2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Action-By-Detection: Efficient Forklift Action Detection for Autonomous Mobile Robots in Warehouses

Alexander Prutsch[1], Horst Possegger[1] and Horst Bischof[1]

***Abstract*— Understanding actions of other agents increases the efficiency of autonomous mobile robots (AMRs) since they encompass intention and indicate future movements. We propose a new method that allows us to infer vehicle actions using a shallow image-based classification model. The actions are classified via bird's-eye view scene crops, where we project the detections of a 3D object detection model onto a context map. We learn map context information and aggregate temporal sequence information without requiring object tracking. This results in a highly efficient classification model that can easily be deployed on embedded AMR hardware. To evaluate our approach, we create new large-scale synthetic datasets showing warehouse traffic based on real vehicle models and geometry.**

I. INTRODUCTION

For autonomous mobile robots (AMRs) the actions of other traffic participants are an important information source to proactively plan the actions and path of the robot. In today's intralogistics, AMRs are frequently used in mixed-traffic scenarios, where they operate alongside with human-driven forklifts. For highly efficient and safe usage of AMRs in mixed traffic, it is not sufficient that AMRs only detect other vehicles as obstacles. It rather requires advanced perception systems which enable an AMR to understand actions of nearby traffic participants.

Current AMRs frequently use 3D LiDAR scanners and 3D object detection models for basic perception tasks like obstacle detection, detection of pallet positions, and localization, but they are not equipped with systems for action perception. We propose a new method to infer the actions of vehicles based on the output of a 3D object detection model and map context data. In our work, we focus on inferring the actions of human-driven forklifts in the vicinity of an AMR.

Assuming that a 3D object detection system is already used on the AMR, our approach adds only a little computational overhead but provides the robot with a strongly improved understanding of different traffic situations. Consequently, the robot can adapt its driving path and behavior based on the actions of other traffic agents. This means, for instance, proactive stopping before narrowings or leaving more space for vehicles operating at a rack. The explicit detection of vehicle actions also has an upside over using trajectory prediction approaches, since in practice AMRs ought to react differently to forklifts, which do not move, depending on their particular action.

Compared to autonomous driving on roads, traffic in intralogistics brings additional challenges because it is more volatile. Driving paths are only loosely bound to traffic lanes, and there are fewer traffic rules. Additionally, vehicles in warehouses are highly agile and versatile due to their wheel geometry. Properly reacting to different warehouse traffic actions is important, because deadlocks caused by AMRs can lead to significant operating delays and financial losses.

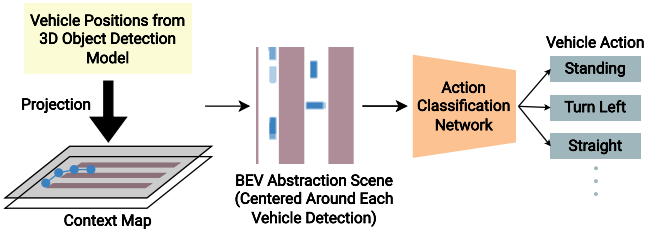

Fig. 1. Overview of our action-by-detection approach. Historic and current vehicle positions are projected onto a context map. Subsequently bird's-eye-view image crops are used to classify the vehicle actions.

We address the efficient action detection task as a bird's-eye view (BEV) image classification problem as illustrated in Figure 1. First, we project vehicle positions detected over a predefined time period onto a simplified warehouse map. Subsequently, we generate image crops centered around each vehicle of interest in the current frame. Then, we use a 2D CNN to classify the action shown on the image crop. Hence, our model jointly learns temporal sequence and spatial context information.

Our approach does not require motion compensation for raw point cloud data, as only the detections must be transformed to a warehouse-global coordinate system. It is also tracking-free, as the 2D CNN implicitly combines the positional information from multiple input time steps-based on our input encoding. Our main advantage over video-based action detection approaches is a significantly less computationally expensive architecture compared to, *e.g.*, 2D CNN with LSTM-based [1], [2], two-stream [3], [4] or 3D CNN-based [5], [6] approaches. Additionally, we directly get action labels matched to 3D positions and do not require to project 3D object detections onto 2D frames or vice-versa.

We conduct extensive evaluations on synthetic large-scale datasets, based on real warehouse traffic scenarios. These show that our approach is able to reliably detect vehicle actions on clean and noisy data. In detailed ablation studies, we demonstrate that i) both spatial context and temporal sequence information are key factors for action classification, ii) our model works well when using the noisier output of a 3D object detector, and iii) that the required context map can be alternatively obtained by a simple BEV reduction of the raw point cloud data. In particular, our approach is detector-

[1] Alexander Prutsch, Horst Possegger and Horst Bischof are with the Institute of Computer Graphics and Vision, Graz University of Technology. Corresponding author: alexander.prutsch@tugraz.at

agnostic and requires no preprocessing of point cloud data. As a result, one can simply utilize a standard 3D object detection model, which generally can be used for diverse tasks on the robot, for processing the raw LiDAR data. In summary, our main contributions include:

- We propose an action-by-detection approach for improved traffic understanding of AMRs. It solves the vehicle action detection task via BEV image classification.
- We show that given a context map and 3D object detections, we are capable of classifying vehicle actions using only a single shallow 2D CNN, which learns temporal and spatial context. Additionally, we demonstrate that the initial warehouse map can be replaced by a simple point cloud projection.
- We provide a detailed model evaluation using oracle data/predictions from a 3D object detection model and an ablation study to highlight our model capabilities.

II. RELATED WORK

A. (Vehicle) Action Detection

Generally, the main focus of vision-based action detection research is on human actions rather than the actions of vehicles in traffic. Vice versa, common multi-modal autonomous driving datasets, like KITTI [7], WAYMO [8], nuScenes [9], and Argoverse 2 [10], do not feature action annotations.

Using these dataset, there is extensive research on the prediction of vehicle motion and trajectories [11], [12], [13], [14], [15] to improve the efficiency and safety of autonomous driving cars. While the task is related to our work, the explicit detection of vehicle actions has an upside for warehouse applications: It is often not sufficient to evaluate if a vehicle trajectory interferes with the AMR path. AMRs in intralogistics should rather consider the actions of nearby vehicles, in order to avoid blocking these (*e.g.* in narrow aisles).

Casas *et al.* [16] proposed a method to learn vehicle intent directly from raw LiDAR point cloud data. Their method, which they evaluated on a private dataset, jointly detects vehicle positions and predicts their intents. Using a custom, raw point cloud-based detection approach, which directly incorporates the detection task into the framework, however, limits the re-usability, since it cannot be applied on top of existing perception pipelines.

Additionally, they have to perform motion compensation using raw point cloud data. In contrast to our approach, they also use a two-stream architecture (one stream processes BEV LiDAR data and the other the context map) to infer the intents. We utilize a single network to process a task-tailored input representation, which only keeps the relevant information and reduces network complexity. Furthermore, they stack multiple time frames in the channel dimensions, which makes their approach less extensible w.r.t. varying the temporal context dimension.

Two datasets that deal with the detection of vehicle actions are the TITAN dataset [17] and the ROAD dataset [18]. The TITAN dataset features video and odometry data but does not include any LiDAR scans. The ROAD dataset is based on the multi-modal Oxford RobotCar Dataset (OxRD) [19]. It extends the OxRD dataset by adding 2D bounding box, action, and event annotations. The authors also provide a baseline model for detecting vehicle actions and road events, where they use RGB images from the OxRD dataset as an input source. This also aligns with the vehicle action detection approach presented in [17] and state-of-the-art work on human action detection, where video data is most frequently used as an input data modality, *e.g.* [20], [5], [21]. Since neither the OxRD nor the ROAD dataset feature 3D vehicle positions annotations, they are not suitable for evaluating our approach on a different problem domain.

B. Bird's-Eye View Scene Representations

Generally, bird's-eye view (BEV) representations of traffic scenes are a widely-used input representation for different perception tasks in autonomous driving, such as trajectory prediction, object detection, and map segmentation. For trajectory prediction, a process for scene rasterisation into multichannel BEV images was proposed by [22] and subsequently also used in trajectory prediction approaches like MTP [11] and CoverNET [12]. Compared to trajectory prediction approaches, we do not require object tracking which makes our system more efficient and robust. A major benefit for intralogistics is that we enable the AMR to react differently to vehicles depending on their action and not only their future path.

Recent work continues to use BEV representations with powerful models for the downstream perception task. For example, Transformers have been investigated with the BEVFormer [23] model which fuses multi-view camera images to a BEV representation that can be used for different tasks. The fusion of different sensor modalities into a unified BEV scene representation has been investigated by BEVFusion [24]. Subsequently, the authors show that their approach can be successfully used for different downstream tasks and outperforms state-of-the-art methods at 3D object detection on the nuScenes [9] dataset.

C. Forklift Activity Recognition

While there are studies on recognizing forklift actions and activities [25], [26], these do not target robotics or computer vision. Instead, they process data sources such as CAN bus data and RFID tags and focus on intelligent warehouse monitoring or determining maintenance cycles. To the best of our knowledge, we are the first to propose a LiDAR-based vehicle action detection approach for AMRs in warehouses.

III. DETECTING WAREHOUSE VEHICLE ACTIONS

Our approach requires only a global warehouse map and vehicle pose predictions from the current and previous LiDAR frames, without associated trajectory information. Our goal is to predict a discrete action label for each vehicle present in the current LiDAR frame. To achieve this, we first generate a task-specific bird's-eye-view (BEV) input representation and subsequently use an image classification network for inference.

A. Input Encoding

Our approach utilizes two main characterizations for extracting scene information: the past driving path of a vehicle and the spatial context where a maneuver is performed. We encode both information into a single bird's-eye view image to obtain a compact input representation that can be processed by state-of-the-art image processing models.

To model the spatial context, we utilize a schematic warehouse map with a pre-defined pixel grid size. Each semantic class is represented by a distinct color to encode the map information as RGB image. For our use case, the most important map feature is the location of racks, but additional features like block storage locations or charging stations can be easily integrated into this representation. Our evaluations also demonstrate how spatial context can be derived from a bird's-eye view projection of raw point cloud data.

To model the vehicle paths, we use a discrete set of vehicle poses. Therefore, we utilize the bounding box centers and orientations predicted by a 3D object detection model. We use the predictions of the current LiDAR frame (keyframe) and combine them with predictions on $N-1$ historic point clouds sampled over the last H seconds. Both N (number of frames in the temporal context) and H (window size) are hyperparameters of our approach. Increasing the number of frames N adds a more detailed path but requires more frequent object detection inference. Increasing the window size H adds more temporal context. A window size of $H = 0$ denotes only using the detections in the keyframe and thus, this case only comprises spatial but no temporal context.

Based on the object centers and heading angles, we render schematic vehicles (*i.e.* fixed-size rectangles) onto our warehouse grid map. The predictions are converted from the local robot coordinate system to a warehouse-global system based on the robot localization data to compensate for the ego-motion. Inspired by fading trajectories in forecasting approaches, *e.g.* [22], we leverage color fading to encode temporal information. In particular, we render the predictions of the current keyframe at time t_c with an opacity of $\alpha = 1.0$. For each previous time step, we linearly reduce α such that for t_{c-H} it reaches a predefined minimum $\alpha_{\text{min}} > 0$.

Since we do not require object trajectories, but instead are interested in implicitly capturing the traffic context in the vicinity of the AMR, we use the same color and style to plot each box (vehicle). Thus, we are not affected by tracking errors and are more robust against missing object detections. In this representation, temporal context is implicitly captured by the spatial relations of boxes with different opacities. By using fixed-size rectangles instead of projected bounding boxes, we further cancel out potentially noisy bounding box estimates which might otherwise introduce a shape bias.

After projecting the current and historic predictions onto the global warehouse map, we generate crops centered at each vehicle prediction in the key frame. This proposal generation yields a BEV image crop for each vehicle with up to N positions for the target vehicle, global context, and potentially nearby other vehicles.

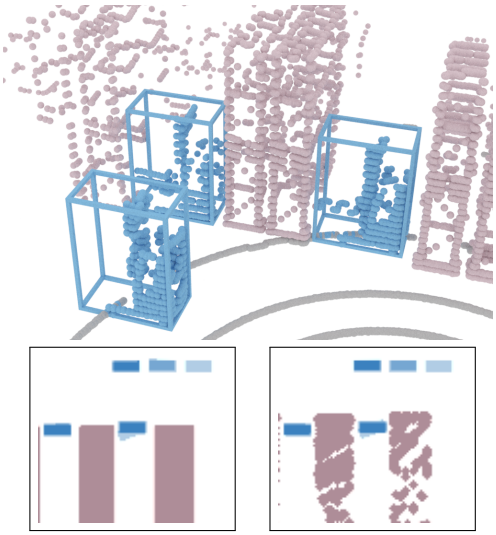

Fig. 2. Input representations for a warehouse scene where two forklifts perform *load handling* and one *drives* in the pre-zone. **Top:** LiDAR point cloud with ground truth vehicle bounding boxes. **Bottom left:** corresponding input representation using map context. **Bottom right:** input representation using the point cloud projection as context.

Figure 2 shows two exemplary input crops. The crop on the right side shows our alternative context, which we use for experiments where we do not utilize a warehouse map. There we project all points of a raw point cloud which are above a predefined height threshold onto a 2D plane. As a result, we get an indication where high objects (in warehouses this mainly corresponds to racks) are located.

B. Classification Model

After generating the input crops, we use a CNN-based image classification network to infer the action shown on a crop. Our main focus is to establish a highly efficient model that can easily be deployed on AMR hardware. Thus, we resort to widely-established and rather simple, but computationally-efficient backbone models. In particular, we experiment with ResNet-18 and ResNet-50 backbones [27]. For the classification head, we use two fully connected layers (*i.e.* 1000×8 and $8 \times n_{\text{classes}}$) and a softmax layer.

IV. EXPERIMENTAL SETUP

A. Synthetic Data Generation

For our experiments, we use synthetic data generated using NVIDIA Omniverse. We implemented a custom simulation setup featuring three different forklifts models based on real-world CAD data. The vehicles are controlled using a Stanley controller [28], which means that the driving of the virtual forklifts is based on the actual wheel geometry of the vehicles. As a result, we achieve realistic vehicle trajectories and behavior in our simulation.

Our virtual warehouse environments are designed based on the layouts and traffic flow of real-world warehouses, which we coordinated with an intralogistics partner company. By incorporating domain expert knowledge, we ensure our scenes and vehicle movements reflect the complexity of real-world scenarios. The environments feature a rack pre-zone,

multiple wide-aisle racks and different distractor objects, which are common in warehouses. The racks are filled with different kinds of cargo. For recording the data, we use virtual LiDAR sensors within NVIDIA Omniverse. We post-process our LiDAR data by filtering ground plane points, adding additive uniform noise and random point drop-out.

Overall, we use 17 driving and 3 static (parked) vehicles in our simulation setup, recorded using 15 statically placed LiDAR sensors. Driving vehicles transport cargoes between randomly selected storage locations in the warehouse. Since we transform our detections into a global warehouse coordinate system, the use of static sensors for evaluation, rather than mobile sensors as in production, has minimal impact on our approach (it only influences the object detection part due to different viewing angles and occlusions). By placing the sensor statically, we can ensure that the sensors always record relevant areas of the warehouse with a lot of traffic.

The annotations exported from our simulation comprise 3D bounding boxes, vehicle class labels, and action labels. We primarily use three action classes: *standing*, *driving* and *load handling*. We also evaluate our approach for more fine-grained classification tasks, where we additionally distinguish *turn left* and *turn right* (Details in Sec. IV-B).

For our experiments, we use two sets of raw data that show different recording sequences. The sequences are recorded in two different warehouse environments. Table I shows an overview of our raw data statistics. The first dataset is used as a train (30% of data) and validation set (70%) of data. We opted to use a larger validation set since preliminary experiments showed that using only 30% of the data is sufficient for training our models. We split the data depending on the recording sensor rather than a temporal sequence split in order to reduce correlation between the train and validation sets. The second dataset is exclusively used as a test set.

B. Datasets Based on Oracle Data

Based on the raw sequences, we generate different datasets (*e.g.* by varying sequence length and context information) to evaluate our action detection approach. As schematic warehouse map, we use a binary mask which encodes for each pixel location if there is a rack. First, we use the annotations exported from our simulation to generate datasets. Using the annotations as a data oracle, we can evaluate our model by assuming that we have a perfect object detection model.

We subsample our data to one key frame per second and use a step size of 3 seconds to generate the historic path context. Given the driving speed of forklifts, this leads to coherent traces while keeping the plotting effort low. For each virtual sensor box location, we filter the ground-truth annotations according to the field-of-view and range of our virtual LiDAR sensor[1].

Table II shows statistics for our baseline datasets based on oracle data. For additional results, we also extend our set of three class labels, which is generated in our simulation setup, to five labels by post-processing the annotations belonging to class *driving*. We want to distinguish between driving straight and left or right turns. To implement this, we utilize a sliding window approach (0.5s time window), and switch labels from driving to *turn* if the angle change in this time window is larger than a vehicle-specific threshold value. For our baseline train/val sequence, this means that out of the 126,845 driving labels, 10,855 are determined as *turn left* and 4,647 as *turn right*. For our test sequence, we have 6,598 *turn left* and 3,160 *turn right* actions.

[1]Following typical specifications of sensors commonly used for robotic applications, we set the maximum range to 34 m and the field-of-view to 180°(we are only interested in vehicles in driving direction of our AMR).

TABLE I

OVERVIEW OF OUR RAW RECORDINGS USED FOR DATASET GENERATION. WE USE 15 SENSORS TO RECORD DATA WITH A FRAME RATE OF 10 HZ.

	Frames	Duration	Frame Rate	Labels
Train/Val	432,000	48 minutes	10 Hz	2,290,186
Test	315,000	35 minutes	10 Hz	2,175,427

TABLE II

OVERVIEW OF OUR BASELINE DATASETS BASED ON ORACLE DATA.

		Labels			
	Key Frames	Standing	Driving	Load Handling	Overall
Train	14,370	15,521	46,446	22,838	84,805
Val	27,211	28,049	80,399	33,756	142,204
Test	31,389	23,227	117,118	29,538	169,883

C. Datasets using 3D Object Detection Model Predictions

To demonstrate the performance using a real perception system, we also evaluate our model using the output from a 3D object detection model. Since our approach follows an action-by-detection paradigm, the performance of our model is directly linked to the performance of the object detection model. If the model recall is low, our input crops would only feature sparse vehicle traces and generally lack vehicle detections for which an action can be inferred. Having low model precision leads to very noisy results and action inference for ghost vehicles. Additionally, the heading angle accuracy has a strong influence.

As a 3D object detection model, we use PointPillars [29]. We chose this approach over more recent approaches due to its computational complexity. We trained a custom model on data from our train/val sequence. On our test set, the model achieves a mAP@0.5 of 52.30 (mAP@0.7 of 50.14).

To match our action labels with the predictions from the object detection model, we iterate over all ground-truth annotations and assign the closest prediction (*Euclidean* distance between ground-truth and prediction center) as the new pose for this annotation. If no prediction is available within a threshold distance (1.5 meters), we skip this annotation. In our evaluation, we treat these cases as misclassification.

D. Model Parameters and Training

For our input encoding, we choose a crop size of $97 \times 97 \times 3$ which corresponds to a spatial resolution of 14.24×14.24 meters. We train our BEV image classification model using a batch size of 64, cross-entropy loss, and stochastic gradient descent (learning rate set to 0.01) as optimizer. For training data augmentation, we use random rotation ($\pm 90°$) and random flips (horizontal and vertical).

TABLE III

CLASSIFICATION ACCURACY USING THE MAP CONTEXT, A TIME FRAME OF $H = 6$ S, AND ORACLE VEHICLE POSITIONS WITHOUT NOISE.

	Validation Set					Test Set				
Backbone	Standing	Driving	Load Handling	Overall	Δ	Standing	Driving	Load Handling	Overall	Δ
ResNet-18	0.9237	0.9753	0.9546	0.9602		0.9005	0.9636	0.9547	0.9534	
ResNet-50	0.9239	0.9700	0.9649	0.9597	-0.0005	0.9013	0.9549	0.9633	0.9490	-0.0044

TABLE IV

ABLATION RESULTS FOR A RESNET-18 BACKBONE USING NOISY DATA (*Noise* INDICATES THE FRACTION OF NOISY SAMPLES), DIFFERENT TIME FRAMES, AND DATA WITHOUT CONTEXT. THE PERFORMANCE DIFFERENCE Δ IS WITH RESPECT TO THE BASELINE IN TABLE III. **BOLD** SETTINGS INDICATE CHANGES TO THE BASELINE CONFIGURATION.

			Validation Set					Test Set				
Context	H (s)	Noise	Standing	Driving	Load Handling	Overall	Δ	Standing	Driving	Load Handling	Overall	Δ
map	6	**100%**	0.9230	0.9691	0.9171	0.9474	-0.0128	0.8992	0.9663	0.9256	0.9497	-0.0037
map	6	**40%**	0.9240	0.9654	0.9373	0.9505	-0.0097	0.9019	0.9553	0.9375	0.9448	-0.0086
map	**9**	-	0.9249	0.9761	0.9619	0.9626	+0.0024	0.9000	0.9659	0.9596	0.9558	+0.0024
map	**3**	-	0.9234	0.9566	0.9542	0.9495	-0.0131	0.9013	0.9504	0.9578	0.9450	-0.0108
map	**0**	-	0.8168	0.9158	0.8182	0.8733	-0.0869	0.5398	0.9089	0.8226	0.8440	-0.1094
none	6	-	0.0000	0.9368	0.9319	0.7509	-0.2093	0.2692	0.9272	0.9279	0.8374	-0.1160
none	**0**	-	0.6148	0.5870	0.6897	0.6167	-0.3435	0.9984	0.0179	0.0353	0.1539	-0.7995

TABLE V

EVALUATING OUR APPROACH ON AN EXTENDED 5-CLASS PROBLEM USING MAP CONTEXT, $H = 6$S AND NO NOISE.

	Validation Set						Test Set					
Backbone	Standing	Driving	Load Handling	Turn Right	Turn Left	Overall	Standing	Driving	Load Handling	Turn Right	Turn Left	Overall
ResNet-50	0.9317	0.9156	0.9809	0.5501	0.8191	0.9160	0.9045	0.8732	0.9801	0.5890	0.7816	0.8833
ResNet-18	0.9521	0.8884	0.9750	0.5350	0.8475	0.9047	0.9267	0.8255	0.9719	0.5608	0.8146	0.8543

Since the two *turn* classes are rather underrepresented in our data, we use a weighted cross-entropy loss when training our models on the 5-class problem. We compute the loss weights based on the class distribution in the train set.

V. RESULTS AND DISCUSSION

A. Baseline

In our baseline experiments, we evaluate the frame-level classification accuracy on the three action classes most relevant for our intralogistics use-case: *standing*, *driving* and *load handling*. The load handling action consists of a vehicle positioning in front of a rack, movement towards it, fork and mast movement during loading/unloading the cargo (while the forklift stands still), and moving away from the rack.

First, we evaluate our models on data generated based on the ground-truth vehicle positions from our annotation data (assuming that we have a perfect 3D object detection model). For our baseline experiments we use a historic time frame of $H = 6$s (corresponding to 3 vehicle positions) and map context. The results in Table III show that our best model achieves an action detection accuracy of up to 0.9602 on our clean validation set. Our accuracy on the test set is slightly lower, but we achieve again very strong results of classification accuracy up to 0.9534. We can also see that using a larger backbone does not improve the performance on our 3-class datasets. Hence, we continue to use ResNet-18 for our experiments due to our focus on efficiency. Testing our approach on standard CPUs and GPUs we could see that inference using ResNet-18 is approximately two times faster than using ResNet-50.

We also analyze the performance of our ResNet-18 baseline model on different phases of an action: We split each action into an early phase (first 25% of frames showing a particular action sequence performed by a vehicle), main phase (subsequent 50% of frames showing a single action) and end (the last 25% of an action). Our baseline model achieves an accuracy of 0.893 on the early phase and an accuracy of around 0.97 on the middle and end phase. At the begin of an action, the model performs worse, since the temporal context still shows the previous action.

B. Detailed Analysis on Oracle Data

To provide a more detailed analysis of our approach, we evaluate models on different datasets, where we vary:

- Context: map context or no context information.
- Temporal sequence length H: 9, 6, 3 or 0 seconds (corresponding to 4, 3, 2 or 1 projected positions).
- Noise: optional uniform noise added to oracle data from annotations (± 0.25 meters to vehicle positions, $\pm 22.5^\circ$ to vehicle heading and ± 5 frames for temporal sampling). When using noise, we corrupt either all samples or a randomly sampled subset with noise.

Table IV shows that adding noise leads to slightly lower accuracy, but the models achieve still very good results. We reach an accuracy of 0.9497 on the test set if all samples are corrupted by noise and 0.9448 if 40% of the data is noisy and the rest is clean.

Increasing the temporal context from 6 to 9 seconds results in a minor improvement ($+0.0024$ test acc.), but slightly increases the computational costs to render the input crops. Lowering the temporal context to only 3 seconds leads

TABLE VI

RESULTS ON OUR TEST SET WHEN USING THE PREDICTIONS FROM OUR POINTPILLARS 3D OBJECT DETECTION MODEL. **BOLD** SETTINGS INDICATE CHANGES W.R.T TO OUR BASELINE CONFIGURATION.

Train Set Configuration			Test Set Configuration		Test Set				
Context	H (s)	Noise	Context	H (s)	Standing	Driving	Load Handling	Overall	Δ
map	6	**40%**	map	6	0.9256	0.9645	0.8339	0.9412	
map	6	**100%**	map	6	0.8624	0.9882	0.7845	0.9409	-0.0003
map	6	-	map	6	0.9156	0.9645	0.7940	0.9343	-0.0069
map	6	**40%**	**pts**	6	0.7129	0.9834	0.3496	0.8564	-0.0848
map	6	-	**pts**	6	0.3904	0.9878	0.3669	0.8079	-0.1333
map	6	**100%**	**pts**	6	0.3589	0.9949	0.3308	0.8030	-0.1382

to worse results due to the limited sequence information (-0.0108 test acc.). Thus, we deem $H = 6$s as a well suited trade-off regarding performance and accuracy.

To confirm our underlying assumption that spatial context and temporal sequence information are both major factors for inferring the current vehicle action, we next evaluate the influence of removing temporal or map context. On the validation and test data, removing the temporal information has less influence than removing the map context (Table IV). The main reason is that the pose of the vehicle with respect to the context (in an aisle or not, orientation w.r.t to racks) gives decent prior information about the current vehicle action. On the test data, keeping only the spatial context leads to a performance drop of -0.1094, while only keeping temporal context performs slightly worse (-0.116).

Spatial context is crucial to determine if a vehicle is currently parked or stands still in front of a rack (due to *load handling*). Without spatial context, the model can only predict the action class based on the training label distribution, *i.e. load handling* instead of *standing*, leading to inferior results (0.0 val acc., 0.2692 test acc. for *standing*). Naturally, when removing all context information, the performance breaks down (0.1539 accuracy on clean test data), as there are no features that allow a reliable prediction.

C. Expanding to 5 Action Classes

Additionally, we evaluate our models on a 5-class problem, where we split the *driving* class into *driving* (straight), *turn left* and *turn right* by post-processing our labels. These five classes fully categorize the typical behavior of forklifts in warehouses. Here, our models achieve an accuracy of up to 0.9160 on the validation data and 0.8833 on the test data (Table V). In this more complex problem setting, using a ResNet-50 backbone gives a performance improvement compared to using a ResNet-18 backbone. Due to the uneven class distribution, we observed that properly selecting the weights for the cross-entropy loss is crucial in this more fine-grained setting (higher accuracy on *turn* classes versus higher overall accuracy).

D. Test on Predictions from 3D Object Detection Model

The previous evaluations demonstrated the conceptual capabilities of our model by assuming a perfect (*oracle*) 3D detection system. In general, however, the detector is a potential limiting factor: If it fails to detect a vehicle in a keyframe, no corresponding action label can be estimated. To investigate this effect, we also evaluate our model using PointPillars [29] detections, a detector that can be deployed on the AMR's embedded hardware. To avoid introducing a bias due to detection failures for extremely sparse objects, we follow the common practice of 3D object detection benchmarks and remove objects that consist of less than 5 LiDAR points. This results in $14,261$ *standing*, $60,143$ *driving* and $11,034$ *load handling* ground truth annotations.

Table VI summarizes this PointPillars-based evaluation. First, we evaluate three models which are trained on clean data/partially noisy data, and noisy data. The models achieve a high accuracy of up to 0.9412, indicating that models which are trained on data corrupted with noise slightly outperform the model trained only on clean data. If we use the detection oracle instead, we could achieve an accuracy of 0.9640 on this reduced dataset, which indicates that detection failures do not severely affect our model.

Next, we experiment with using only raw point cloud data as context information to reduce the dependency on the context map. We project all points above a height threshold of $0.9\,\mathrm{m}$ onto the bird's-eye-view and use this sparse point representation to denote potential rack locations in the warehouse (see also Figure 2). This sparse context map leads to a slight performance drop, where our best model achieves an accuracy of 0.8564. The performance gap is larger for the *standing* and *load handling* classes. The main reason is that context information is essential to determine if a vehicle is parked or stands in front of a rack as part of a *load handling* action. This issue is most pronounced for vehicles far away from the AMR, due to the lower point cloud density, which leads to sparser context information.

VI. CONCLUSIONS

We present a new action-by-detection approach that infers vehicle actions in warehouses based on object poses readily available from a 3D detector. We utilize a highly efficient 2D CNN which learns spatial context and temporal path information from a suitable bird's-eye view abstraction. Our models achieve excellent classification results, both conceptually (assuming a perfect detection pipeline) and in a practical scenario by relying on the predictions of a PointPillars object detector. In detailed evaluations we demonstrate that our approach can also be applied when no schematic map is available (by alternatively obtaining context from raw point cloud data) and that our approach can easily be extended to more fine-grained action classification problems.

REFERENCES

[1] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond Short Snippets: Deep Networks for Video Classification," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4694–4702.

[2] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2625–2634.

[3] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional Two-Stream Network Fusion for Video Action Recognition," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1933–1941.

[4] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6299–6308.

[5] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast Networks for Video Recognition," in *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6202–6211.

[6] C. Feichtenhofer, "X3D: Expanding Architectures for Efficient Video Recognition," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 203–213.

[7] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[8] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, *et al.*, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2446–2454.

[9] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 621–11 631.

[10] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays, "Argoverse 2: Next Generation Datasets for Self-driving Perception and Forecasting," in *Proc. of the Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*, 2021.

[11] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal Trajectory Predictions for Autonomous Driving Using Deep Convolutional Networks," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2090–2096.

[12] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal Behavior Prediction Using Trajectory Sets," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 14 074–14 083.

[13] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "VectorNet: Encoding HD Maps and Agent Dynamics From Vectorized Representation," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 525–11 533.

[14] J. Gu, C. Sun, and H. Zhao, "Densetnt: End-to-end trajectory prediction from dense goal sets," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 15 303–15 312.

[15] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov, *et al.*, "MultiPath++: Efficient Information Fusion and Trajectory Aggregation for Behavior Prediction," in *Proc. of the International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7814–7821.

[16] S. Casas, W. Luo, and R. Urtasun, "IntentNet: Learning to Predict Intention from Raw Sensor Data," in *Proc. of the Conference on Robot Learning*, 2018, pp. 947–956.

[17] S. Malla, B. Dariush, and C. Choi, "TITAN: Future Forecast using Action Priors," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 186–11 196.

[18] G. Singh, S. Akrigg, M. Di Maio, V. Fontana, R. J. Alitappeh, S. Khan, S. Saha, K. Jeddisaravi, F. Yousefi, J. Culley, *et al.*, "ROAD: The Road Event Awareness Dataset for Autonomous Driving," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 45, no. 1, pp. 1036–1054, 2022.

[19] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The Oxford Robotcar Dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[20] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik, "AVA: A Video Dataset of Spatio-Temporally Localized Atomic Visual Actions," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[21] Z. Tong, Y. Song, J. Wang, and L. Wang, "VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training," *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, vol. 35, pp. 10 078–10 093, 2022.

[22] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, N. Singh, and J. Schneider, "Uncertainty-Aware Short-Term Motion Prediction of Traffic Actors for Autonomous Driving," in *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[23] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai, "Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," in *Proc. of the IEEE/CVF European Conference on Computer Vision (ECCV)*. Springer, 2022, pp. 1–18.

[24] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. L. Rus, and S. Han, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2774–2781.

[25] A. Motroni, A. Buffi, P. Nepa, M. Pesi, and A. Congi, "An Action Classification Method for Forklift Monitoring in Industry 4.0 Scenarios," *Sensors*, vol. 21, no. 15, p. 5183, 2021.

[26] K. Chen, T. Rögnvaldsson, S. Nowaczyk, S. Pashami, E. Johansson, and G. Sternelöv, "Semi-Supervised Learning for Forklift Activity Recognition from Controller Area Network (CAN) Signals," *Sensors*, vol. 22, no. 11, p. 4170, 2022.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[28] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing," in *Proc. of the IEEE American Control Conference*, 2007, pp. 2296–2301.

[29] 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 the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.