabling object detection in a single pass. However, early versions struggle with class imbalance and small object detection. RetinaNet [5] addresses this by introducing Focal Loss, which reduces the impact of easy background samples during training, allowing the network to focus on harder object detections and improving performance.

Axis-aligned bounding boxes, commonly used in the above methods, are unsuitable for accurately determining the orientation and shape of traffic participants in LiDAR-based object detection. Incorporating an orientation angle through oriented bounding boxes improves localization, as shown in methods like [17–20]. Additionally, open-source toolboxes like [21, 22] enable modifying state-of-the-art 2D detectors to support this feature.

C. Input Encodings

Several methods convert 3D point clouds into 2D images by projecting them onto BEV or front-view planes [23–27], enabling the use of 2D CNNs for object detection. Techniques like [10, 23] address height loss during projection by encoding height in grid-based or voxel-based representations. Birdnet [24] proposes a BEV encoding with height, intensity, and density channels, normalizing density to account for sensor variations. Other methods [13, 14] project 3D points to a 2D plane, mapping x, y coordinates and distance to RGB channels. In our approach, we adopt the height encoding. However, we additional introduce a novel temporal encoding that incorporates information from consecutive frames into a single image, enhancing the model's ability to capture dynamic changes.

III. Research Vehicle and System Overview

This section introduces our research vehicle and gives an overview over the relevant components of the autonomous driving system [28]. The research vehicle is a modified Volkswagen Passat GTE and shown in Fig. 2. As discussed already, our setup differs from many self-driving cars by using compact, strategically positioned Scala Gen 1 LiDAR sensors. Three are mounted at the front (right, center, left) and three at the rear, ensuring full coverage while maintaining a conventional vehicle appearance. They operate at $25\,\mathrm{Hz}$, have a 145° horizontal and 3.2° vertical FOV, and capture three of the four layers per scan. Similar sensors are already used in mass production, e.g., by Audi. Unlike top-mounted 360° LiDARs, this setup is less bulky and the modifications of the vehicle are barely visible.

In addition to the LiDAR sensors, the car is equipped with a camera, radio detection and ranging (radar) sensors, an inertial measurement unit (IMU), two global navigation

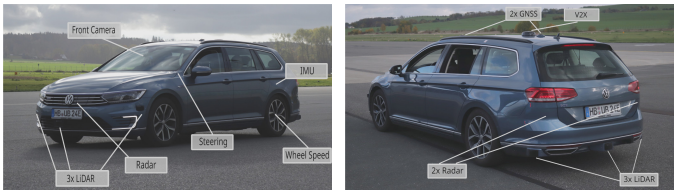


Fig. 2. Front and rear view of our research vehicle with the relevant sensors. (Figure adopted from [29].)

satellite system (GNSS) receivers, as well as wheel speed and steering angle sensors. Processing takes place on an Intel i9-9900K CPU with dual NVIDIA RTX 2080Ti GPUs.

The algorithm proposed in this paper processes the data of each LiDAR sensor separately (in parallel). As additional input, it uses the ego motion of the vehicle, which is provided by a Kalman-filter-based odometry approach [30, 31]. The output of the algorithm, i.e., the detected objects, are further processed by a multi-object tracking system. It uses one Kalman filter per tracked object and performs late fusion of the detected objects from all six LiDAR sensors as well as from camera and radar.

IV. Encoding Strategies and Rotated Object Detection

The setup of our research vehicle with quasi-2D LiDAR sensors poses some challenges for the use of 3D object detection algorithms:

- **Limited Feature Representation:** Quasi-2D point clouds lack sufficient detail for effective feature extraction employed by sophisticated 3D Object Detection models.
- **Model Overfitting:** 3D models are optimized for dense data and they tend to overfit when applied to point clouds containing limited number of points.

To find a trade-off between real-time processing and accurate object detection, we opt to utilize existing 2D object detection models by projecting our LiDAR data into BEV images. This decision is motivated by low point density per scan, the fact that it mainly provides 2D information and the available GPU capacity of our research vehicle. In this section, we first propose different input encodings for mapping scans to images. After that, we discuss different models for detecting rotated objects in those images.

A. Input Encoding Strategies

To enhance the performance of 2D object detection on our LiDAR data, we introduce four input encoding strategies:

1) Binary Encoding: We project the LiDAR data into a BEV representation and assign a value of 1 to the occupied pixels, which correspond to the measured points, and 0 to free space (see Fig. 3a). This encoding provides minimum contextual information about the point cloud and serves as a baseline for our evaluation with respect to the other encoding strategies.

2) Height Encoding: We compensate the loss of elevation information that occurs due to the 3D to 2D projection by incorporating height encoding into the images. Each point's z-coordinate is normalized to $[-1, 2]$ where -1 accounts for road undulations and 2 is the maximum height up to which a distinction appears to be meaningful considering we are focusing on traffic participant. Note that due to the limited vertical FOV and the mounting positions in the bumper, the minimum and maximum values are rarely reached. The normalized values are then mapped to a blue-to-red gradient, with blue representing low elevation and red high (see Fig. 3b).

3) Temporal Encoding: In order to capture temporal correlation between consecutive LiDAR frames, we introduce temporal encoding. First, we transform the previous LiDAR scan to the current scan's coordinate system by taking the mounting pose of the LiDAR sensor and the ego movement of the vehicle into account. Then, we project both the point clouds into BEV images, where the red channel is used for the information from the previous scan, while the green channel is used for the current scan.

This adds temporal information to the image, since colors represent the scene at different points in time (see Fig. 3c):

- Red pixels were occupied in the previous scan but are free in the current scan.
- Green pixels were free in the previous scan but are occupied in the current scan.
- Yellow pixels are occupied in both scans.
- Black pixels are free in both scans.

This encoding enhances the model's ability to infer the direction of motion of the traffic participants, which is essential in autonomous driving. Furthermore, it helps distinguishing between static and dynamic objects, which is not possible in the Binary encoding.

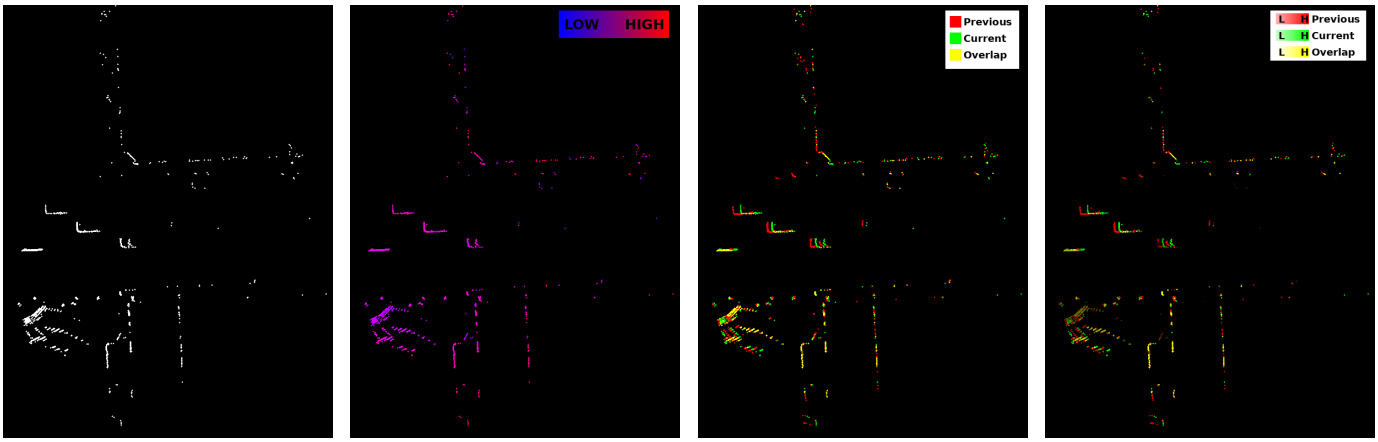
4) Temporal+Height Encoding: In this case, we combine Temporal encoding with Height encoding to create a more informative representation of our data. Here, instead of using uniform colors to encode temporal information, we modulate the pixel intensities based on the height information (see Fig. 3d). This encoding provides more comprehensive context in the images compared to the previous encodings by incorporating both temporal variation between consecutive frames and the associated height information.

B. Rotated Object Detection

As mentioned in Sect. I, we employ three widely used deep-learning based 2D object detectors which are Faster-RCNN, RetinaNet and YOLO. They are applied to the data encoded as described above.

The standard representations of these models can only handle axis-aligned bounding boxes, where the annotations are in the format of $(x_{\min}, y_{\min}, w, h)$. Here $(x_{\min}, y_{\min})$ represent the coordinates of the top-left corner of the bounding box in pixel space and w, h refer to the width and height of the box.

Since, it is important for autonomous vehicles to correctly predict the orientation and shape of the traffic participants, the models need to be able to handle oriented bounding boxes.



(a) Binary Encoding

(b) Height Encoding with a gradient color map.

(c) Temporal Encoding with consecutive frames in a single image.

(d) Temporal+Height Encoding with elevation encoded into intensity of color map.

Fig. 3. Different Input Encodings implemented in this paper.

Those are represented by (x_c, y_c, w, h, α), where (x_c, y_c) represents the center of the bounding box, w, h are the width and height of the box and α denotes its orientation angle. Training with oriented bounding boxes enables the networks to provide more precise localization information. For anchor-based models, we modify the anchor sizes, aspect ratios, etc. in the configuration (as outlined in [21]) to better suit our data. The specifics of each model are described below.

1) Faster-RCNN: This 2-stage model is known for its high accuracy in object detection tasks. Its Region Proposal Network (RPN) effectively generates object proposals that are then refined for precise localization, making it a reliable choice for detecting different traffic participants. Additionally, [13, 23] who adopt similar data encoding strategies, incorporate the Faster R-CNN network in their evaluation studies. We define anchor sizes (as width-to-length ratios) based on the average and maximum dimensions of traffic participants (see Tab. I) as:

$$S = \{1.2,\ 1.8,\ 3.2,\ 5.0,\ 11.5\}. \tag{1}$$

This set ensures that the anchors effectively enclose objects of varying sizes in the dataset. For aspect ratios, we follow the configuration from [3] and set them to $1:1$, $2:1$, and $1:2$. The aspect ratios scale the anchor size S to define the width and height of the anchor boxes Moreover, we set the angle range for the anchors within $(-\pi, \pi)$ to capture the full range of the orientation angles present in our dataset.

2) RetinaNet: This one-stage detector can handle class-imbalances and is suitable for detecting objects at multiple scales [5]. Given the dataset's class imbalance (see Tab. I), where vehicles dominate (73.2%), while pedestrians (25.74%) and cyclists (1.06%) are underrepresented, RetinaNet's Focal Loss can mitigate the bias toward majority classes by focusing on difficult-to-detect instances. It has a multi-scale Feature Pyramid Network (FPN), which is suitable for robust detection of larger vehicles as well as smaller pedestrians and cyclists and has the potential to improve recall for smaller objects.

To detect objects at varying scales and orientations, we employ a Rotated Anchor Generator in our RetinaNet framework similar to [21]. The anchor sizes follow an octave-based scaling approach with a base size of 4 pixels and 4 scales per octave. The aspect ratios and feature map strides for our data are defined as

$$AR = \{1.0,\ 0.5,\ 2.0,\ 2.5\} \tag{2}$$

$$Strides = \{8,\ 16,\ 32,\ 64,\ 128\} \tag{3}$$

Here, *AR* represents the set of aspect ratios (width-to-length) chosen to accommodate different object shapes and (Strides) determine the scale at which anchors are generated.

Similar as for Faster-RCNN, we set the angle range for the anchors to $(-\pi, \pi)$. This configuration ensures effective object detection across all traffic participants, from small pedestrians to large vehicles.

3) YOLO: YOLO models are widely recognized for their speed, making them ideal for real-time applications [22]. YOLO's single-stage architecture allows for efficient prediction of object classes and oriented bounding boxes, making it ideal for processing our data. The latest version, YOLOv11, introduced in 2024 in [22], includes support for oriented bounding boxes and is used in this paper.

V. Dataset and Training

Deep neural networks need large amounts of labeled data for training to perform reliable object detection. However, our research vehicle has not yet collected enough data to effectively train the 2D object detectors. To resolve this, we convert a public dataset such that it resembles our quasi-2D

TABLE I
STATISTICS FOR TRAFFIC PARTICIPANTS IN THE TRAINING SET.

Class	Occurrence (%)	Avg Length (m)	Avg Width (m)	Max Length (m)	Max Width (m)
Vehicle	73.2	4.9	2.13	19.4	7.0
Pedestrian	25.74	0.973	0.88	4.3	2.67
Cyclist	1.06	1.8	0.43	3.5	1.6

LiDAR data for training and evaluating our models. In this section, we first discuss the conversion of the dataset and then describe the specifics of the training itself.

A. Dataset Conversion

Due to the limited availability of suitable (quasi-2D) training data, we selected the popular Waymo Open Dataset [32]. It consists of 230,000 annotated LiDAR frames across 1,150 scenes, captured in urban and suburban environments at different times of the day. This dataset was selected because of its large scale, offering significantly more data than KITTI's [33] 15,000 frames, as well as its high-quality annotations and diverse driving conditions (sunny, rainy, and foggy scenarios). However, the dataset has been recorded with multiple high-resolution 360° LiDARs. To adapt the Waymo point clouds to closely resemble the data from our quasi-2D LiDAR setup, we first generate a 3D occupancy grid map (resolution: $0.1\,\mathrm{m}$) based on the 3D scans. Then we utilize the Bresenham algorithm [34] to perform raycasting in the grid map with the parameters (mounting pose, FOV, resolution) of our LiDAR sensors to simulate a scan of those. In this process, we limit the point cloud range to $100\,\mathrm{meters}$ since our focus is on urban driving, where the close vicinity is most important. Furthermore, beyond that distance the point cloud becomes too sparse for reliable detection.

This transformation reduces spatial resolution by projecting 3D data onto a 2D plane. As a result, vertical detail is lost, and the data is converted from continuous points to pixel-based representations. However, the impact of losing vertical detail is minor due to our sensor's inherently narrow FOV along that axis. Original label accuracy is preserved, ensuring the dataset remains valid for evaluation.

B. Training

We train and evaluate all three models using the transformed Waymo data with 150,000 samples from 798 sequences for training and 40,000 samples from 202 sequences for validation. The training and validation split is preserved from the original Waymo dataset. For the training set, we present the average, maximum, and minimum sizes (length and width) of the traffic participants, along with their occurrence percentages in Tab. I. The traffic participants considered include *Vehicle*, *Pedestrian*, and *Cyclist*.

The training is performed on a NVIDIA RTX A6000 GPU equipped with 48 GB of VRAM memory. All three 2D object detection models are pre-trained on the DOTAv1.0 dataset [35]. We trained the models using the settings recommended by the developers from the official repositories [21, 22], and selected the model checkpoint corresponding to the epoch with the best validation set performance. For training (cf. Sect. IV-B1), we employ the Rotated Faster R-CNN with a ResNet50 backbone from [21]. The model is trained for 12 epochs, each comprising 4,500 iterations, using a batch size of 32. We use the SGD optimizer with momentum of 0.9, a learning rate of 0.005, and weight decay of 0.0001. The Cosine Annealing scheduler is used, which gradually decreased the learning rate to a minimum of 1e-6. The rest of the model specifications are kept the same as in [21].

For training RetinaNet (cf. Sect. IV-B2), we use the pre-trained Rotated-RetinaNet with a ResNet50 backbone from [21]. Similar training strategies are employed as for training the Rotated-Faster-RCNN network.

For training YOLO (cf. Sect. IV-B3), we use the latest pre-trained YOLOv11-OBBn model from [22], where *n* denotes the nano variant. It is the most compact and lightest version of all YOLO models, making it a particularly suitable choice for our vehicle because of the faster inference time and low computational load. The model is trained on a batch size of 32, for 100 epochs as in [22]. We use the SGD optimizer with a learning rate of 0.01, momentum of 0.9, and weight decay of 0.0005. LambdaLR is used as scheduler with an initial learning rate of 0.01 and reducing it linearly to 0.0001 over 100 epochs. We train YOLO with more epochs (100, compared to 12 for FasterRCNN and RetinaNet) because it is a much lighter model and needs more epochs to learn the parameters more effectively.

VI. EVALUATION

The evaluation covers a quantitative analysis of the accuracy, a qualitative comparison of the different encoding methods, and an investigation of the inference time.

A. Quantitative Accuracy

The accuracy is assessed using Average Precision (AP) from [21] with an Intersection over Union (IoU) threshold of 0.5 for each class and overall. For calculating the AP across all the classes, we have done a weighted average depending on the number of instances of a particular class in the validation set. We conduct the evaluation at different distances $(0\text{–}30\,\mathrm{m},\ 30\text{–}50\,\mathrm{m},\ \text{and}\ 50\text{–}100\,\mathrm{m})$ from the ego vehicle and also provide inference time (Inf. time) for each of the models.

The evaluation results of the three 2D object detection models (Faster-RCNN, RetinaNet and YOLO) using the different encoding methods (Binary, Height, Temporal, Temporal+Height) and a 3D detection model (PV-RCNN) are summarized in Tab. II. We can see that Faster-RCNN consistently outperforms RetinaNet and YOLO across all encoding types, particularly for vehicle detection where it achieves the highest AP values. The Temporal+Height encoding yields the best results overall across all the models, particularly improving AP values for pedestrians and cyclists.

For vehicular detection, Faster-RCNN and RetinaNet, being more complex two-stage and anchor-based models respectively, yield higher accuracy even at mid-range distance

TABLE II
EVALUATION RESULTS FOR OBJECT DETECTION MODELS WITH DIFFERENT ENCODINGS AND THEIR INFERENCE TIMES

Model	Encoding	Vehicle AP (IoU=0.5)			Pedestrian AP (IoU=0.5)			Cyclist AP (IoU=0.5)			mAP (IoU=0.5)			Inf. time *
		0-30m	30-50m	50-100m	0-30m	30-50m	50-100m	0-30m	30-50m	50-100m	0-30m	30-50m	50-100m	
PV-RCNN	3D point cloud	56.2	40.8	33.1	21.1	12.3	4.5	25.6	18.4	7.8	45	31.73	24	∼30
Faster-RCNN	Binary	89.5	76.7	58.2	26.5	23.8	11.6	20.7	7.1	3.5	69.4	67.5	56.4	∼40
	Height	90.9	81.5	64.7	**28.3**	25.5	16.1	17.4	6.4	3.6	70.8	71.8	62.8	
	Temporal	90.2	80.1	63.8	26.8	28.3	15.7	36.9	24.2	14.8	70.1	71.2	62	
	Temp.+Height	**91.1**	**82.4**	**66.9**	**28.3**	**31.7**	**16.5**	**38.5**	**28.3**	**15.1**	**71.2**	**73.7**	**65**	
RetinaNet	Binary	88.3	72.8	54.5	24.4	20.3	7.9	21.1	3.4	0.07	67.9	63.7	52.7	∼36
	Height	**89.4**	76.9	58.7	22.6	22.6	**14.8**	13.4	10.3	0.5	68	67.5	56.9	
	Temporal	89	76.9	60.5	23.9	26.5	14.6	38.8	20.8	**11.1**	68.4	68.3	58.8	
	Temp.+Height	**89.4**	**78.5**	**61.6**	**25.6**	**29.9**	8.9	**40.1**	**25.7**	10.3	**69.3**	**70.2**	**61.6**	
YOLO	Binary	84.7	64.7	43.8	17.1	13.9	1.7	14.7	1.4	0	63.1	55.9	42.2	∼12
	Height	**88.1**	70.8	51.2	25.2	20.3	**4**	16.3	3.2	0	68	62	49.4	
	Temporal	86.5	68.3	44.9	20.9	12	0.8	22.5	3.2	0	65.6	58.6	43.2	
	Temp.+Height	**88.1**	**72.3**	**51.8**	**29.5**	**24.7**	2.3	**35.2**	**12.8**	**1.8**	**69.5**	**64.1**	**49.9**	

* Average inference time per image on the full validation set (in milliseconds).

(30-50m). In contrast, YOLO, which is optimized for real-time performance, exhibits performance drop at mid to long distances.

For pedestrian detection, we observe an interesting case that the accuracy for Faster-RCNN and RetinaNet improves in mid range distances for the temporal encodings. At close range, ground reflections introduce significant noise, and due to the BEV projection, pedestrians often blend into nearby objects and ground. The effect gets aggravated in the temporal encodings as we use consecutive overlapping frames, making it difficult for the detectors to distinguish actual pedestrian pixels from background noise. However, in mid ranges, ground reflection is much less, leading to better accuracy.

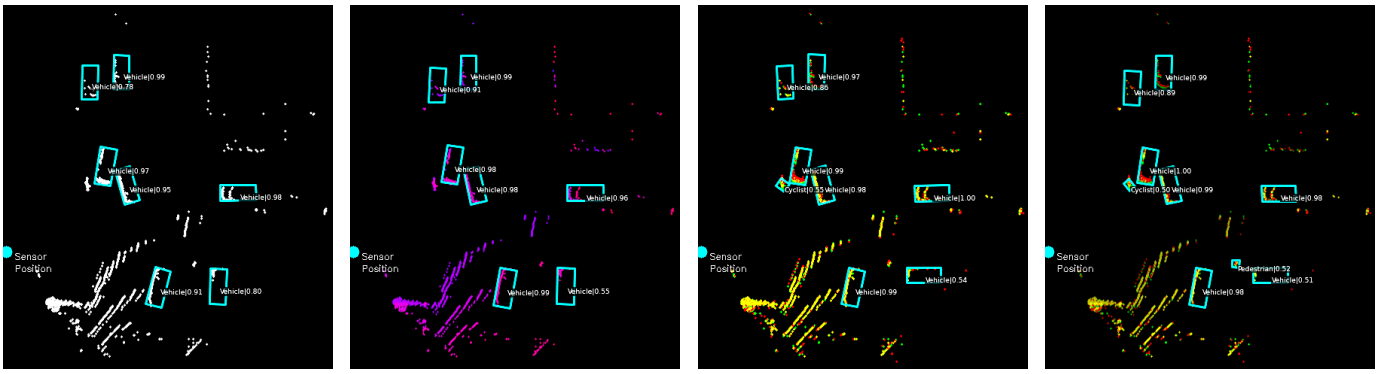
For cyclist detection, we see a large jump in accuracy with the temporal encodings across all the models. This is because cyclists move at higher speeds compared to pedestrians and the temporal encodings are able to capture motion patterns over consecutive frames. By leveraging this information, the models learn to distinguish the cyclists more accurately, indicating the efficiency of our encoding method. We compare our approach with the state-of-the-art 3D detector PV-RCNN by directly feeding it the quasi-2D point cloud, using the same training configurations as in [36]. The results indicate that the 3D detector struggles to extract meaningful features from our sparse 4-layer LiDAR data, resulting in inferior performance compared to 2D detectors. This gap is due to our naive application of 3D models without incorporating temporal information. Furthermore, the limited vertical FOV of our sensor produces limited vertical data, diminishing the usefulness of depth cues. Given that our pipeline focuses on detecting objects within a 2D plane for subsequent tracking, 2D detectors are more suitable for this task. It is important to note that while the 3D model's inference time is comparable to that of 2D detectors in our setup, processing a complete 3D point cloud would substantially increase computational complexity.

B. Qualitative Analysis of Encoding Methods

For the qualitative comparison, we use the best-performing model (Rotated Faster-RCNN) and analyze the results across all encoding methods. The detections for one scene are shown in Fig. 4. We can observe that vehicular detection is consistently good in all the models because vehicles maintain a well-defined geometric structure even after projecting point clouds into BEV images. However, Temporal encoding becomes particularly beneficial when vehicles have fewer pixels associated with them. This advantage becomes evident in the bottom-rightmost detection, which is incorrect for both the binary- and height-encoded images. However, with Temporal encoding, the model leverages motion cues from the past frame, allowing it to better infer the shape as well as the orientation of the vehicle. Both temporal encoding methods successfully detect cyclists in the image, while the Temporal+Height encoding goes a step further by accurately identifying smaller objects like pedestrians.

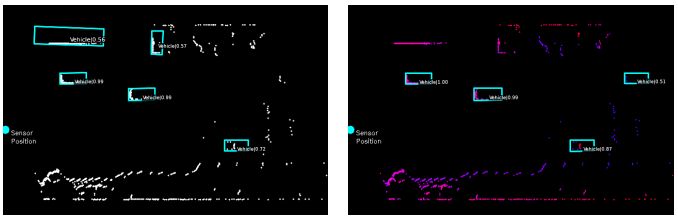
In Fig. 5, we further illustrate the importance of Height encoding over Binary encoding. Without taking height into account, the model mistakenly classifies background structures as vehicles due to the absence of elevation information (see Fig. 5a). However, with Height encoding, the model can accurately distinguish between background elements and actual traffic participants. This reduces false positives and enhances overall detection accuracy. A similar improvement can be observed when comparing Temporal encoding with Temporal+Height encoding, where incorporating height information further refines object differentiation and reduces misclassification.

The effectiveness of temporal encoding for learning the heading angle of traffic participants is illustrated in Fig. 6. By encoding the current frame in the green channel and the previous frame in the red channel, the model can infer the direction of movement. The predicted bounding box orientation allows us to distinguish the front from the rear of the vehicle, providing insight into its heading direction. In contrast, the Binary encoding lacks motion information,



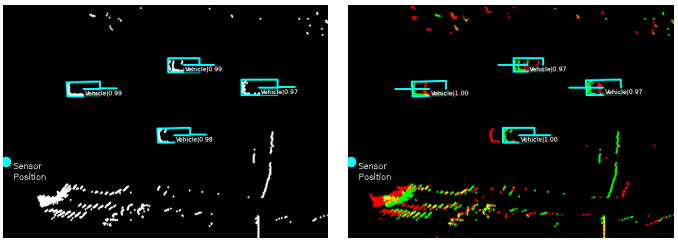
(a) Binary Encoding (b) Height Encoding (c) Temporal Encoding (d) Temporal+Height Encoding

Fig. 4. Detection results across different input encodings from the Faster-RCNN model



(a) Binary Encoding (b) Height Encoding

Fig. 5. Height encoding enhances detection accuracy by effectively differentiating vehicles from background elements, thereby reducing false positives.



(a) Binary Encoding (b) Temporal Encoding

Fig. 6. Temporal encoding enables the model to learn heading angles by capturing motion between frames, unlike binary encoding which lacks motion cues.

preventing the model from reliably assigning heading angles. The same applies to the Height encoding, which is not shown here.

These qualitative results highlight the necessity and suitability of incorporating temporal and height information into the encoding process, as they have a noticeable impact on improving detection accuracy and reducing false positives.

C. Inference Time

Among all the models evaluated, YOLO is selected as the preferred model for our research vehicle, as it offers the best trade-off between inference speed and detection accuracy. While Faster-RCNN and RetinaNet exhibit better performance in certain scenarios, YOLO's real-time efficiency makes it the most practical choice for deployment. The YOLO model has been incorporated into our C++ fusion framework in order to evaluate the inference time of the detection model within the perception pipeline of the research vehicle. To ensure efficient inference and seamless C++ compatibility, the model is exported in TorchScript format. Along with deploying the model, a comprehensive pre-processing and post-processing pipeline is implemented in C++, encompassing the encoding of LiDAR into images and the transformation of detected bounding boxes from image coordinates to the odometry coordinate system, which is used by the object tracking algorithm. To evaluate the performance, we measured the execution time of each stage within the pipeline. The timing analysis is conducted by running the pipeline on 1,000 LiDAR scans using a system configuration identical to the research vehicle.

TABLE III
EXECUTION TIME FOR PERCEPTION PIPELINE USING YOLO MODEL TRAINED ON TEMPORAL ENCODED IMAGES.

Pipeline Stage	**Execution Time** *
Pre-processing	∼1
Tensor creation and transfer (CPU to GPU)	∼2
Detection model inference	∼3
Post-processing	Immeasurable (∼0)
Total Execution Time	∼6

* Average execution time per image on the full validation set (in milliseconds).

Tab. III provides a summary of the results. Comparing the Python-based detection model to its TorchScript version revealed a significant reduction in inference time. The TorchScript model achieves an inference speed of approximately ∼3 ms, whereas the Python-based model requires around ∼12 ms for detection. Additionally, the total pipeline execution time (∼6 ms) ensuring real-time perception in autonomous systems.

VII. CONCLUSION

In this paper, we introduced a novel object detection approach utilizing quasi-2D LiDAR sensors, providing a compact

and scalable alternative to traditional 3D LiDAR systems and leveraging 2D convolutional networks for object detection. By transforming point clouds into BEV images and introducing four different input encoding strategies (Binary, Height, Temporal, Temporal+Height), we enhanced spatial and temporal information in our data. We compared the encodings by utilizing three different state-of-the-art object detection algorithms (Faster-RCNN, RetinaNet, YOLO) and applied them to a real-world dataset containing vehicles, pedestrians and cyclists. Additionally, we compare our method to a 3D detector (PV-RCNN) by using the full quasi-2D point cloud as input. Our results show that our approach outperforms the 3D detector in accuracy. Specifically, Height encoding reduces false positives over Binary encoding by leveraging elevation data. Furthermore, the Temporal and Temporal+Height encodings enable accurate heading angle estimation and significantly improve detection performance, especially for cyclists. These encodings contribute to robust object detection and have already been integrated into our research vehicle's perception pipeline by using YOLO for its balance of speed and accuracy.

As a next step, we plan to incorporate additional intermediate frames to improve pedestrian detection, as their slower movement requires finer temporal resolution. Also, we plan to remove ground points from the point cloud before projecting them into BEV images, reducing noise from ground reflections to improve short-range accuracy of small traffic participants. We also aim to evaluate the impact of integrating temporal neural network blocks, such as RNNs or LSTMs, to retain information from previous frames. Additionally, we intend to explore the use of lightweight backbones for the computationally intensive detectors to find the optimal balance between accuracy and inference speed for real-world deployment.

ACKNOWLEDGMENT

This research was partially funded by German Aerospace Center (DLR) Space Administration with financial means of the German Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag, projects "MUTIG-VORAN" (grant No. 50 NA 2202A) and "Safety Control Center" (grant No. 50 NA 2302B). We would also like to thank Henning Zimmerman and Simon Kaemena for their assistance and support in this research.

REFERENCES

[1] T. S. Combs, L. S. Sandt, M. P. Clamann, and N. C. McDonald, "Automated vehicles and pedestrian safety: Exploring the promise and limits of pedestrian detection," *American Journal of Preventive Medicine*, vol. 56, no. 1, pp. 1–7, 2019. [Online]. Available: https://doi.org/10.1016/j.amepre.2018.06.024

[2] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[5] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[6] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[7] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[8] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 922–928.

[9] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1513–1518.

[10] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/10/3337

[11] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[12] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li, "PV-RCNN++: Point-voxel feature set abstraction with local vector representation for 3D object detection," *International Journal of Computer Vision*, vol. 131, no. 2, pp. 531–551, 2023.

[13] G. Chen, F. Wang, S. Qu, K. Chen, J. Yu, X. Liu, L. Xiong, and A. Knoll, "Pseudo-image and sparse points: Vehicle detection with 2D LiDAR revisited by deep learning-based methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 12, pp. 7699–7711, 2021.

[14] T. Zou, G. Chen, Z. Li, W. He, S. Qu, S. Gu, and A. Knoll, "KAM-Net: Keypoint-aware and keypoint-matching network for vehicle detection from 2-D point cloud," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 207–217, 2022.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed,

C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37.

[17] X. Yang, J. Yang, J. Yan, Y. Zhang, T. Zhang, Z. Guo, X. Sun, and K. Fu, "SCRDet: Towards more robust detection for small, cluttered and rotated objects," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 8231–8240.

[18] J. Ding, N. Xue, Y. Long, G.-S. Xia, and Q. Lu, "Learning RoI transformer for oriented object detection in aerial images," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2844–2853.

[19] X. Yang and J. Yan, "Arbitrary-oriented object detection with circular smooth label," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*. Springer, 2020, pp. 677–694.

[20] W. Qian, X. Yang, S. Peng, J. Yan, and Y. Guo, "Learning modulated loss for rotated object detection," in *AAAI conference on artificial intelligence*, 2021, pp. 2458–2466.

[21] Y. Zhou, X. Yang, G. Zhang, J. Wang, Y. Liu, L. Hou, X. Jiang, X. Liu, J. Yan, C. Lyu, W. Zhang, and K. Chen, "MMRotate: A rotated object detection benchmark using PyTorch," in *30th ACM International Conference on Multimedia*, 2022.

[22] G. Jocher, J. Qiu, and A. Chaurasia, "Ultralytics YOLO," https://github.com/ultralytics/ultralytics. [Online]. Available: https://ultralytics.com

[23] S. Wirges, T. Fischer, C. Stiller, and J. B. Frias, "Object detection and classification in occupancy grid maps using deep convolutional networks," in *21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3530–3535.

[24] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. De La Escalera, "BirdNet: A 3d object detection framework from LiDAR information," in *21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3517–3523.

[25] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3D object detection and tracking," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 11 784–11 793.

[26] M. Ren, A. Pokrovsky, B. Yang, and R. Urtasun, "SBNet: Sparse blocks network for fast inference," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[27] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[28] A. Folkers, C. Wellhausen, M. Rick, X. Li, L. Evers, V. Schwarting, J. Clemens, P. Dittmann, M. Shubbak, T. Bustert, G. Zachmann, K. Schill, and C. Büskens, "The OPA3L system and testconcept for urban autonomous driving," in *25th IEEE International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 1949–1956.

[29] C. Wellhausen, "Sensor fusion in localization, mapping and tracking," Ph.D. dissertation, Cognitive Neuroinformatics, University of Bremen, Bremen, 2024.

[30] J. Clemens, C. Wellhausen, T. L. Koller, U. Frese, and K. Schill, "Kalman filter with moving reference for jump-free, multi-sensor odometry with application in autonomous driving," in *23rd International Conference on Information Fusion (FUSION)*. IEEE, Jul. 2020.

[31] J. Clemens and C. Wellhausen, "The square-root unscented and the square-root cubature Kalman filters on manifolds," *Sensors*, vol. 24, no. 20, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/20/6622/pdf

[32] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[33] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3292–3310, 2023.

[34] J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing," *Proceedings of EuroGraphics*, vol. 87, 1987.

[35] J. Ding, N. Xue, G.-S. Xia, X. Bai, W. Yang, M. Yang, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Object detection in aerial images: A large-scale benchmark and challenges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[36] OpenPCDet Development Team, "OpenPCDet: An open-source toolbox for 3D object detection from point clouds," https://github.com/open-mmlab/OpenPCDet, 2020.