

CARLA Real Traffic Scenarios – novel training ground and benchmark for autonomous driving

Błażej Osiński^{2,7,*,\$}, Piotr Miłoś^{1,3}, Adam Jakubowski^{8,\$}, Paweł Zięcina^{9,\$}, Michał Martyniak¹, Christopher Galias⁴, Antonia Breuer⁵, Silviu Homoceanu⁵ and Henryk Michalewski^{2,6}

Abstract—This work introduces interactive traffic scenarios in the CARLA simulator, which are based on real-world traffic. We concentrate on tactical tasks lasting several seconds, which are especially challenging for current control methods. The CARLA Real Traffic Scenarios (CRTS) is intended to be a training and testing ground for autonomous driving systems. To this end, we open-source the code under a permissive license and present a set of baseline policies. CRTS combines the realism of traffic scenarios and the flexibility of simulation. We use it to train agents using a reinforcement learning algorithm. We show how to obtain competitive policies and evaluate experimentally how observation types and reward schemes affect the training process and the resulting agent’s behavior.

I. INTRODUCTION

The field of autonomous driving is a flourishing research field stimulated by the prospect of increasing safety and reducing demand for manual work. The ability to drive a car safely requires a diverse set of skills. First, it requires low-level control of a car – being able to keep the lane, make turns, and perform emergency stops. It also requires an understanding of the traffic rules (such as speed limits, right of way, traffic lights) and adhering to them. Finally, it requires high-level navigation – deciding which route to the destination is optimal.

Source dataset	Maneuver type	Our custom maps	No. of scenarios	
			train	validation
NGSIM openDD	highway lane change	2	1750	467
	drive through a roundabout	7	51752	12927

TABLE I: CARLA Real Traffic Scenarios (CRTS).

Computer-based systems can tackle these different skills with varying levels of success. High-level navigation is effectively solved by widely available GPS-based systems. There is also a growing body of work showing that the car control level can be efficiently solved using machine learning [1], [2]. Perhaps the hardest to automate is tactical planning tasks with a time horizon of several seconds, which require interactions with other traffic participants. A classic example is an unguarded 4-way intersection (investigated in e.g. [3]). In this work, we focus on *changing lanes on highways*

and *driving through roundabouts*. We present CARLA Real Traffic Scenarios (CRTS), including 9 new maps and a set of interactive traffic scenarios in the CARLA simulator [4], which may serve both for creating and evaluating autonomous driving systems. Importantly, these are extracted from datasets collected in the wild: NGSIM [5] and openDD [6] and thus pose real-world challenges. See Table I for basic statistics regarding datasets and new maps. Example videos are available at the project website: <https://sites.google.com/view/carla-real-traffic-scenarios/>.

To the best of our knowledge, this is the first publicly available work that adapts tactical-level real-world traffic data to a simulator with rich plausible physics and thus has a number of advantages. First and foremost, it allows for fast and relatively easy testing and comparison between various control methods and their implementation details. These may include planning versus policy approaches, learning algorithms, and methods with different levels of abstraction. Moreover, most recently published datasets (e.g. [7], [5], [8]) provide data from a fixed configuration of sensors. Our simulation-based solution removes this restriction and allows for easy data collection and testing of multiple sensor sets.

Using datasets listed in Table I as a source of trajectories offers an important benefit of facing situations that actually happened in the real world. Based on such trajectories, we build interactive scenarios. In a scenario, we replace a vehicle performing a maneuver with the ego vehicle simulated by CARLA and controlled by an algorithm in a closed-loop manner. The other vehicles follow the trajectory recorded in the dataset. The ego vehicle is tasked to execute the same maneuver. Performance is measured based on the final output (whether the ego-vehicle reached its target location) and negative events during driving (e.g. crashes, rapid turns, changes in speed). In particular, this implies that the policy steering the ego vehicle can use other than original strategies to perform maneuvers. We believe that it is important to understand these strategies if they are different from human driving styles and in what respects. Similarly to [9], we think that such an analysis, in the single-agent setting with realistic surroundings, might provide important clues for building multi-agent models. Moreover, one may be able to assess if the policy exploits the simulation’s particularities, which might be critical in the goal of transferring the policy to the real world.

Our aim is to provide training and testing ground for other researchers. The set of train scenarios is big enough to train or fine-tune models used for control. We standardize

*Correspondence to b.osinski@mimuw.edu.pl.

¹deepsense.ai, Warsaw, Poland, ²University of Warsaw, Warsaw, Poland

³Institute of Mathematics, Polish Academy of Sciences, Warsaw, Poland

⁴Jagiellonian University, Kraków, Poland

⁵Volkswagen AG, Wolfsburg, Germany

⁶Google, Warsaw, Poland, ⁷Lyft Level 5, London, UK

⁸Synerise, Warsaw, Poland, ⁹NVIDIA Corporation

\$ - work done while at deepsense.ai



Fig. 1: Scene from openDD dataset together with its in simulation equivalent.

the evaluation protocol by arranging train and validation split. We note that the validation scenarios allow us to test generalization to previously unseen circumstances. CRTS can be useful in various research applications, including learning methods and more classical planning. We note that drone-collected datasets can provide extensive stream of realistic road data as they contain information about multiple and interacting actors. Moreover, they required relatively small hardware investment. The price to pay is that they do not contain precise information about driver actions. For this reason, standard imitation learning, which is commonly used in literature [10], is not easily applicable here. In this work we use reinforcement learning (RL) to circumvent this restriction. Another advantage of RL is it can acquire novel solutions or detect corner cases. We show the versatility of CRTS, which stems from the fact that it runs in simulation. In particular, we compare how various observations modes (bird’s-eye view, visual input, and lidar and camera) affect the training and performance of the resulting policies. We also study generalization from the train set to the test set, which has recently become a popular research direction in RL; see e.g. [11], [12].

To summarize, we submit the following contributions:

- A set of driving scenarios based on real data in the CARLA simulator, including 9 new custom maps. The source code for the scenarios is available online.¹ See Section II
- Competitive driving policies utilizing the PPO algorithm [13] in various observation and rewards setups. See Section IV.
- Quantitative analysis of these policies with respect to task efficiency and generalization, which indicates that the bird’s-eye view observations and dense rewards outperform other tested methods. See Section IV.

II. CARLA REAL TRAFFIC SCENARIOS (CRTS)

We use recent version 0.9.9.4 of CARLA [4], an open-source simulator for autonomous driving research based on Unreal Engine 4. CARLA features open assets, built-in maps, weather settings, and multiple vehicles with different physical parameters and various sensors. Two visual quality levels (LOW and EPIC) are supported. We utilize two datasets, NGSIM and openDD, of 2D trajectories collected by drones to obtain a set 3D scenarios with plausible physics delivered by CARLA.

a) NGSIM: In these scenarios, the ego vehicle is tasked with performing a lane change maneuver on a highway. The source of the data is The Next Generation Simulation (NGSIM) dataset [5], [14], which contains vehicles trajectories collected on American highways. For parsing the data, we used code from [9]. We developed two custom CARLA maps of the locations covered by the dataset (southbound US 101 in Los Angeles, CA, and eastbound I-80 in Emeryville, CA).

b) openDD: The openDD dataset [6] is a large-scale trajectory dataset recorded from bird’s eye view. It focuses on roundabouts, which are an example of interaction-intensive traffic not regulated by traffic lights. openDD is the largest publicly available trajectory dataset recorded from a drone containing over 84k trajectories distilled from 62 hours of video data. The data was collected on 7 roundabouts – five located in Wolfsburg and two in Ingolstadt (both in Germany). We developed 7 custom CARLA maps corresponding to these roundabouts, see example in Figure 1.

c) Extraction algorithm: Based on the dataset of trajectories we built interactive scenarios. In a given scenario, we replace the vehicle performing a maneuver with the ego vehicle, whose physics is simulated by CARLA and controlled by an algorithm in a closed-loop manner. The other vehicles follow the trajectory recorded in the dataset. The ego vehicle is tasked with executing the same maneuver. Performance is measured based on final output (whether the ego-vehicle reached its target location) and negative events during driving (e.g. crashes or performing the task for too long).

From the NGSIM dataset, we extracted over 2k scenarios of lane change maneuvers and assigned 1750 for training and

¹<https://github.com/deepsense-ai/carla-real-traffic-scenarios>. We provide also a Python package producing bird-eye view observation mode, which could be useful in unrelated projects: <https://github.com/deepsense-ai/carla-birdeye-view>.

467 for validation. From the openDD dataset, we obtained over 64k scenarios of roundabout crossing and assigned 51752 for training and 12927 for validation.

Details of the scenario extraction procedure are provided in Appendix V-A.1. The scenarios can be accessed using the standard OpenAI Gym interface [15]. The ego vehicle model is Audi A2.

A. Observation space

One of the crucial advantages of CRTS is its flexibility due to the use of simulation. In particular, one can easily test various observation settings. In our experiments we provide three major options: *bird’s-eye view*, *visual input*, and *lidar and camera*; see examples in Figure 2.

The *bird’s-eye view* input covers approximately $47m \times 38m$ meters of physical space encoded in 186×150 pixels. Semantic information is contained in 5 channels corresponding respectively to road, lanes, centerlines, other vehicles, and the ego vehicle. See details in Appendix V-B.1.

In *visual* experiments we use 4 cameras (front, left, right, and rear), each having a field of vision of 90 degrees and a resolution of 384×160 . In *front camera* experiments we use only the front-facing camera.

In *lidar and camera* experiments, we use a single front camera and 360° lidar inputs. The simulated lidar has a range of 80 meters, 32 channels, horizontal field of view between -15° , and $+10^\circ$; these parameters resemble lidar settings available in autonomous vehicles (see e.g. [16]). In order to be able to use a convolutional neural network to process the lidar input, the raw data (pointcloud) is projected to a 2D view; this is a popular approach (see [17]).

The observation also includes a high-level navigational command (either `GO_STRAIGHT`, `TURN_RIGHT`, or `TURN_LEFT`), as inspired by [10]. These are used to indicate the lane change direction in NGSIM and the roundabout exit in openDD.

B. Action space

The ego vehicle is controlled by steering and speed. The steering wheel’s angle can be set continuously and is processed by the CARLA engine to steer the wheel (the angle of the wheels can vary from -80° to 80° , which is normalized to $[-1, 1]$ for the policy output). The speed is controlled indirectly by issuing commands to a PID controller, which controls the throttle and break. For both steering and speed, the policy is modeled using the Gaussian distribution with the mean and the logarithm of the variance output by a neural network.

C. Metrics

We suggest reporting the success rate of the performed maneuver as the primary metric and we recommend that the performance on a test scenario is reported using an average result from at least 50 episodes.

As our code is open, other custom metrics can be defined. One could, for example, imagine measuring the comfort of driving.

III. RELATED WORK

Autonomous driving is one of most important topics in contemporary AI research with a potential for huge impact. It is beyond the scope of this paper to discuss the whole field. We refer to [2], [3], [17] for general surveys.

Our work concerns several aspects: using realistic simulations, using real-world data, and training end-to-end policies using reinforcement learning.

Over the years, simulations have become a routinely-used tool for studying real-world control problems. Their clear advantages include making data collection cheaper and alleviating safety issues. From them to be useful, the critical research problem of transfer to the real environment has to be solved. Successful attempts, for example [18], usually address the problem in two ways: by making the simulation more realistic and by achieving highly adaptable policies. The latter is often obtained using domain randomization, a technique pioneered in [19] (and later extended in many works), which hinges on the assumption that a policy robust to a wide spectrum of simulated conditions would also be applicable to one special instance: reality.

Unsurprisingly, using simulation is an attractive research avenue for the autonomous driving community. The two popular packages CARLA [4] and AIRSIM [20] use Unreal engine to build efficient photo-realistic simulations with plausible physics. Similarly, TORCS [21] is a widely-used racing car simulator. There are also other simulators which are attractive for autonomous driving research; we refer to [22] for a comprehensive survey.

On the other hand autonomous driving companies amass ever-increasing datasets of driving data collected in the wild. There is also an increasing number of publicly available datasets. A non-exhaustive list includes NGSIM [5] and highD [7] datasets recorded on highways. inD [23] and openDD [6] concentrate on maneuvers on intersections and roundabouts. KITTI [8], A2D2 [16], and [24] datasets consists of footage recorded from cars using various sensors (RGB cameras, lidar) along with various preprocessing. SDD [25] and CITR [26] put emphasis on humans and human-car interactions.

Methods using deep neural networks (DNN) and learning have become quite fruitful for autonomous driving. There are three major paradigms of building DNN-based solutions (which are partially analogous to the discussion in cognitive sciences). The first one, *mediated perception*, is perhaps the most prominent today; see e.g. [27], [8]. It advocates using a vast number of sub-components recognizing all relevant objects (other cars, traffic lights, etc.) to form a comprehensive representation of the scene, which is then fed into a deep neural network. On the other side of the spectrum is the *behavior reflex* approach, which suggests learning a direct mapping from raw observations to control actions; see [28], [29] for two prominent examples. The third paradigm, *direct perception*, lays in the middle. It recommends learning affordances (e.g. the angle of the car relative to the road), which form a compact representation, which can then be used by another DNN to control the car; see e.g. [30]. Some other

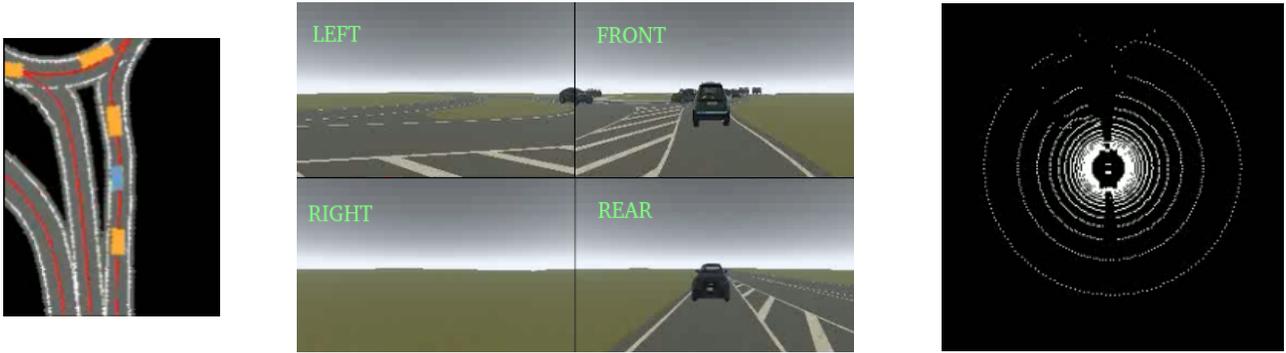


Fig. 2: Different observations types: bird’s-eye view, 4 cameras, and lidar’s projection on 2D view.

works include [31], which utilizes an augmented form of behavioral cloning with perturbed trajectories to successfully train an agent in simulation and shows its performance on a real car. [10] introduce high-level navigation commands and propose imitation learning conditioned on them to train an agent in simulation and test its performance on a toy car. [32] show that training an agent in a teacher-student framework (where the teacher has access to privileged information) leads to state-of-the-art performance on previous benchmarks. [33] presents a model-based approach to learning autonomous driving behaviors—a learned RNN model of dynamics is optimized directly with SGD. [34] raises explicit concerns about the safety of RL-based solutions. They observe that policy gradient techniques suffer from high variance when dealing with catastrophic events of low risk. To alleviate that, they propose decomposing the policy into a high and a low level one. The former is learnable, while the latter is hard-coded to ensure safety. [35] and [36] train a reinforcement learning agent in simulation and show successful transfer to a real-world vehicle. As the field grows fast, we only can only scratch the surface; see [37] for a recent overview of deep reinforcement learning methods used for autonomous driving.

Performing maneuvers is an important aspect of driving. [38] use a deep Q-network with continuous action and state spaces for automated lane change maneuvers. [39] propose a new state representation and utilize an asynchronous DQN architecture in a multi-agent setting. See [40] for a survey on more classical methods employed in lane change maneuvers.

Another initiative similar to ours is the CARLA autonomous driving leaderboard at <https://leaderboard.carla.org/>. It presents a suite of scenarios, infrastructure, and an evaluation protocol to standardize the evaluation of autonomous driving methods. This aim is in many aspects similar to ours. The key distinction is that our scenarios are modeled using real-life data. Moreover, CRTS is meant to be more of an open-ended research suite than a challenge; thus, it contains a suite of scenarios which is sufficiently large to enable meaningful training and does not impose strong constraints (e.g., sensor choice). All-in-all, these two initiatives (the CARLA autonomous driving leaderboard and CRTS) share many similarities and goals and we imagine they could be merged at some point.

IV. EXPERIMENTS

Our experiments’ main aim was to show how CRTS can be used in a versatile way to answer research questions concerning autonomous driving, sometimes arriving at counter-intuitive conclusions. The results are intended to be baselines for other researchers who would like to utilize CRTS. There are numerous further open questions, some of which we list in Section V.

The specific research questions are: a) can reinforcement learning be used to obtain maneuvering policies perform on test scenarios? b) how does the observation mode affect the training and the quality of resulting policies? c) how do the trained maneuvering strategies compare to realistic (human) ones?

In our experiments we used a slightly modified PPO algorithm [13]. PPO is one of the most popular model-free RL algorithms known for its flexibility and stability. Being an on-policy algorithm, it usually requires a substantial number of training samples. Importantly, using a simulator mitigates this issue.

a) *Rewards*: The choice of the reward function is an important aspect of RL. This can both affect the training performance and the quality of the final policy. Our driving scenarios allow for easy experimentation with various choices. In our benchmark PPO experiments, we used dense rewards for intermediate progress towards completing a maneuver. This, in particular, made our training rather stable and independent of the random initialization. In Section IV-C, we provide more details and present experiments with other rewards schemes. Interestingly, in some cases, a sparse rewards setting offers competitive training performance.

b) *Generalization*: Generalization is a critical and multi-faceted topic in machine learning. In our experiments, we test generalization between scenarios.² We note, however, that CRTS can be used to test generalization in other domains, such as perception (by varying weather setting and other visual details) or dynamics (by varying the car model and physics settings).

Table II presents the success rate on the test set. Our key conclusions are that the bird’s-eye view generalizes almost

²In this work we fix the low-quality CARLA settings, set the standard weather, and choose the ego vehicle to be Audi A2.

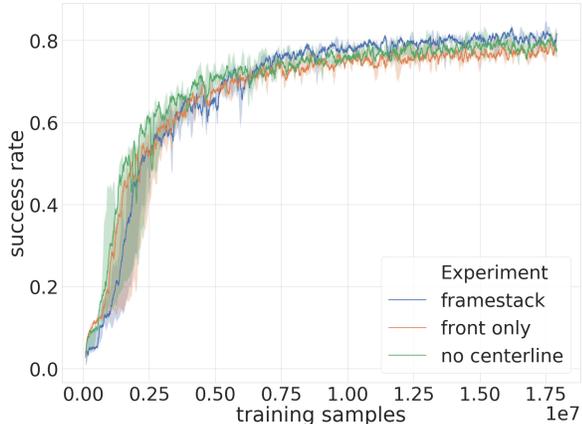


Fig. 3: Above: training curve for different variants of bird’s-eye view comparisons on NGSIM. In each case three experiments were run. Below: the same situation from openDD in three views. From left to right: base, front only, no centerline.

perfectly and is considerably better than the other observation modalities and that the dense rewards are important to obtain good generalization. Details are provided in the subsequent sections.

Our result is a datapoint in the paradigm discussion outlined in Section III, supporting mediated perception rather than the behavior reflex approach. The importance of these conclusions depends strongly on how well they transfer to the real world. Assessing such transfer is generally an open research question – two popular complementary methods are domain randomization (see e.g. [18]) and quantifying sim-to-real alignment (see e.g. [41]).

A. Bird’s-eye view experiments

Bird’s-eye view, see Section II-A and Figure 4, is a top-down view of the road situation. It contains privileged information, unavailable for normal road actors. It has been popularized in [32] in their two-phase training procedures. It also resembles information that could be extracted from a perception system and HD-maps in a typical AV stack; see e.g. [42].

An interesting question is if (and how much) bird’s-eye facilitates training; see Section IV-B. Here, we analyze the influence of various elements. Along with the “basic” experiment we ran the following modifications: *front only*, *no centerline*, and *frame stack*; see Appendix V-B.1 for technical details.

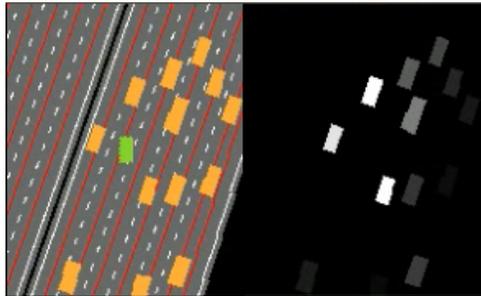


Fig. 4: Bird’s-eye view and saliency map

Experimentally, we found little difference during training, see Figure 3. These results are somewhat surprising for us. In the *front only* experiments, we remove the rear part of the observation. Contrary to our initial intuitions, it barely affects the training. In the *front only* experiments, we remove the rear part of the observation. Contrary to our initial intuitions, it barely affects the training. One important piece of information missing in the bird’s-eye view experiments is that of (relative) speed of other vehicles. A standard way to mitigate it is to use framestack (stacking a few consecutive observations, 4 in our case), which allows us to infer the speed. Again, contrary to what we expected, the training performance is weaker, probably due to optimization issues with the bigger input.

It is important to compare to evaluations on the test scenarios. Here the picture is much different. All ablations present worse performance compared to the standard bird’s-eye view setup. While underperforming of the *front only* experiments is expected, the seemingly redundant centerlines and much worse behavior of framestack policies might be surprising. The latter can be possibly explained by overfitting to bigger input and is subject to further studies.

a) *Custom/semantic saliency map*: Using neural networks for policies has the disadvantage of being relatively hard to interpret and debug. In order to alleviate this issue, we developed a new simple method of generating *semantic saliency maps*, which measure the contribution of other traffic participants towards decisions taken by the policy. This is achieved by erasing a participant (a car) from the input, passing the modified observation through the network, and comparing how much the output (steering and speed) has changed. In Figure 4 you can see an example of a situation faced by the ego vehicle (green), together with its semantic saliency map. As expected, it mostly pays attention to the nearest cars.

B. Other modalities

a) *Comparison of various observation modes*: As mentioned before, using a simulator makes it easy to test the performance of algorithms with different possible inputs. Apart from the bird’s-eye view input, we examined visual input (consisting of 4 RGB cameras), and a lidar-based setup (consisting of lidar and one front-facing camera). The observation spaces for both experiments are detailed in Section II-A.

	Base experiments			Bird’s-eye ablations		
	bird’s-eye	visual	lidar & camera	front only	no centerline	framestack
ngsim	0.787 ± 0.029	0.770 ± 0.022	0.776 ± 0.014	0.782 ± 0.035	0.782 ± 0.016	0.805 ± 0.022
opendd	0.862 ± 0.037	0.886 ± 0.023	0.805 ± 0.106	0.824 ± 0.058	0.833 ± 0.029	0.762 ± 0.083

TABLE II: Means and standard deviations of success rates of various models, obtained over three experiments. For each experiment success rate is calculated on the same scenarios from the validation set.

During training, all three modalities showed similar performance. However, as evident from Table II, there is a difference in performance on the validation set. The bird’s-eye view policy was best, visual input was slightly worse, and lidar plus camera was noticeably weaker. A possible explanation of this fact might be that data augmentation is required to achieve generalization in RL, as argued in recent work [43], [44].

We run one additional ablation experiment: in the *front camera* experiments we use only one camera. Interestingly, in most cases, that is enough to successfully perform a lane change maneuver, because cars on the highway usually maintain a certain distance and the learned policy will try to exploit this by changing the lane as fast as possible when there are no other cars in front.

C. Reward experiments

	Rewards experiments		
	dense	sparse	no failure penalty
ngsim	0.828	0.741	0.707
opendd	0.914	0.829	0.757

TABLE III: Success rates on test scenarios for CRTS scenarios with bird’s-eye view input.

In all the above experiments, we used the *dense reward* scheme. In this section we show comparison to two other schemes: *sparse* and *no failure penalty*. In the sparse scheme, the agent gets +1 for the successful completion of a scenario maneuver and -1 when it fails. In the dense scheme, additional rewards are given for partial completion of a maneuver. In our case, we divide every scenario into 10 parts, each part worth 0.1. The *no failure* scheme is the same as dense except that the -1 failure penalty is not issued. The details of the rewards are described in Appendix V-A.2.

The sparse rewards are perhaps the most natural, as they are close to the success rate metrics. We add the -1 penalty to make the learning of failure cases faster. Making rewards denser is often used in practice (sometimes referred to as “reward shaping”). The rationale is to make learning easier by decreasing the effective planning horizon. The major drawback is that such “shaping” is only a proxy of the desired objective and may lead training into an unintended direction (often unexpected to the experimenter).

In Figure 5 we report *success rate* on the training set. We can see that the dense rewards perform best; the sparse starts training considerably later, but after that, quickly catches up reaching slightly lower level. Perhaps surprisingly, removing

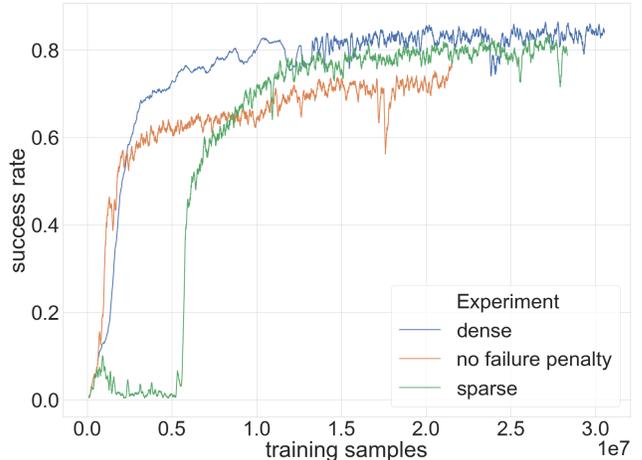


Fig. 5: Reward schemes comparison for NGSIM scenarios with bird’s-eye view input.

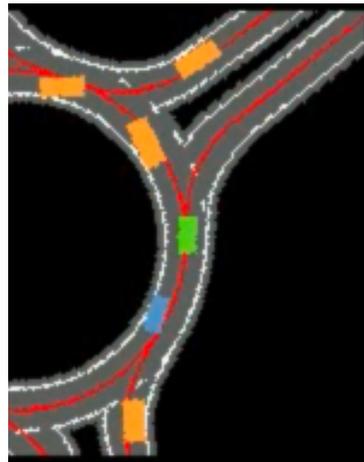


Fig. 6: Ego-vehicle position (green) and original car (blue).

the failure penalty significantly hinders the later phases of training. The results on the test set, see Table III, “stretch” the above differences, making the dense reward setting a clear winner.

D. Qualitative analysis of driving styles

There are many interesting questions concerning obtained driving policies. Perhaps the most compelling one is asking if they are “natural”. In an attempt to answer it we developed a tool for comparing the ego vehicle behavior with the original

drive from the data-set; see Figure 6.³ We observed that, in general, our policies drive much faster than the original ones. This aggressive driving style might be due to a lack of rewards for comfort and traffic rule following (we also observed a couple of situations driving on the left lane). The quality of driving looks correct, although a few cases of avoidable crashes were observed. In the front-view experiments, there are rare instances of crashes with an unobserved rear vehicle.

V. CONCLUSIONS AND FURTHER WORK

Our work introduces CRTS – a new training and evaluation ground for autonomous driving based on real-world data. Our benchmark follows good practices of providing a train/test split. We hope that CRTS will serve as a platform for developing new methods and a sanity check before deployment in the real world.

We provide benchmark evaluations using reinforcement learning; we observe that there is still much room for improvement. We compare various observation settings and assess generalization and realism of obtained behaviors.

There are many research directions to pursue further. Perhaps the most apparent is adding more scenarios in number and type (e.g. [7]). In this work we focused on the maneuver success rate. This can be extended to serve more specific purposes, e.g. for measuring comfort. Generalization in other domains, like changing weather settings and vehicle models can also be tested. We saw that a sparse reward setting yields considerably worse results. It could be an interesting reinforcement learning problem to achieve better performance, perhaps using an algorithm like SAC that would provide better exploration and sample efficiency. Another direction is experiments with neural architecture – perhaps using LSTMs would bring better results. Our scenarios could be used with other learning methods (e.g. imitation learning or steering via waypoints) or more classical planning methods.

Another research avenue is casting the problem into a multi-agent setup by making other driving actors controllable. An important new challenge, in this case, will be maintaining behaviors resembling real-world ones.

APPENDIX

A. Benchmark scenarios

1) *Scenario extraction algorithm*: In this section we describe the procedure of creating scenarios. In both the cases of NGSIM [14] and openDD [6], the datasets are scanned for maneuver events – lane change and entering roundabouts, respectively. The car performing the maneuver is declared to be the ego vehicle and a scenario is formed by mapping all traffic participants except for the ego vehicle to CARLA. A vehicle appearing in the field of view for the first time is spawned within the location and velocity matching the dataset. The model of vehicle is chosen from the CARLA library to match the dimension of the replayed car (measured using the Jaccard similarity). Later, consistency with the dataset

³We also strongly recommend visiting our website <https://sites.google.com/view/carla-real-traffic-scenarios/> to see the corresponding videos.

is enforced every 100ms (by setting again the locations and velocities).

The initial speed and position of the ego vehicle is determined in the same way and in subsequent frames it is maintained by the CARLA physics engine according to the received actions.

The set of such created scenarios is divided randomly into the train and test set in the 80/20 ratio.

a) *NGSIM*: The lane change event is declared as changing the lane id corresponding to a given vehicle. The scenario starts 5 sec before the lane change event. The scenario is considered successful if, at least for 1 sec, the ego vehicle is located less than 30cm of the target lane and its yaw is smaller than 10 degrees. The scenario is unsuccessful if either a collision occurs, the car leaves both the starting and target lane or there is a timeout of 10 sec.

During the scenario the chauffeur commands are issued: `LANE_CHANGE_` (*dir*) if the ego vehicle is on the starting lane, *dir* $\in \{left, right\}$ denoted the direction to the target lane, otherwise `LANE_FOLLOW` is issued.

For the NGSIM we remap position of the rear and front axle of vehicle (to match the CARLA convention). We also smooth positions using a 1.5 sec window.

b) *openDD*: The scenarios begin when the ego vehicle is approx. 20 meters before the roundabout entry. The scenario is considered successful if the ego vehicle exits the roundabout via the same exit as the reference driver. The scenario is unsuccessful if either a collision occurs, the car moves away more than 3m from the original trajectory or there is a timeout of 1.5 times of the time of the original drive.

During the whole scenario `LANE_FOLLOW` command is issued until the car passes the last exit before the target one. Then the command changes to `TURN_RIGHT`.

The openDD dataset is recored at 30 fps, which we downsample to match our 10 fps.

2) Rewards:

a) *NGSIM*: The distance to the target lane is segmented into 10 pieces. A reward of 0.1 is issued each time the car moves a segment closer to the target lane and is penalized with -0.1 if it moves to a more distant segment.

At the end of the scenario a reward of 1 (resp. -1) is issued if the scenario is successful (resp. unsuccessful).

b) *openDD*: The original trajectory is segmented into 10 pieces. The agent is rewarded with 0.1 for passing each segment. At the end of the scenario a reward of 1 (resp. -1) is issued if the scenario is successful (resp. unsuccessful).

c) *Sparse reward*: In the above setting ± 0.1 “dense” rewards are issued for partial progress. In the sparse regime these are omitted.

d) *No failure penalty*: We do not issue -1 for unsuccessful scenarios.

B. Training details

1) *Observations for bird’s-eye view*: The observation in all *bird’s-eye view* baseline experiments is a tensor of shape (186,150,5), which corresponds to a real-world area of approximately $47m \times 38m$. The ego vehicle is located in the

center; see Figure 4. There are five channels, each representing a distinct category of CARLA world features: roads (grey area on Figure 4), lanes (solid and dashed white lines), centerlines (red lines), other vehicles (orange), and the ego vehicle (green). In *front only* experiments we use (186, 150, 5) tensors covering only area in front of the ego vehicle. In *no centerline* experiments the input is (186, 150, 4) as we omit the centerline channel. In *framestack* experiments we stack 4 consecutive observations into (186, 150, 20) tensors.

ACKNOWLEDGMENT

We greatly acknowledge support of MathWorks which provided us with a free licence of RoadRunner. We used RoadRunner to prepare new CARLA maps.

The work of Piotr Miłoś was supported by the Polish National Science Center grants UMO-2017/26/E/ST6/00622. This research was supported by the PL-Grid Infrastructure. We extensively used the Prometheus supercomputer, located in the Academic Computer Center Cyfronet in the AGH University of Science and Technology in Kraków, Poland. Our experiments were managed using <https://neptune.ai>. We would like to thank the Neptune team for providing us access to the team version and technical support.

REFERENCES

- [1] A. Tampuu, M. Semikin, N. Muhammad, D. Fishman, and T. Maitinen, "A survey of end-to-end driving: Architectures and training methods," *arXiv:2003.06404*, 2020.
- [2] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *arXiv:2002.00444*, 2020.
- [3] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Proceedings of Robotics: Science and Systems*, 2016.
- [4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *CoRL 2017*. PMLR, 2017.
- [5] J. Halkias and J. Colyar, "NGSIM interstate 80 freeway dataset," US Federal Highway Administration, Washington, DC, USA, 2006.
- [6] A. Breuer, J.-A. Termöhlen, S. Homoceanu, and T. Fingscheidt, "openDD: A large-scale roundabout drone dataset," *arXiv:2007.08463*, 2020.
- [7] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *ITSC 2018*.
- [8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *International Journal of Robotics Research (IJRR)*, 2013.
- [9] M. Henaff, A. Canziani, and Y. LeCun, "Model-predictive policy learning with uncertainty regularization for driving in dense traffic," in *ICLR*, 2019.
- [10] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *ICRA 2018*. IEEE, 2018.
- [11] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *International Conference on Machine Learning*, 2019.
- [12] A. Zhang, N. Ballas, and J. Pineau, "A dissection of overfitting and generalization in continuous reinforcement learning," *arXiv:1806.07937*, 2018.
- [13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.
- [14] U.S. Department of Transportation Federal Highway Administration, "Next generation simulation (NGSIM) vehicle trajectories and supporting data," 2016.
- [15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv:1606.01540*, 2016.
- [16] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, and P. Schuberth, "A2D2: Audi Autonomous Driving Dataset," 2020. [Online]. Available: <https://www.a2d2.audi>
- [17] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, 2020.
- [18] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell *et al.*, "Solving rubik's cube with a robot hand," *arXiv:1910.07113*, 2019.
- [19] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," *arXiv:1611.04201*, 2016.
- [20] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*. Springer, 2018, pp. 621–635.
- [21] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "Torcs, the open racing car simulator," 2000. [Online]. Available: <http://torcs.sourceforge.net>
- [22] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors*, vol. 19, no. 3, p. 648, 2019.
- [23] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The inD dataset: A drone dataset of naturalistic road user trajectories at german intersections," *arXiv:1911.07602*, 2019.
- [24] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Igloukov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," 2020.
- [25] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *ECCV 2016*.
- [26] D. Yang, L. Li, K. A. Redmill, and Ü. Özgüner, "Top-view trajectories: A pedestrian dataset of vehicle-crowd interaction from controlled experiments and crowded campus," in *IV 2019*. IEEE, 2019.
- [27] S. Ullman, "Against direct perception," *Behavioral and Brain Sciences*, vol. 3, no. 3, p. 373–381, 1980.
- [28] D. Pomerleau, "ALVINN: an autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems 1*. Morgan Kaufmann, 1988, pp. 305–313.
- [29] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *arXiv:1604.07316*, 2016.
- [30] C. Chen, A. Seff, A. L. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *ICCV 2015*. IEEE Computer Society, 2015.
- [31] M. Bansal, A. Krizhevsky, and A. Ogale, "ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst," *arXiv:1812.03079*, 2018.
- [32] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *CoRL 2019*. PMLR, 2019.
- [33] S. Shalev-Shwartz, N. Ben-Zrihem, A. Cohen, and A. Shashua, "Long-term planning by short-term prediction," *CoRR*, vol. abs/1602.01580, 2016. [Online]. Available: <http://arxiv.org/abs/1602.01580>
- [34] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *CoRR*, vol. abs/1610.03295, 2016. [Online]. Available: <http://arxiv.org/abs/1610.03295>
- [35] A. Bewley, J. Rigley, Y. Liu, J. Hawke, R. Shen, V. Lam, and A. Kendall, "Learning to drive from simulation without real world labels," *arXiv:1812.03823*.
- [36] B. Osiński, A. Jakubowski, P. Miłoś, P. Zięcina, C. Galias, and H. Michalewski, "Simulation-based reinforcement learning for real-world autonomous driving," *arXiv:1911.12905*, 2019.
- [37] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *arXiv:2002.00444*, 2020.
- [38] P. Wang, C. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *IV 2018*. IEEE, 2018.
- [39] C. Chen, J. Qian, H. Yao, J. Luo, H. Zhang, and W. Liu, "Towards comprehensive maneuver decisions for lane change using reinforcement learning," 2018.
- [40] D. Bevil, X. Cao, M. Gordon, G. Ozbilgin, D. Kari, B. Nelson, J. Woodruff, M. Barth, C. Murray, A. Kurt, K. Redmill, and Ümit

- Özgüner, "Lane change and merge maneuvers for connected and automated vehicles: A survey," *IEEE Transactions on Intelligent Vehicles*, 2016.
- [41] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," in *Proc. Rob: Sci and Sys*, 2018.
- [42] A. Breuer, S. Elflein, T. Joseph, J. Bolte, S. Homoceanu, and T. Fingscheidt, "Analysis of the effect of various input representations for LSTM-based trajectory prediction," in *ITSC 2019*.
- [43] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *arXiv:2004.14990*, 2020.
- [44] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," *arXiv:2004.13649*, 2020.