Simulation-based reinforcement learning for real-world autonomous driving

Błażej Osiński[1,4], Adam Jakubowski[1], Paweł Zięcina[1], Piotr Miłoś[1,5],
Christopher Galias[1,3], Silviu Homoceanu[2] and Henryk Michalewski[1,4]

***Abstract*— We use synthetic data and a reinforcement learning algorithm to train a driving system controlling a full-size real-world vehicle in a number of restricted driving scenarios. The driving policy uses RGB images as input.**

We show how design decisions about perception, control and training impact the real-world performance.

I. Introduction

This work focuses on verification whether using synthetic data from a simulator it is possible to obtain a driving system which can be deployed in a real car. Our policies were trained end-to-end using a reinforcement learning (RL) algorithm and confirmed to be useful when tested on restricted scenarios with a full-sized passenger vehicle with state-of-the-art equipment required for Level 4 autonomy.

A number of design decisions were made, taking into consideration the restriction of business environments. In particular we use mostly synthetic data, with labelled real-world data appearing only in the training of the segmentation network. We also decided to use only the RGB input provided by a single camera.

The driving policy is evaluated only concerning its real-world performance on multiple scenarios outlined in Section III. To complete a scenario, the driving agent needs to execute from approximately 250 to 700 actions at 10 Hz at speed varying from 15 to 30 km/h (4 to 8 m/s). In some of our experiments, the learned controller outputs the steering command directly. In other, the controller outputs waypoint which is processed to steering using a proprietary control system. In this work, we decided to limit intermediate human-designed or learned representations of the real world only to semantic segmentation. The semantic segmentator used in our system is the only component trained using the real-world data – its training process mixes real-world and synthetic images. Our driving policies are trained only in simulation and directly on visual inputs, understood as RGB images along with their segmentation masks, except experiment R6 where we use only segmented images. The input contains also selected car metrics and a high-level command inspired by [1].

Using reinforcement learning and RGB inputs was a conscious decision. The goal behind this choice was to answer the following research question: Is the system backed by the state-of-the-art RL methods able to learn driving in the end-to-end fashion?

In particular, is it able to acquire an intermediate representation of a scene, which is transferable from simulation to the real world? We note that being able to train end-to-end is desirable as it reduces human engineering effort. Even more importantly, it also eliminates errors arising when gluing a heterogeneous system consisting in particular of separate perception and control modules.

A major difficulty related to learning in simulation was stated in Section 5 of [2]: "when using a realistic simulator, an extensive hyperparameters search becomes infeasible". We are using the same realistic simulator as in [2] — CARLA, based on Unreal Engine 4. In order to alleviate the difficulty related to the time consuming training we implemented a parallelized training architecture inspired by IMPALA [3], Ape-X [4], OpenAI Five [5], Horovod [6], DBA3C [7], see also recent work [8]. Details of our parallelisation approach are presented at the project webpage `http://bit.ly/2maqbJj`. With our current infrastructure and parallelization methods, we generated as much as 100 years of simulated driving experience – in our view enough for the limited scenarios considered in this work. To verify whether synthetic data from a simulator helps in improving driving skills we conducted the following experiments which constitute the main contribution of this work:

1. In simulation: we verify the influence of visual randomizations on transfer between different scenarios in simulation; results are summarized in section IV-A.

2. In real-world scenarios: we deploy 9 models listed in Table I in 9 scenarios. In total we report results gathered over more than 400 test drives. See Section IV-B for a detailed description.

model	description
R1	standard randomizations
R1-reg	as R1 but lower entropy and L2-regularization
R2	standard randomizations and waypoint
R3	as R1 but also dynamics randomizations and policy with memory
R4	standard randomizations and continuous actions
R5	as R1, but with auxiliary task: depth prediction
R6	as R1, but just segmentation input without RGB
R1-baseline	as R1 but low randomizations
R3-baseline	as R3 but feed-forward network

TABLE I

In Section IV-C we describe two failure cases and in Section IV-D we assess a proxy metric potentially useful

*This work was not supported by any organization

[1] deepsense.ai
[2] Volkswagen AG, Wolfsburg, Germany
[3] Jagiellonian University
[4] University of Warsaw
[5] Institute of Mathematics of the Polish Academy of Sciences

for offline evaluation of models. To facilitate the review process, we provide recordings from 9 autonomous test drives `https://bit.ly/2k8syvh`. The videos should not be distributed. The test drives are in correspondence with the scenarios listed in Figures 2.

II. Related work

a) Synthetic data and real-world robotics: Synthetic images were used in the ALVINN experiment [9]. In [10] was proposed a training procedure for drones and in [11], [12], [13], [14], [5] were proposed experiments with robotic manipulators where training was performed using only synthetic data. Progressive nets and data generated using the MuJoCo engine [15] were used in [16] to learn policies in the domain of robot manipulation. A driving policy for a one-person vehicle was trained in [17]. The policy in [17] is reported to show good performance on a rural road and the training used mostly synthetic data generated by Unreal Engine 4. Our inclusion of segmentation as described in Section III-.0.f is inspired by sim2real experiments presented in [18]. Visual steering systems inspired by [10] and trained using synthetic data were presented in [19], [20].

b) Synthetic data and simulated robotics: Emergence of high-quality general purpose physics engines such as MuJoCo [15], along with game engines such as Unreal Engine 4 and Unity, and their specialized extensions such as CARLA [2] or AirSim [21], allowed for creation of sophisticated photo-realistic environments which can be used for training and testing of autonomous vehicles. A deep RL framework for autonomous driving was proposed in [22] and tested using the racing car simulator TORCS.

Reinforcement learning methods led to very good performance in simulated robotics, see for example solutions to complicated walking tasks in [23], [24]. In the context of CARLA, impressive driving policies were trained using imitation learning [1], [25], affordance learning [26], reinforcement learning [27], and a combination of model-based and imitation learning methods proposed in [28]. However, as stated in [17]: "training and evaluating methods purely in simulation is often 'doomed to succeed' at the desired task in a simulated environment" and indeed, in our suite of experiments described in Section III most of the simulated tasks can be relatively easily solved, in particular when a given environment is deterministic and simulated observations are not perturbed.

c) Reinforcement learning and real-world robotics: An extensive survey of various applications of RL in robotics can be found in [29, Section 2.5]. The role of simulators and RL in robotics is discussed in [30] in Section IV. In [10], [12], [13], [14], [5], [18], [17], [16] policies are deployed on real-world robots and training is performed mostly using data generated by simulators. [31] proposes a system with dynamics trained using real-world data and perception trained using synthetic data. Training of an RL policy in the TORCS engine with a real-world deployment is presented in [32].

III. Environment and learning algorithm

a) Simulator: We use CARLA ver. 0.9.5 [2], an open-source simulator for autonomous driving research, based on Unreal Engine 4. CARLA features open assets, including seven built-in maps, 14 predefined weather settings, semantic segmentation, as well as camera and LIDAR sensors (in our experiments we only use RGB information). Camera position, orientation, and settings are customizable. CARLA also features multiple vehicles with different physical parameters. Two visual quality levels (`LOW` and `EPIC`) are supported; the latter implements visual features including shadows, water reflections, sun flare effect, and antialiasing.

b) Simulated and real-world scenarios: The real-world deployments consist of 9 scenarios, see Figure 2. The scenarios include turns and an overpass. In training, we assume that the simulated environment is static, without any moving cars or pedestrians, hence a number of a human driver interventions during test deployments in real traffic is unavoidable. We developed new CARLA-compatible maps which cover approximately 50% of the testing grounds used in real-world deployments. We use these maps along with maps provided in CARLA for training, with some scenarios reserved for validation only.

In all scenarios agent's goal is to follow a predefined route from start to finish. A route is a list of checkpoints on the map. The number of timesteps for a given scenario ranges from 250 to 700. Agents are expected to drive in their own lanes, but other traffic rules are ignored.

c) Rewards in simulation and metrics of the real-world performance: In simulation, the agent is rewarded for following the reference trajectory (within some margin). The episode fails if the agent diverges more than 5 meters or collides with an obstacle. In the real world, for each scenario, we measure the percentage of distance driven autonomously (i.e. without human intervention); results are presented in Figures 1 and 6. Since tests were made in an uncontrolled environment with other vehicles and pedestrians, the human driver was instructed to take over in all situations which were potentially risky. We measure also divergence from expert trajectories (Figure 4).

d) Actions: Vehicles are controlled by two values: throttle and steering. The throttle is controlled by a PID controller with speed set to a constant, and thus our neural network policies only command the steering. We explore various possibilities for actions spaces. Typically, the policy is modeled as a probability distribution over the angle of the steering wheel. Unless stated otherwise, we used discretized angles. Their values are not distributed evenly - with more of them around 0 to improve smoothness of driving without increasing the action space too much (viz. [0., $\pm$0.01, $\pm$0.02, $\pm$0.03, $\pm$0.05, $\pm$0.08, $\pm$0.12, $\pm$0.15, $\pm$0.2, $\pm$0.25, $\pm$0.3, $\pm$0.4], values are in radians). In experiment R4 we use continuous values for the angle modeled with Gaussian distributions, and in R5 the policy outputs waypoints.

e) Observations: The observation of the agent consists of an RGB image from a single front camera which is downscaled to the resolution 134×84 pixels. The RGB

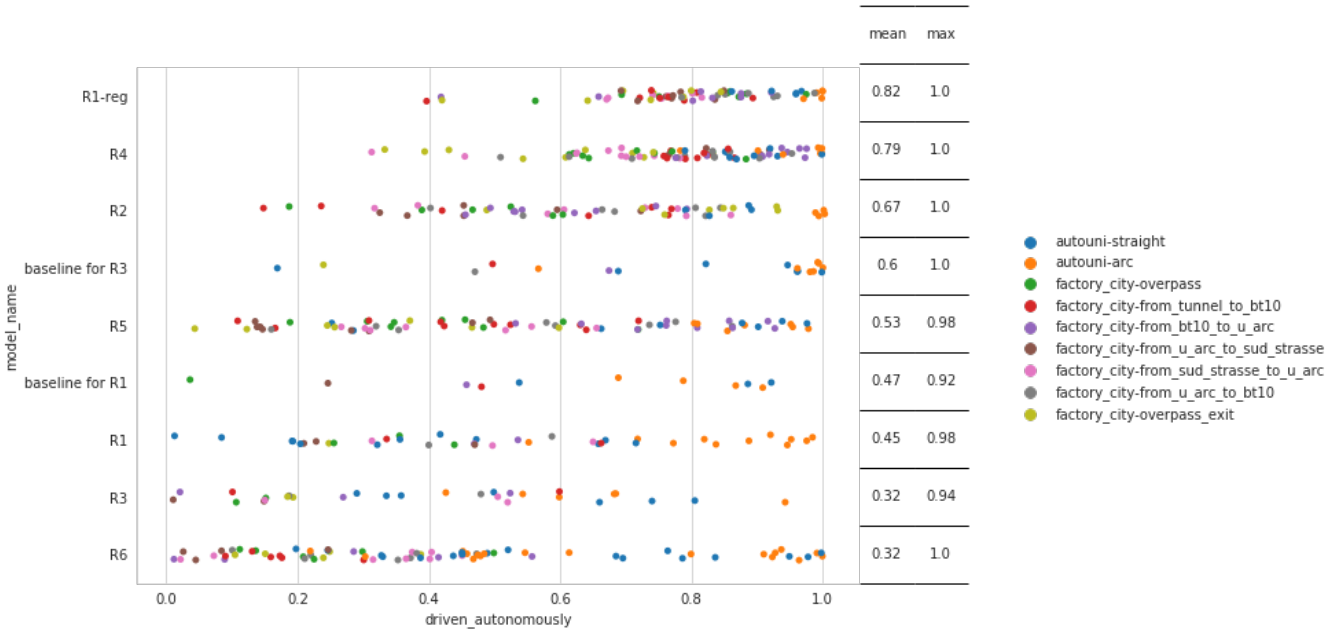

Fig. 1: Summary of experiments R1-R6 with baselines across nine scenarios. The columns to the right show the mean and max of autonomy (the percentage of distance driven autonomously). Models are sorted according to their mean performance. Print in color for better readability.

Fig. 2: All real-world scenarios used in our experiments. Left map: (a) `autouni-arc`, (b) `autouni-straight`. Center map: (c) `factory_city-overpass*`, (d) `factory_city-overpass_exit`. Right map: (e) `factory_city-tunnel-bt10*`, (f) `factory_city-bt10-u_turn`, (g) `factory_city-u_turn-sud_strasse`, (h) `factory_city-sud_strasse_u_turn*`, (i) `factory_city-u_turn-bt10*`. Scenarios marked with asterisk were used for training in simulation.

observation is concatenated with its semantic segmentation and two car metrics: speed and acceleration. There are two exceptions: in experiment R2 we also provide steering and in experiment R6 we do not include the RGB image. The camera position and orientation in simulation was configured to reflect the real-world setup. The agent is also provided with a high-level navigation command: lane follow, turn right/left or go straight.

f) Semantic segmentation: The semantic segmentation model is trained in a supervised way separately from the reinforcement learning loop. We used the U-Net [33] architecture and synthetic data from CARLA, the Mapillary dataset [34] as well as real-world labeled data from an environment similar to the one used in test drives.

g) Network architecture: RL policy is implemented using a neural network, see its simplified architecture in Fig. 3. As the feature extractor for the visual input (RGB and semantic segmentation) we use the network from [3]. Our choice was influenced by [35], where this network was shown to generalize well between different RL environments. Note that policy transfer between simulation and reality can be seen as a generalization challenge.

h) Learning algorithm: We used OpenAI Baselines [36] `ppo2` (see the project webpage `http://bit.ly/2maqbJj` for training hyperparameters). Thanks to dense rewards the training in simulation was quite stable across models and hyperparameters. For deployments we have decided to use 1-4 models per experiment type, using roughly 100M synthetic frames per training (equivalent to about 115

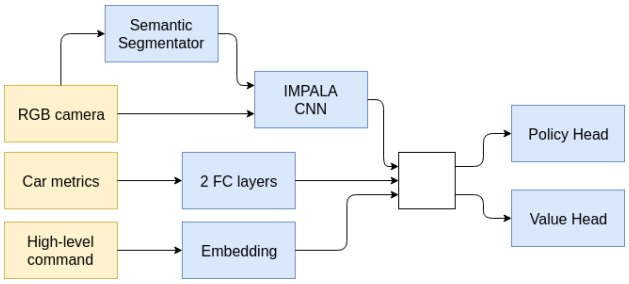

Fig. 3: Network architecture.

days of simulated time).

i) System identification: Inspired by the importance of system identification for sim-to-real transfer demonstrated in [37], we configured the CARLA simulator to mimic some values measured in the car used for deployment. These were the maximal steering angle and the time for a steering command to take effect.

IV. Experiments

Figure 1 summarizes the performance of our models in terms of the percentage of distance *driven autonomously* in all scenarios. See Figure 6 for a more fine-grained presentation.

Our best models are R1-Reg (regularized version of randomizations) and R4 (continuous actions) followed by R2 (control via waypoint instead of direct steering).

A. Experiment in simulation

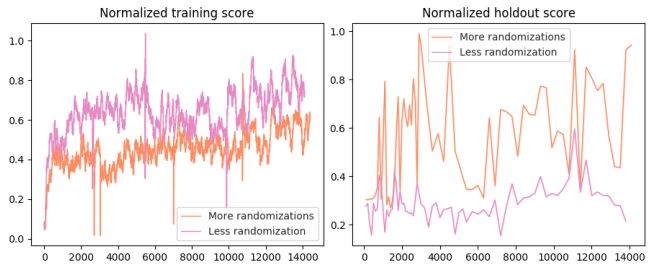

Fig. 5: Left: Episode scores obtained during training. Less randomization variant is easier and faster to train. Right: On a holdout town with holdout weather better results hold for a model trained with more randomization.

Experiment S1: In this experiment, we measure in simulation how randomizations affect performance. To this end, we apply fewer randomizations then the set used throughout all other experiments. We conclude that the model trained with this standard set generalizes better to drive in a holdout town and with a holdout weather setting, see Figure 5.

feature	Few randomizations	Standard randomizations
Weathers	One weather	10 weather settings
Unreal engine quality	[LOW]	[LOW, EPIC]
Visual randomizations	<NONE>	[Noise, Cutout, Brightness, Blur]
Towns	6 different towns	6 different towns

TABLE II

B. Experiments in the real world

See Figures 1 and 6 for an overview of performance of our models on nine real-world scenarios. Below we describe in more details our all real-world experiments.

Experiment R1 - base: This experiment provides baselines for comparisons with experiments R2-R6. To this end, we prepared three models. First two: R1 and baseline for R1 were trained respectively with standard and few randomizations settings, see Table 7 (analogously to S1). They were evaluated in the real world and performed disappointingly, mostly due to severe wobbling.

To address this issue we fine-tuned R1 by further training: reducing policy's entropy and including L2-regularization. The resulting model - R1-reg - was behaving significantly better. Improved performance in RL generalization when using L2-regularization was previously reported in [35].

Experiment R2 - waypoints: Following the approach presented in [38], we train model R2 to predict the next waypoint using branched neural network architecture with separate heads for each of the high-level command (such as turn left or lane follow).

Given a waypoint, low-level steering of the driving wheel is executed in order to reach this point. In simulation, it is realized by a PID controller while in the case of the real car, we use a proprietary control system. To ensure similar performance in simulation and reality, we limit the action space of the RL agent to waypoints reachable by both of the controllers. It consists of points within a radius of 5 meters of the car. The action space is discrete - potential waypoints are located every 5 degrees between -30 and 30, where 0 is the current orientation of the vehicle.

Experiment R3 – dynamics randomizations: In [39], [5] dynamics randomization is pointed as an important ingredient of the sim-to-real transfer. In our training of model R3 dynamic randomization parameters are sampled for each episode. Intuitively, an agent is expected to infer the dynamics parameters at the beginning of the episode and utilize it until its end for smooth driving. To this end, it needs to be endowed with memory, which in our case is a GRU memory cell [40].

We introduced randomization to the following aspects of the environment: target speed, steering response (including random multiplicative factor and bias), latency (the delay between observation and applying policy's response to it), and noise in car metrics observation (speed, acceleration, wheel angle).

R3 model exhibits inferior performance, somewhat surprisingly, even when compared with a baseline, for which we trained a feed-forward network (i.e., without memory) under dynamics randomization. We speculate that this is due to overfitting when using high-capacity models with memory. We intend to investigate this in future work.

Experiment R4 – continuous actions: In model R4, we use a continuous action space. In training, steering is sampled from a Gaussian distribution. Its mean is outputted by a neural network based on observations while its standard deviation is learnable and shared across observations. In evaluation, we use a deterministic policy by taking the mean of the distribution.

Experiment R5 – auxiliary depth: Auxiliary tasks are an established method of improving RL training, see e.g. [41]. Following that, in experiment R5, apart for the policy,

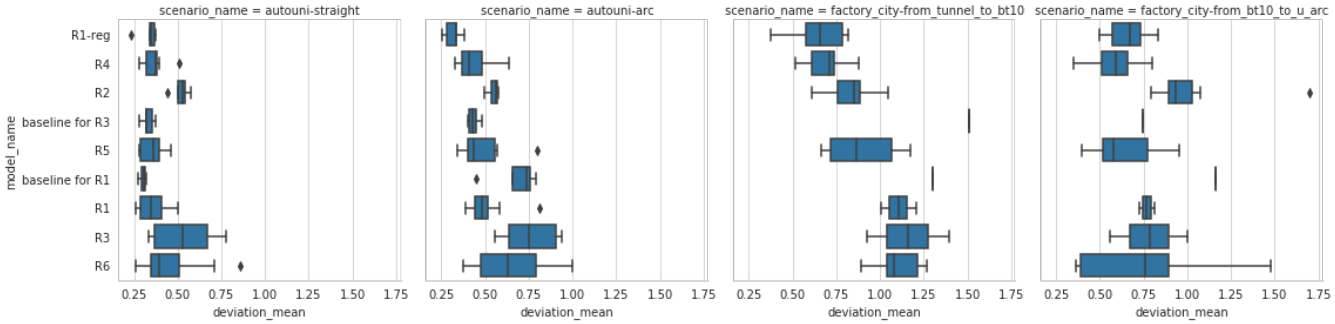

Fig. 4: Average deviation of models from expert trajectories. Measurements based on GPS. The project website `http://bit.ly/2maqbJj` contains information on all scenarios.

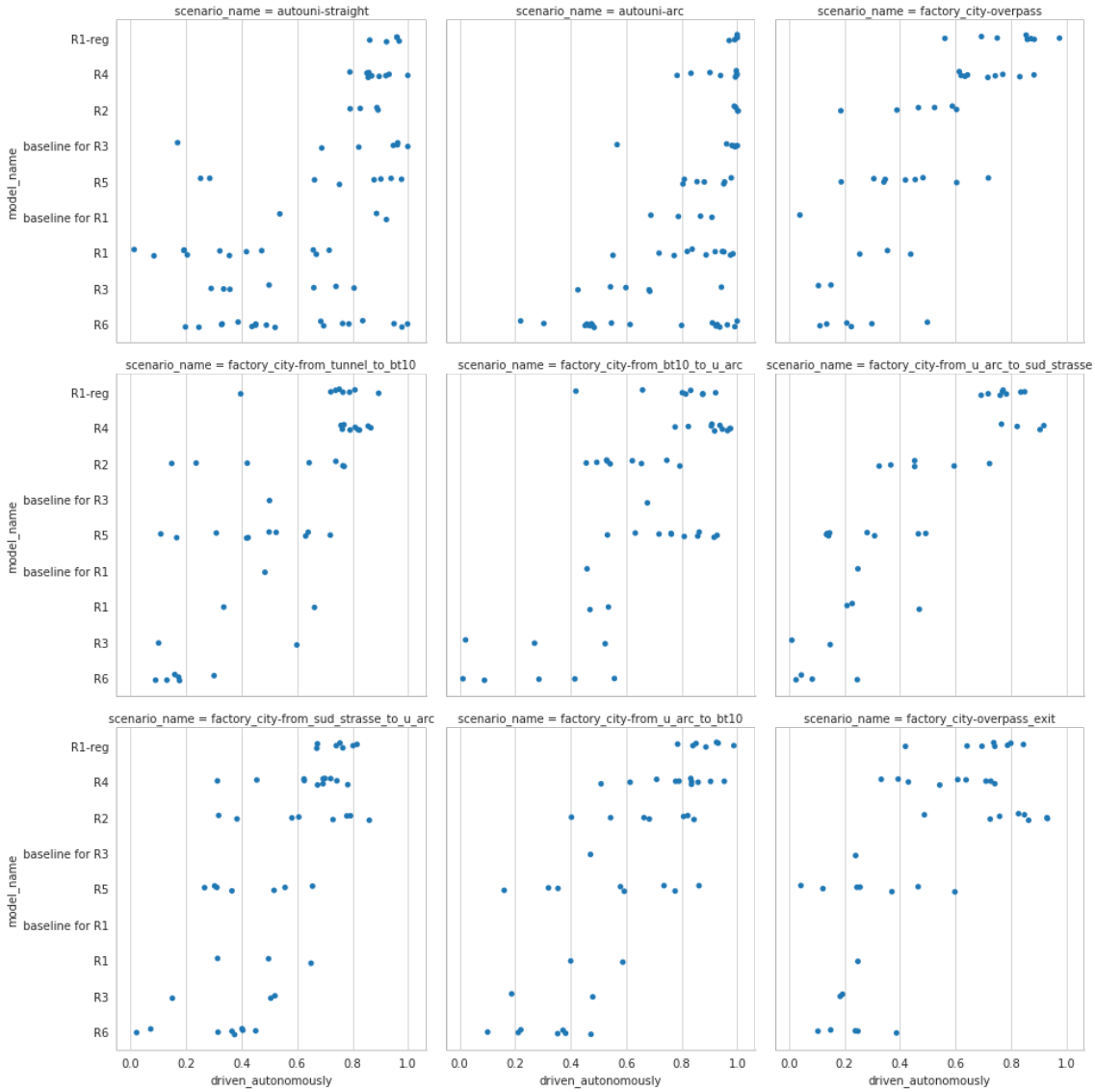

Fig. 6: Summary of experiments R1-R6 across 9 scenarios with baselines. Each subfigure represents performance for a given deployment scenario.

our neural network predicts depth. The depth prediction is learned in a supervised way, along with the RL training. This auxiliary task slightly speeds the training in simulation. In real-world evaluations, model R5 showed slight improvement over R1 and baseline R1 experiments.

Experiment R6 – segmentation-only: Similarly to [38] we test hypothesis that segmentation is a useful common representation space for the simulation to real transfer. In our

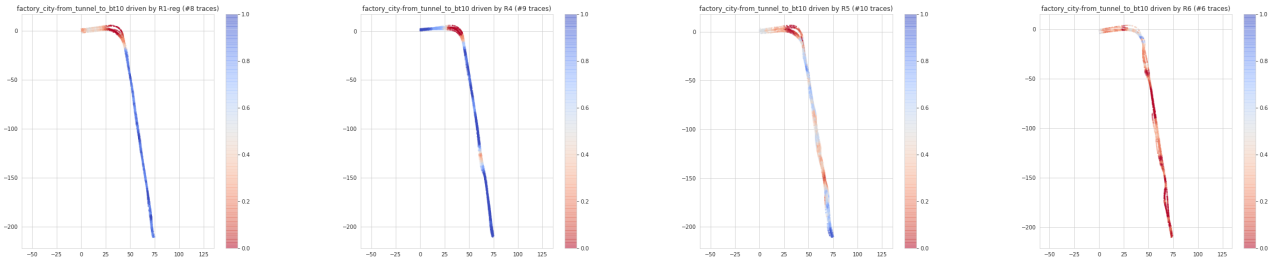

Fig. 7: Qualitative comparison of autonomy with two good models R1-reg and R4 along with two lagging models R5 and R6. The graphs show the scale between blue (full autonomy in all experiments) and red (human assistance in all experiments). Graphs for other models and other routes are available via the project webpage `http://bit.ly/2maqbJj`.

R6 experiment we trained models taking only segmentation as input, for which we recorded overall weak performance. We speculate that in our case, the RGB input carries crucial information, not present in segmentation (e.g. about depth).

C. Selected failure cases

1) Single-line versus double-line road markings : In initial experiments we have used CARLA's TOWN1 and TOWN2 maps which feature only double-line road markings. When evaluated on real-world footage, such a policy was not sensitive to single-line road markings, whereas it was sensitive to double-line road markings in simulation.

This problem was fixed after introducing our custom CARLA maps which feature a single-line road markings.

2) Bug in reward function resulting in driving over the curb: Our reward functions includes a term that penalizes for not sticking to the center of a lane. In our initial implementation distance used for calculating the penalty was using all X, Y and Z spatial coordinates. Due to technical reasons our list of lane-center positions was actually placed above the road in the Z axis. This resulted in a policy that drives with two right side wheels placed on a high curb so its elevation is increased and distance to the center-line point above the ground is decreased. The fix was to calculate penalty using only X and Y coordinates.

D. Offline models evaluation

A fundamental issue in sim-to-real experiments is that good performance in simulation does not necessarily transfer to real-world. It is aggravated by the fact that real-world testing is costly both in time and resources. Inspired by [42] we introduced a proxy metric, which can be calculated offline and correlates with real-world evaluations. Namely, for seven scenarios with prefix `factory_city` we obtained a human reference drive. Frame by frame, we compared the reference steering with the one given by our models calculating the mean square error, mae. We observe a clear trend, see Figure 8. While this result is still statistically rather weak, we consider it to be a promising future research direction. We present an additional F_1 metric and more details on the project webpage `http://bit.ly/2maqbJj`.

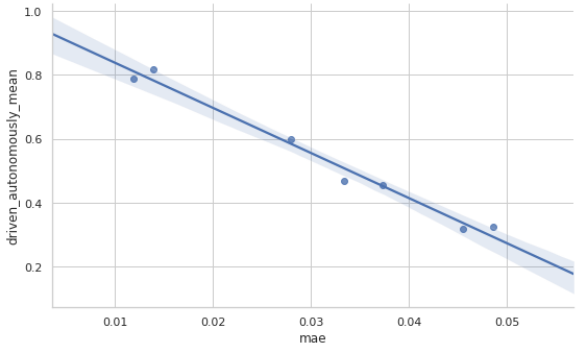

Fig. 8: Dependence of mean $driven\_autonomously$ metric (for all models with exception of R2 and R5) on their mae with the reference drives. Model R2 is not included due to the different action space and we were unable to process all data for model R5 – new figure with 8 reference dots will be available later via the project webpage `http://bit.ly/2maqbJj`.

V. Conclusions and future work

We presented a comprehensive overview of a series of experiments intended to train an end-to-end driving policy using the CARLA simulator. Our policies were deployed and tested on a full-size car exhibiting substantial level of autonomy in a number of restricted driving scenarios.

The current results let us to speculate about the following promising directions: using more regularization, utilizing continuous action spaces and waypoints and using off-line proxy metric. While we obtained poor results with memory-augmented architectures, we plan to investigate the topic further.

We also consider other training algorithms which use a replay buffer such as V-trace [3] and SAC [43]. The asymmetric actor-critic architecture presented in [12] and a generator-discriminator pair similar to the one in [44] can be also beneficial for training of driving polices. Another interesting and challenging direction is integration of an intermediate representation layer — for example a 2D-map or a bird's-eye view, as proposed in [27], [28], [45], [46]. Focusing RL training on fragments of scenarios with the highest uncertainty, see, e.g., [47] might improve driving stability. Integration of model-based methods similar to [48] would be a desirable step towards better sample efficiency.

References

[1] F. Codevilla, M. Müller, A. Dosovitskiy, A. López, and V. Koltun, "End-to-end driving via conditional imitation learning," *CoRR*, vol. abs/1710.02410, 2017. [Online]. Available: http://arxiv.org/abs/1710.02410

[2] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv oloreprint arXiv:1711.03938*, 2017.

[3] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu, "IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures," *CoRR*, vol. abs/1802.01561, 2018. [Online]. Available: http://arxiv.org/abs/1802.01561

[4] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," *CoRR*, vol. abs/1803.00933, 2018. [Online]. Available: http://arxiv.org/abs/1803.00933

[5] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *arXiv preprint arXiv:1808.00177*, 2018.

[6] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.

[7] I. Adamski, R. Adamski, T. Grel, A. Jedrych, K. Kaczmarek, and H. Michalewski, "Distributed deep reinforcement learning: Learn how to play atari games in 21 minutes," *CoRR*, vol. abs/1801.02852, 2018. [Online]. Available: http://arxiv.org/abs/1801.02852

[8] M. Toromanoff, É. Wirbel, and F. Moutarde, "Is deep reinforcement learning really superhuman on atari?" *CoRR*, vol. abs/1908.04683, 2019. [Online]. Available: http://arxiv.org/abs/1908.04683

[9] D. Pomerleau, "ALVINN: an autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems 1, [NIPS Conference, Denver, Colorado, USA, 1988]*, 1988, pp. 305–313.

[10] F. Sadeghi and S. Levine, "(cad)^{2}rl: Real single-image flight without a single real image," *CoRR*, vol. abs/1611.04201, 2016. [Online]. Available: http://arxiv.org/abs/1611.04201

[11] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," *CoRR*, vol. abs/1707.02267, 2017. [Online]. Available: http://arxiv.org/abs/1707.02267

[12] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," in *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, 2018. [Online]. Available: http://www.roboticsproceedings.org/rss14/p08.html

[13] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8460528

[14] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, W. Zaremba, and P. Abbeel, "Domain randomization and generative models for robotic grasping," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, 2018, pp. 3482–3489. [Online]. Available: https://doi.org/10.1109/IROS.2018.8593933

[15] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, 2012, pp. 5026–5033. [Online]. Available: https://doi.org/10.1109/IROS.2012.6386109

[16] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," *CoRR*, vol. abs/1610.04286, 2016. [Online]. Available: http://arxiv.org/abs/1610.04286

[17] A. Bewley, J. Rigley, Y. Liu, J. Hawke, R. Shen, V. Lam, and A. Kendall, "Learning to drive from simulation without real world labels," *CoRR*, vol. abs/1812.03823, 2018. [Online]. Available: http://arxiv.org/abs/1812.03823

[18] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," *CoRR*, vol. abs/1812.07252, 2018. [Online]. Available: http://arxiv.org/abs/1812.07252

[19] F. Sadeghi, A. Toshev, E. Jang, and S. Levine, "Sim2real view invariant visual servoing by recurrent control," *CoRR*, vol. abs/1712.07642, 2017. [Online]. Available: http://arxiv.org/abs/1712.07642

[20] F. Sadeghi, "Divis: Domain invariant visual servoing for collision-free goal reaching," *CoRR*, vol. abs/1902.05947, 2019. [Online]. Available: http://arxiv.org/abs/1902.05947

[21] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," *CoRR*, vol. abs/1705.05065, 2017. [Online]. Available: http://arxiv.org/abs/1705.05065

[22] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *CoRR*, vol. abs/1704.02532, 2017. [Online]. Available: http://arxiv.org/abs/1704.02532

[23] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. A. Riedmiller, and D. Silver, "Emergence of locomotion behaviours in rich environments," *CoRR*, vol. abs/1707.02286, 2017. [Online]. Available: http://arxiv.org/abs/1707.02286

[24] L. Kidzinski, S. P. Mohanty, C. F. Ong, Z. Huang, S. Zhou, A. Pechenko, A. Stelmaszczyk, P. Jarosik, M. Pavlov, S. Kolesnikov, S. M. Plis, Z. Chen, Z. Zhang, J. Chen, J. Shi, Z. Zheng, C. Yuan, Z. Lin, H. Michalewski, P. Milos, B. Osinski, A. Melnik, M. Schilling, H. J. Ritter, S. F. Carroll, J. L. Hicks, S. Levine, M. Salathé, and S. L. Delp, "Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments," *CoRR*, vol. abs/1804.00361, 2018. [Online]. Available: http://arxiv.org/abs/1804.00361

[25] N. Rhinehart, R. McAllister, and S. Levine, "Deep imitative models for flexible inference, planning, and control," *CoRR*, vol. abs/1810.06544, 2018. [Online]. Available: http://arxiv.org/abs/1810.06544

[26] A. Sauer, N. Savinov, and A. Geiger, "Conditional affordance learning for driving in urban environments," in *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, 2018, pp. 237–252. [Online]. Available: http://proceedings.mlr.press/v87/sauer18a.html

[27] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," *CoRR*, vol. abs/1904.09503, 2019. [Online]. Available: http://arxiv.org/abs/1904.09503

[28] N. Rhinehart, R. McAllister, and S. Levine, "Deep imitative models for flexible inference, planning, and control," *CoRR*, vol. abs/1810.06544, 2018. [Online]. Available: http://arxiv.org/abs/1810.06544

[29] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1-2, pp. 1–142, 2013. [Online]. Available: https://doi.org/10.1561/2300000021

[30] N. Sünderhauf, O. Brock, W. J. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, and P. Corke, "The limits and potentials of deep learning for robotics," *CoRR*, vol. abs/1804.06557, 2018. [Online]. Available: http://arxiv.org/abs/1804.06557

[31] K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, "Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight," *CoRR*, vol. abs/1902.03701, 2019. [Online]. Available: http://arxiv.org/abs/1902.03701

[32] B. Tan, N. Xu, and B. Kong, "Autonomous driving in reality with reinforcement learning and image translation," *CoRR*, vol. abs/1801.05299, 2018. [Online]. Available: http://arxiv.org/abs/1801.05299

[33] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[34] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *International Conference on Computer Vision (ICCV)*, 2017. [Online]. Available: https://www.mapillary.com/dataset/vistas

[35] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *International Conference on Machine Learning*, 2019, pp. 1282–1289.

[36] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "OpenAI Baselines," https://github.com/openai/baselines, 2017.

[37] 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 *Robotics: Science and System XIV*, T. H. Hadas Kress-Gazit, Siddhartha Srinivasa and N. Atanasov, Eds., 2018.

[38] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," *CoRR*, vol. abs/1804.09364, 2018. [Online]. Available: http://arxiv.org/abs/1804.09364

[39] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1–8.

[40] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[41] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. [Online]. Available: https://openreview.net/forum?id=SJ6yPD5xg

[42] F. Codevilla, A. López, V. Koltun, and A. Dosovitskiy, "On offline evaluation of vision-based driving models," *CoRR*, vol. abs/1809.04843, 2018. [Online]. Available: http://arxiv.org/abs/1809.04843

[43] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *CoRR*, vol. abs/1801.01290, 2018. [Online]. Available: http://arxiv.org/abs/1801.01290

[44] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," *CoRR*, vol. abs/1709.07857, 2017. [Online]. Available: http://arxiv.org/abs/1709.07857

[45] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F. Chou, T. Lin, and J. Schneider, "Motion prediction of traffic actors for autonomous driving using deep convolutional networks," *CoRR*, vol. abs/1808.05819, 2018. [Online]. Available: http://arxiv.org/abs/1808.05819

[46] M. Bansal, A. Krizhevsky, and A. S. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *CoRR*, vol. abs/1812.03079, 2018. [Online]. Available: http://arxiv.org/abs/1812.03079

[47] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," in *British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4-7, 2017*, 2017. [Online]. Available: https://www.dropbox.com/s/jgozsaobbk98azy/0205.pdf?dl=1

[48] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," *CoRR*, vol. abs/1811.01848, 2018. [Online]. Available: http://arxiv.org/abs/1811.01848