Learning to drive fast on a DuckieTown highway

Thomas Wiggers and Arnoud Visser

Intelligent Robotics Lab, University of Amsterdam, The Netherlands

Abstract. We train a small Nvidia AI JetRacer to follow the road on a small DuckieTown highway. In the real-world, roads do not always have the same appearance, so the system should not be trained on lane markings alone but on the complete view of the front camera. To make this possible, the system is trained in simulation using a recent reinforcement learning approach in an end-to-end fashion. This driving experience is then transferred to the circumstances encountered on a real track. Transfer learning is surprisingly successful, although this method is very sensitive to the details of the vehicle dynamics. We trained multiple models at different speeds and evaluated their performance both in simulation and in the real world. Increasing the velocity proves difficult, as the learned policy breaks down at higher speeds. The result is a small Nvidia AI JetRacer, which is able to drive around a DuckieTown highway, based on simulated experiences.

Keywords: Reinforcement Learning · Proximal Policy Optimization · Autonomous Driving

1 Introduction

Autonomous driving has the potential to become one of the most impactful developments of the 21^{st} century [2]. In recent years, many advances have been made in the pursuit of fully self-driving cars, most notably in artificial intelligence research [4]. One of the main challenges posed in this pursuit is the safety concerning these systems in the real-world [5]. The lack of transparency in these systems means they are difficult to evaluate and verify [21].

With the Meaningful Control of Autonomous Systems initiative (MCAS)[1], we aim to accelerate research on the topic. To do so, an environment is needed which allows experimenting with different perception, decision and control algorithms.

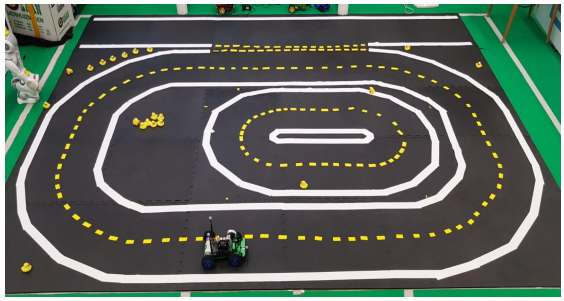
The Duckietown [17] platform aims to be a friendly and safe environment to test and demonstrate artificial intelligence (AI) techniques for autonomous driving. The platform provides small vehicles, DuckieBots, and standardized worlds, DuckieTowns, where the DuckieBots can drive around. Together with the DuckieTown simulator [6] the platform provides the infrastructure for both simulated and real-world autonomous driving research.

Because the current automated vehicles like Tesla Autopilot provide only support on highways, we modified the DuckieTown standard world (inner track in Fig. 1a) into a DuckieTown highway (outer track in Fig. 1a). We also upgraded

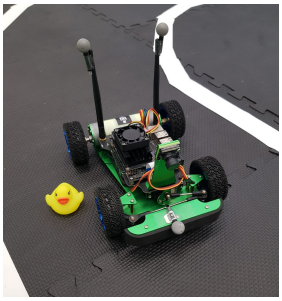
the DuckieBot with a faster front wheel steering vehicle, the Nvidia AI JetRacer[2] shown in Fig. 1b. Recently DuckieTown also introduced a DuckieBot version (DB21)[3] which has the Nvidia Jetson Nano board as computing platform. This makes our development directly applicable to the new generation of DuckieBots, although the three wheeled DB21 version cannot reach the same speeds as the Nvidia AI Jetracer.

Reinforcement learning techniques have been shown to be able to learn to drive autonomously [11]. It has been shown that Proximal Policy Optimization (PPO) can be used to solve different types of challenges, such as the OpenAI Gyms[4], more efficiently than previous techniques. Moreover, PPO methods are simpler to implement and offer better sample complexity than previous methods [20].

As part of the MCAS initiative, we train a model in an end-to-end fashion in a simulator using recent reinforcement learning techniques to control a real-world vehicle around a track. The contribution of this paper is the use of the more recent reinforcement learning technique PPO in combination with the transfer to the real-world domain.



(a) The DuckieTown highway used for real-world evaluation. The outer track circle is used for the JetRacer.



(b) The JetRacer vehicle used for real-world evaluations.

Fig. 1: The hardware setup of our experiment

2 Related work

The problem of avoiding obstacles and keeping to a traffic lane has a long history of research. In 1989 Pomerleau [18] showed with ALVINN that neural networks have the capacity to control a vehicle based on its surroundings. The developments of convolutional neural networks (CNN) in this century enabled the processing of higher resolution images [14], which allows the usage of the full resolution of the frontal camera.

[2] https://www.waveshare.com/jetracer-ai-kit.htm
[3] https://www.duckietown.org/about/hardware
[4] https://gym.openai.com/

End-to-end imitation learning is a highly effective technique that can learn both the representations and policy to solve a given task, without the need for expert knowledge [7]. This is demonstrated by Bojarski et al. [3] with their DAVE-2 vehicle. Leveraging the recent increases in computing power, their system is fully capable of driving in traffic.

In [11] the authors discuss that while RL has great potential, the lack of literature and large scale datasets prevents mature solutions. Specifically, domain adaptation from a simulator to the real-world is often challenging [11]. The simulation-reality gap thoroughly tests model generalization, as the model must both be feature-level and policy-level domain invariant. Steps to reduce the gap can be taken by transforming virtual images into synthetic real images [16]. Promising results are also shown by using a more realistic simulator, the World Rally Championship 6 game, as demonstrated by Jaritz *et al* [10].

Training a self-driving car to stay on the road with Proximal Policy Optimization (PPO) has been demonstrated by Holen *et al* [8]. However, the transfer learning was from 2D simulation to 3D (and vise versa), while in this paper it is demonstrated for 3D simulation to a real vehicle.

Learning to drive in the DuckieTown world, and its simulator, has already been approached by many researches and competitors of the Arfiticial Intelligence Driving Olympics (AI-DO). In the first edition of AI-DO [23], the top contestants used multiple approaches, ranging from classical to reinforcement learning (RL). Classical approaches, such as the Hough transform for line detection, are still competitive in 2018. From the results of the competition, it is clear that transferring to the real-world poses challenges. Most participants did not survive for more than 3 seconds. As each participant is only given 5 trials, it is difficult to draw any conclusions on whether the learning approaches performed better than the classical ones.

The imitation learning approach by Team JetBrains is elaborated in [19]. A simple convolutional neural network is trained in an end-to-end fashion, with their best result trained on both real-world and simulated data. This method managed to traverse 30 tiles in 60 seconds in their real-world testing. Using this method, the team managed first place in the 2019 competition.

A limitation of imitation learning is that it has difficulty generalizing to unseen scenarios [9]. The policy is limited to learning from the data distribution of the dataset generated by the teacher. Allowing the agent to optimize its policy in the target environment could therefore yield better generalization.

Reinforcement learning offers a framework for learning in such environments. Reinforcement learning has been implemented in the DuckieTown environment by [1]. In this paper, a deep reinforcement learning approach using Deep Q-Networks (DQN) is presented. The agent is trained in the simulator, and then evaluated in the real-world. Their results show that the approach is comparable to the 2019 winners, team JetBrains.

As DQN can only operate in the discrete domain, the robot is controlled using three actions. However, the agent may benefit from continuous control over the

vehicle. Recent techniques such as PPO have been shown to outperform DQN, and are capable of learning in the continuous domain.

3 Method

To drive the vehicle in its lane on the road, a controller is required. Traditionally, handcrafted policies are used to determine the appropriate steering angle for the situation. We use RL to learn the appropriate control policy using a convolutional neural network.

3.1 Reinforcement learning framework

In [15] discrete control is shown to perform poorly in the target domain, as it suffered from severe side-to-side wobbling. This wobbling can be observed as well in the results from [10], which also implemented a discrete action space. While discretization has often been shown to improve learning, in the context of steering a car continuous control might still be preferable.

Proximal Policy Optimization (PPO) is a policy gradient method for reinforcement learning. This method optimises a surrogate objective function using stochastic gradient ascent. This offers benefits over previous methods and allows for better sample complexity. We use PPO as it is capable of implementing continuous action spaces.

The DuckieTown simulator is based on OpenAI's Gym framework and is a drop-in replacement with existing frameworks such as OpenAI baselines. While the simulator fidelity is quite low, the real-world DuckieTown environment is rather simple, featuring large road markings in an indoor environment. It does not often feature visual effects such as shadows, water reflections, or sun flare effects, reducing the need for a photo-realistic simulator. We use PPO to train the full network in the DuckieTown simulator.

3.2 Network architecture

We use a simple architecture due to the comparatively low complexity of our environment. This architecture is adapted from [12], and is capable of solving the CarRacing-v0 environment. Furthermore, this reduces computational needs and training duration. The network architecture is shown in Table 1. The input of the model consists of the last 4 frames in grayscale. The network outputs parameters α and β for a beta distribution from which the steering angle is sampled. A separate head is used as a critic for PPO.

Layer	Size	Channels	Kernel	Stride	Activation
Input	96x96	4			
Conv1	96x96	8	4	2	ReLU
Conv2	47x47	16	3	2	ReLU
Conv3	23x23	32	3	2	ReLU
Conv4	11x11	64	3	2	ReLU
Conv5	5x5	128	3	1	ReLU
Conv6	3x3	256	3	1	ReLU
Flatten	256				
Layer	Input	Output			Activation
Critic Fc1	256	100			ReLU
Critic Fc2	100	1			
Actor Fc1	256	100			ReLU
Actor Fc2 α	100	1			Softplus
Actor Fc2 β	100	1			Softplus

Table 1: The network architecture used to drive autonomously.

3.3 Reward function and setup

A similar reward function to [1] is used, shown in Equation 1. This is similar to previous work, such as [10]. The reward function incorporates both the angle θ to the lane, and distance d to the center of the track, shown in Fig. 2, as well as a base negative reward. In contrast to Jaritz et al. [10], the reward is not scaled by the velocity v as we use a fixed velocity. In addition, a small negative offset with a value of 0.1 is included to incentivize action.

$$\begin{aligned} \text{Ours} \quad & R = \cos(\theta) - d - 0.1 \\ \text{Jaritz et al. [10] and Almási et al. [1]} \quad & R = v(\cos(\theta) - d) \\ \text{Mnih et al. [13]} \quad & R = v(\cos(\theta)) \end{aligned} \tag{1}$$

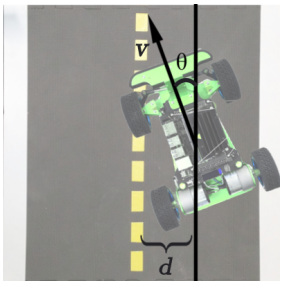


Fig. 2: Reward function symbols seen from the center of the JetRacer

The state space consists of the last 4 frames, to provide the model with more information about its position. These frames are downscaled to 96x96 gray-scale

images and are rendered using the same 160° field of view as the camera module present on the JetRacer.

To simplify the task and allow for training and evaluation at different speeds the velocity is fixed, and only the steering angle is predicted by the network. Furthermore, each action is repeated 8 frames, which helps improve driving behavior as shown by [22]. Each episode consists of at most 500 timesteps, or until the vehicle crosses the road markings after which the episode is ended with a penalty. In 500 timesteps a perfect agent is able to perform at least one lap around large maps. To prevent unfairly difficult starting positions, the vehicle is oriented to within 4°offset to the lane direction at the start of each episode. For all experiments, the simulator is set to render images at 30 frames per second. The hyper-parameters used for training are summarized in Table 2.

Parameter	Value
Learning rate	0.001
Replay buffer size	2000
Batch size	128
Gamma	0.99
Optimization epochs	10
Max gradient norm	0.5
Value function clip param	0.1

Table 2: The hyper-parameters used for training using PPO.

We experimented using three models trained in three different circumstances. The first model, Baseline, is trained on the "Straight road" map at 0.14m/s to provide a baseline. The second model, MediumSpeed, is trained on the "Zigzag" map at a fixed speed of 0.14m/s. The final model, HighSpeed, is also trained on the "Zigzag" map with a higher speed of 0.35m/s, which allows for real-world evaluations at a higher speed. The "Zigzag" map is used because it offers a good balance between left turns, right turns, and straight roads. It also features some objects and vehicles outside of the track as a distraction. All maps are illustrated in Fig. 3.

To evaluate the performance of the trained models we count the number of infractions occurring in 100 episodes. An infraction is counted whenever the agent touches the outside lines of the road, after which the episode is terminated. The mean and standard deviation of the positional error is also measured and presented relative to the width of the road, to indicate the models' ability to follow the lane. The positional error is measured as the difference between the agent position to the center of the lane it must follow. The final evaluation is performed on the real-world DuckieTown highway using the JetRacer. This world has been adjusted to have larger corners, which are necessary for the JetRacer due to its large turning radius. The DuckieTown highway is comparable in length to the "Loop empty" map. The number of infractions during the attempts at the laps are counted. Similar to the episodes in the simulation, after each infraction

the JetRacer is reset to its starting position. A lap is therefore only successful and counted if the JetRacer fully completes the track from its starting position.

For the real-world evaluation, we deviate slightly from other papers. As the JetRacer is wider than a standard Duckiebot, moves faster, and is limited in its turning radius, we did not count crossing into the opposing lane and only considered leaving the track as an infraction.

4 Results

Using reinforcement learning three different models have been trained to control the vehicle. Even after hours of training mistakes are still made by the model. The models are evaluated by counting the number of infractions occurring in 100 episodes. An infraction is counted whenever the agent touches the outside lines of the road. Fewer infractions indicate the model learned a better driving policy.

4.1 Simulator evaluation

In Table 3 the number of infractions in 100 episodes is shown, as well as the lane position error mean and standard deviation relative to the width of the road. These metrics provide insight into how well the agent was able to maintain the correct position on the lane.

Surprisingly, the Baseline model trained only on the "Straight road" map generalizes very well. The model performs well on unseen maps that include turns, and even maps which include intersections. This is likely due to the many corrections and recoveries needed to learn to follow the straight track during training.

The models MediumSpeed and HighSpeed learn to stay on the road quite well. They have not learned to drive perfectly, as in some of the corners the models struggle to recover from errors and exceeds the track limits. In maps

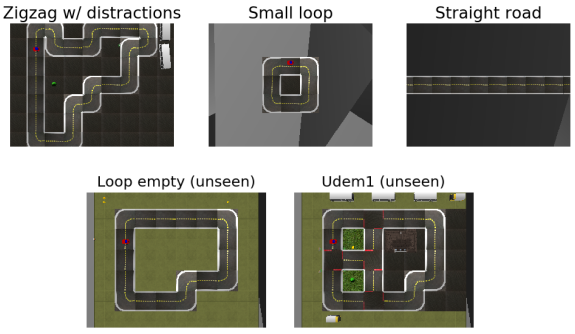


Fig. 3: DuckieTown maps used for training.

Model name	Speed	Evaluation map	Infractions	Lane pose mean error	Lane pose std dev
Baseline	0.14m/s	udem1	36/100	7.3 %	12.5 %
Baseline	0.14m/s	Zigzag	74/100	9.2 %	15.5 %
Baseline	0.14m/s	small loop	29/100	9.9 %	15.8 %
Baseline	0.14m/s	loop empty	29/100	5.7 %	12.2 %
MediumSpeed	0.07m/s	Udem1	12/100	14.0 %	16.9 %
MediumSpeed	0.07m/s	Zigzag	21/100	16.7 %	18.6 %
MediumSpeed	0.07m/s	Small loop	**0/100**	20.4 %	17.4 %
MediumSpeed	0.07m/s	Loop empty	**12/100**	13.6 %	17.3 %
MediumSpeed	0.14m/s	Udem1	**8/100**	7.0 %	11.6 %
MediumSpeed	0.14m/s	Zigzag	30/100	10.4 %	15.8 %
MediumSpeed	0.14m/s	Small loop	7/100	19.5 %	17.8 %
MediumSpeed	0.14m/s	Loop empty	25/100	14.1 %	18.1 %
MediumSpeed	0.28m/s	Udem1	48/100	7.2 %	12.6 %
MediumSpeed	0.28m/s	Zigzag	80/100	12.2 %	17.5 %
MediumSpeed	0.28m/s	Small loop	40/100	8.5 %	14.4 %
MediumSpeed	0.28m/s	Loop empty	38/100	7.9 %	14.8 %
HighSpeed	0.35m/s	Udem1	27/100	6.1 %	7.7 %
HighSpeed	0.35m/s	Zigzag	**17/100**	3.5 %	5.9 %
HighSpeed	0.35m/s	Small loop	22/100	3.5 %	6.4 %
HighSpeed	0.35m/s	Loop empty	19/100	4.4 %	5.8 %
HighSpeed	0.42m/s	Udem1	28/100	6.0 %	7.3 %
HighSpeed	0.42m/s	Zigzag	43/100	4.7 %	8.4 %
HighSpeed	0.42m/s	Small loop	49/100	5.9 %	9.8 %
HighSpeed	0.42m/s	Loop empty	62/100	5.0 %	6.7 %
MediumSpeed	∼0.5 m/s	DuckieTown Highway (real-world)	**10/100**	-	-
HighSpeed	∼0.5 m/s	DuckieTown Highway (real-world)	100/100	-	-

Table 3: Infractions and lane position accuracy score's for trained models. Each map is evaluated for 100 episodes and 500 timesteps. Best result for each map is highlighted.

without difficult corners where recovery errors are less likely, such as "Small loop" the agent makes few mistakes. These errors are often introduced due to side-to-side "wobbling" behavior on straight roads. On the unseen map "Loop empty", which is similar to "Zigzag", the model performance is equal. An example of the driving behavior can be seen in this video[5].

Once trained, the models perform surprisingly well on the "Udem1" map, which introduces unseen 3 and 4-way intersections. This is likely because the intersections provide multiple valid directions for the vehicle to continue, and are therefore less susceptible to imperfect positioning due to wobbling. An example of the resulting driving behavior on this map can be seen in this video[6].

Increasing the velocity causes both the MediumSpeed and HighSpeed agents to crash much more often, as shown in table 3. This effect has slightly been mitigated by reducing the action repetition from 8 to 4 during evaluation, resulting in better performance at speeds above training speed. Reducing the speed also reduced the number of infractions incurred by the MediumSpeed model.

4.2 Real-world evaluation

The real-world performance of the MediumSpeed model is tested using the JetRacer and the DuckieTown Highway. The base speed of the JetRacer is relatively high, at around 0.5m/s. This higher speed is compensated by processing as many frames as possible, around 15 per second.

The real-world evaluation was performed using the MediumSpeed model, as it is the best model that transferred to the real-world. The JetRacer performed the laps at its base speed, as increasing the speed renders the model incapable of completing a lap. The HighSpeed model failed to transfer to the real-world as it fails to complete a single lap at all tested speeds.

The JetRacer completed 100 laps, in which it caused 10 infractions by leaving the track. These infractions are often caused by poor positioning going into the corners of the track, due to some wobbling behavior on the straights. An example of a lap can be seen in this video [7].

5 Discusion

In the previous work of [1], the agents are shown to have a success rate of 96% in the real-world. Our approach shows a 90% success rate, which is competitive with to the result of [1] given the stochastic nature of these tests.

The agents presented in AI-DO are capable of driving 13 tiles in 30 seconds [23]. The 2019 winners are capable of driving up to 19 tiles. Given that the tiles are 0.42m, this gives a speed estimate of 0.16 m/s to 0.26 m/s. The base speed

[5] Driving on the Zigzag map used for training: https://youtu.be/7cju-CylTHQ
[6] Driving on the unseen Udem1 map: https://youtu.be/7xWd9aIqzmg
[7] Driving in the real-world Duckietown Highway: https://youtu.be/hi1qfQrnXq4

of the JetRacer is significantly faster than this, driving at almost 0.5 m/s. While our agent drives significantly faster in the real world, these figures are not directly comparable due to the relaxed rules concerning lane infractions.

In Osiński et al. [15] it is shown that discrete control performs poorly to stay on the road. Simply introducing continuous control doesn't solve this issue. This is likely caused by the reward function, which does not discriminate between an agent that often corrects course erratically and one that corrects more carefully. The wobbling is also seen in our real-world tests, and is often the root cause of later infractions.

Increasing the speed of the vehicle quickly increases the number of infractions caused by the agent. This suggest the learned policies are not robust against changes in vehicle dynamics. This effect is again demonstrated in the real-world evaluations, where the tested models are highly sensitive to the vehicle dynamics. While reducing the action repetition to increase the update frequency can help to improve the speed of the vehicle, it does not directly address the problem arising due to the simulator reality gap.

6 Conclusion

The MediumSpeed model trained using proximal policy optimization is capable of driving autonomously on several maps. Moreover, it performs well on unseen maps with new road types such as intersections. This shows high levels of generalization can be obtained using reinforcement learning. This generalization extends to real-world tracks, as the model can successfully complete multiple laps around the DuckieTown highway.

We show that the approach is comparable to state-of-the-art, but drives significantly faster. Increasing the speed more proves difficult, as policy-level generalization concerning the vehicle dynamics is not readily learned. This is exemplified by the failure of the HighSpeed model to transfer to the real-world. Randomisation of the vehicle dynamics in the simulator could help close the simulator-reality gap, yielding models with a more general policy. These might be able to better control vehicles outside of the vehicle in its training domain, such as high-speed real-world vehicles like the JetRacer.

While training in a simulator enables the use of reinforcement learning, correctly adapting to the real-world domain requires high levels of generalization. The use of intermediate state spaces, such as those produced by segmentation networks as in [14], could further increase the ability of the agent to operate in the real-world.

The trained model and created environment provide a real-world test bed for continued research on safe autonomous driving for the MCAS initiative.

Acknowledgements

Many thanks to Douwe van der Wal and Thomas van Orden for the interesting discussions and helpful suggestions.

References

1. Almási, P., Moni, R., Gyires-Tóth, B.: Robust reinforcement learning-based autonomous driving agent for simulation and real world. preprint arXiv:2009.11212 (2020)
2. Anderson, M.: The road ahead for self-driving cars. IEEE Spectrum **57**(5), 8–9 (2020)
3. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. preprint arXiv:1604.07316 (2016)
4. Broggi, A., Zelinsky, A., Özgüner, Ü., Laugier, C.: Intelligent vehicles. In: Springer Handbook of Robotics, pp. 1627–1656. Springer (2016)
5. Brooks, R.: Robotic cars won't understand us, and we won't cut them much slack. IEEE Spectrum **54**(8), 34–51 (2017)
6. Chevalier-Boisvert, M., Golemo, F., Cao, Y., Mehta, B., Paull, L.: Duckietown environments for openai gym. https://github.com/duckietown/gym-duckietown (2018)
7. Grigorescu, S., Trasnea, B., Cocias, T., Macesanu, G.: A survey of deep learning techniques for autonomous driving. Journal of Field Robotics **37**(3), 362–386 (2020)
8. Holen, M., Saha, R., Goodwin, M., Omlin, C.W., Sandsmark, K.E.: Road detection for reinforcement learning based autonomous car. In: Proceedings of the 2020 The 3rd International Conference on Information Science and System. pp. 67–71 (2020)
9. Hussein, A., Gaber, M.M., Elyan, E., Jayne, C.: Imitation learning: A survey of learning methods. ACM Computing Surveys (CSUR) **50**(2), 1–35 (2017)
10. Jaritz, M., De Charette, R., Toromanoff, M., Perot, E., Nashashibi, F.: End-to-end race driving with deep reinforcement learning. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 2070–2075 (2018)
11. Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A.A., Yogamani, S., Pérez, P.: Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems (2021), preprint arXiv:2002.00444
12. Ma, X.: Car racing with pytorch. https://github.com/xtma/pytorch_car_caring (2019)
13. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning. pp. 1928–1937 (2016), preprint arXiv:1602.01783
14. Muller, U., Ben, J., Cosatto, E., Flepp, B., Cun, Y.L.: Off-road obstacle avoidance through end-to-end learning. In: Advances in neural information processing systems. pp. 739–746 (2006)
15. Osiński, B., Jakubowski, A., Zięcina, P., Miłoś, P., Galias, C., Homoceanu, S., Michalewski, H.: Simulation-based reinforcement learning for real-world autonomous driving. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 6411–6418. IEEE (2020), preprint arXiv:1911.12905
16. Pan, X., You, Y., Wang, Z., Lu, C.: Virtual to real reinforcement learning for autonomous driving. In: British Machine Vision Conference. pp. 11.1–11.13. BMVC (2017), preprint arXiv:1704.03952
17. Paull, L., Tani, J., Ahn, H., Alonso-Mora, J., Carlone, L., Cap, M., Chen, Y.F., Choi, C., Dusek, J., Fang, Y., et al.: Duckietown: an open, inexpensive and flexible platform for autonomy education and research. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 1497–1504 (2017)

18. Pomerleau, D.A.: Alvinn: An autonomous land vehicle in a neural network. In: Advances in neural information processing systems. pp. 305–313 (1989)
19. Sazanovich, M., Chaika, K., Krinkin, K., Shpilman, A.: Imitation learning approach for ai driving olympics trained on real-world and simulation data simultaneously (2020), preprint arXiv:2007.03514
20. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017), preprint arXiv:1707.06347
21. Shalev-Shwartz, S., Shammah, S., Shashua, A.: On a formal model of safe and scalable self-driving cars. preprint arXiv:1708.06374 (2017)
22. Sharma, S., Lakshminarayanan, A.S., Ravindran, B.: Learning to repeat: Fine grained action repetition for deep reinforcement learning (2017), preprint arXiv:1702.06054
23. Zilly, J., et al.: The ai driving olympics at neurips 2018. In: The NeurIPS '18 Competition. pp. 37–68. Springer (2020)