Proceedings of Machine Learning Research 123:77–89, 2020 NeurIPS 2019 Competition and Demonstration Track

F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning

Matthew O'Kelly MOKELLY@SEAS.UPENN.EDU
Hongrui Zheng HONGRUIZ@SEAS.UPENN.EDU
Dhruv Karthik DHRUVKAR@SEAS.UPENN.EDU
Rahul Mangharam RAHULM@SEAS.UPENN.EDU
University of Pennsylvania

Editors: Hugo Jair Escalante and Raia Hadsell

Abstract

The deployment and evaluation of learning algorithms on autonomous vehicles (AV) is expensive, slow, and potentially unsafe. This paper details the F1TENTH autonomous racing platform, an open-source evaluation framework for training, testing, and evaluating autonomous systems. With 1/10th-scale low-cost hardware and multiple virtual environments, F1TENTH enables safe and rapid experimentation of AV algorithms even in laboratory research settings. We present three benchmark tasks and baselines in the setting of autonomous racing, demonstrating the flexibility and features of our evaluation environment.

Keywords: Reinforcement Learning, Autonomous Vehicles, Autonomous Racing

1. Introduction

Unlike supervised learning tasks, the minimal evaluation environment for reinforcement learning-based (RL) approaches to robotics must include more than a labeled dataset. We propose that RL evaluation environments should incorporate: (1) accurate and efficient simulators to reduce the time and cost of testing, (2) accessible hardware to correlate simulation results with real-world performance, and (3) adaptable benchmarks which include strong baselines supported by competition to drive innovation and reduce overfitting. While others (*e.g.* Brockman et al., 2016) have described a subset of these requirements, their efforts classify the validation, realism, and accuracy of simulation tools, and availability of hardware as future goals rather than necessary components. In response, we create an open-source RL evaluation environment (simulator, hardware, benchmarks, baselines, and competitions) based on a scaled autonomous vehicle situated in a racing environment.

Accurate simulation is a necessary component of any *practical* evaluation system, as real-world evaluations are dangerous, expensive, and time consuming. A number of other studies note the sensitivity of RL-based agents to simulation details (*c.f.* Lewis et al., 2019; Henderson et al., 2018). Indeed, like all simulators, ours fails to capture agent vehicle dynamics with perfect fidelity. However, we follow classical system identification methods to actively quantify and minimize this discrepancy with respect to performance on a real hardware platform. Additionally, we propose simple methods to more accurately simulate a 2D LIDAR, IMU and cameras in a photorealistic environment.

Hardware-based experiments must support simulation-based evaluation in order to ensure that an approach performs well in reality. As noted by Yang et al. (2020), the hardware platform used must not limit potential solutions. However, similar benchmarks such as Balaji et al. (2019) impose sensor configuration and training method restrictions, whereas others (Goldfain et al., 2019) are too expensive to be used widely. In contrast, we offer a full-fledged Open-AI Gym API for controlling a real world vehicle *with* the ability to redefine *step/reward* functions in an intuitive manner for numerous sensor configurations. Furthermore, running experiments does not require a special track, as our carefully tuned SLAM packages use the onboard LIDAR to create maps of tracks (eg. a corridor loop), and seamlessly use those maps within simulation. Finally, our hardware is open-source, documented, affordable, and supported by an active community.

In addition to simulation and hardware environments we also define a set of **benchmark tasks** by which to standardize evaluations. The context of these tasks, autonomous racing, aids in the definition of evaluation metrics; unlike day-to-day driving, each episode has a clear winner and is likely to expose the agent to the most difficult regimes of the dynamics. Importantly, the benchmarks naturally include both single agent and multi-agent formulations which are of interest to multiple communities.

Contributions and Organization: In Section 2, we compare our simulation environment and hardware support with existing platforms. In Section 3, we describe in detail the simulators that serve as a virtual evaluation environment, enabling rapid implementation and dissemination of algorithms in an environment where reproducibility is enforced. In Section 4, we describe the low-cost, 1/10th scale vehicle and standardized software infrastructure that supplements our virtual benchmarking tools, enabling the use of a wide range of RL algorithms in reality. In Section 5 and 6, we describe three benchmarks in our environment, spanning continuous control and RL methods.

2. Related work

It is still unknown how deep RL will be used in autonomous vehicles; however, a recent survey (Kiran et al., 2020) highlights numerous promising directions. Nevertheless, the goal of this work is not to describe autonomous driving methods, but rather to build tools which enable a community wide effort. As such, one closely related body of literature includes autonomous vehicle simulators (*e.g* Shah et al., 2017; Dosovitskiy et al., 2017; Quiter and Ernst, 2018; LG Electronics, Inc., 2019). Although these tools are widely used in deep RL research, none specifically include a corresponding vehicle platform. More general simulation environments used within the RL community (*e.g.* Todorov et al., 2012; Coumans and Bai, 2016) utilize multi-body physics. In contrast, our simulator is narrow; it uses simple car-like vehicle dynamics described as continuous ODEs. While our domain of interest is targeted, the task is still challenging due to the non-holonomic behavior of the robot, the use of high-dimensional measurements, and interactions with external dynamic agents.

Other related work includes open-source implementations of full-scale autonomous vehicle stacks (Baidu Apollo Team, 2017; Kato et al., 2018). Our approach differs because we prioritize the use of scaled, inexpensive hardware that does not require special permission or insurance for operation. Of course, there are now many scaled autonomous vehicle projects available to the community (*c.f.* Fishberg et al., 2019; Gonzales et al.; Roscoe, 2019; Srinivasa et al., 2019; Goldfain et al., 2019). The F1TENTH hardware is most similar to RACECAR, and in fact shares some low-level drivers; however, F1TENTH differs in its simulation capabilities which are more easily extended beyond pedagogical settings. Our goal of creating a standardized evaluation system is shared with Balaji et al. (2019). While the DeepRacer platform has a lower upfront cost, the lack of sensing modalities such as LIDAR, non-standard hardware acceleration for neural network components, and closed nature of the simulation system limit the use of DeepRacer in research settings.

3. Simulation Tools

In this section we describe the containerized simulation engine for the 1/10th scale vehicle platform of Section 4. We note that simulated environments include a *reality gap* which is characterized by the discrepancy between the performance of the system obtained in simulation relative to the deployed performance. We describe how our simulator design can mitigate and control factors which influence the reality gap.

3.1. Simulator design

The simulator can operate in two modes: (1) a ROS-based mode designed for software-in-the-loop (SIL) testing and (2) an OpenAI Gym (Brockman et al., 2016) mode designed for distributed training. The OpenAI Gym simulator mode supports the additional option to generate photo-realistic camera observations which enables the testing and training of a computer vision pipeline.

The SIL-oriented simulator is a plug-and-play replacement for the 1/10th scale platform itself. The response of the vehicle to control commands is captured using the single track model described in Althoff et al. (2017). The simulator also replaces the sensors used on the actual vehicle (see Section 4). Synthetic sensors mimic odometry information from the electronic speed controller and laser scans from LIDAR. The same message types and ROS publisher subscriber mechanisms are used to communicate the sensor observations. The primary use-case for this simulation mode is debugging shortly before deployment as implementation errors related to ROS and the vehicles interface can be found safely.

The training-oriented simulator is created with the RL community's needs in mind. In contrast to the SIL mode, training mode explicitly steps time with the given action input in order to create deterministic rollouts. Since the simulator is capable of simulating multiple vehicles and their sensor observations, the API is designed such that all agents are stepped simultaneously. Determinism has two benefits; first, experiments are repeatable, a feature useful both for debugging and in service of our stated goal of creating fair and accurate evaluations. Second, the explicit stepping enables the physics engine to take advantage of faster than real-time execution (up to a 6x speedup per-worker on commodity cloud instances). In addition, we provide a containerized executable of the simulator which includes a ZeroMQ (Hintjens, 2013) interface implementing the MapReduce pattern (Dean and Ghemawat, 2008) for scaling distributed training algorithms. Finally, we implement a bridge to the vehicle's ROS interface which enables easy deployment after training has completed, see Figure 2(c). Unlike other work (Balaji et al., 2019) our system is open source and can be deployed on any cloud-service.

Depending on the sensor payload carried on board the vehicle it may be necessary for the simulator to simulate a camera. Thus, the training-oriented simulator can also

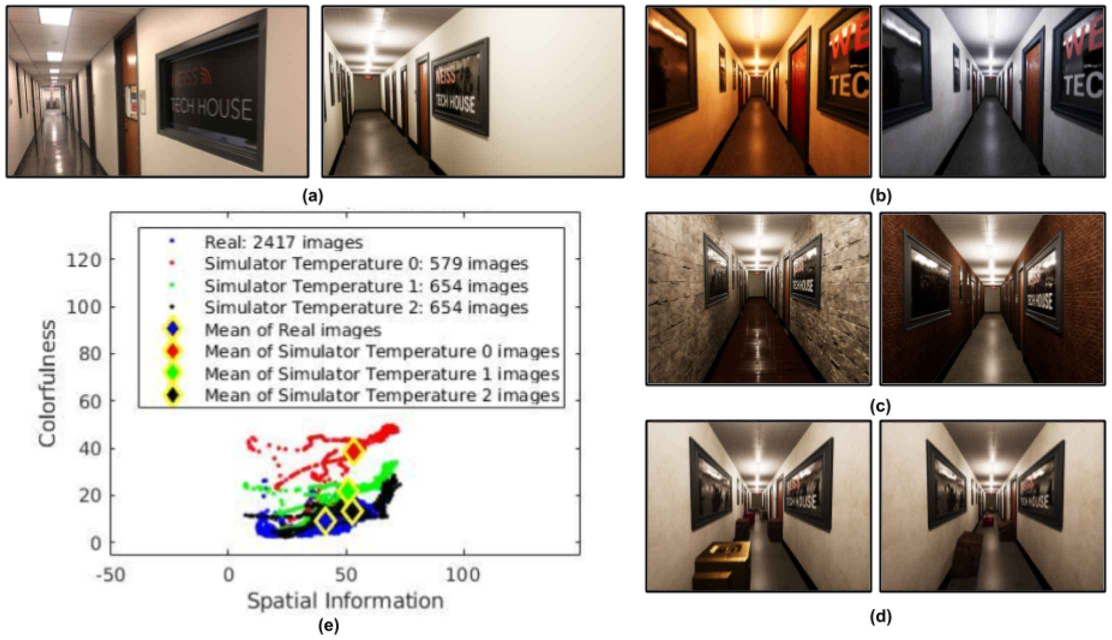
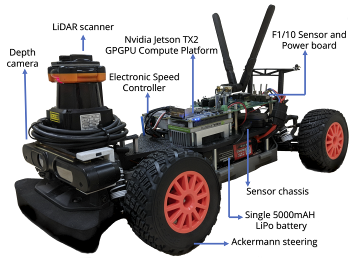

(a)
(b)
(c)
(d)
(e)
Real: 2417 images
Simulator Temperature 0: 579 images
Simulator Temperature 1: 654 images
Simulator Temperature 2: 654 images
Mean of Real images
Mean of Simulator Temperature 0 images
Mean of Simulator Temperature 1 images
Mean of Simulator Temperature 2 images
Colorfulness
Spatial Information
WEISS TECH HOUSE

Figure 1: (a) Left: reality, right: simulation (b) DR via rendering parameters (Temperature) (c) DR via texture replacement (d) DR via static obstacle generation (e) Tuning the rendering parameters (temperature) with natural image statistics

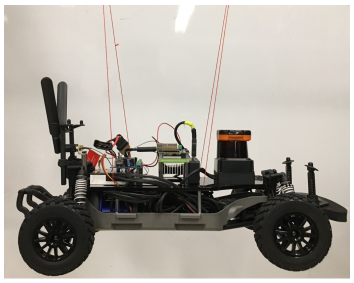
provide synthetic images; our implementation utilizes Unreal Engine 4 which we access via a plugin, UnrealCV (Qiu et al., 2017). We discuss the specifics of our modeling choices via an example environment which mimics a track instance used for experiments (see Figure 1) in Section 3.2. A benchmark and baseline implementation for the training mode simulator with synthetic image generation is described in Section 6.2.

3.2. Addressing the reality gap

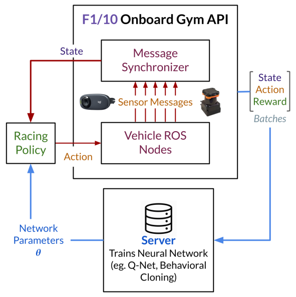
In this section we consider two sources of error in simulated evaluations of vehicle performance. The first source is the vehicle dynamics model itself. The simulator provides variable vehicle model parameters including: mass, center of mass, moment of inertia, dynamic friction coefficient between the four tires and the environment surface, cornering stiffness, and maximum acceleration/deceleration rates. In order to address the reality gap from a system dynamics perspective, the simulator contains parameters for the vehicle and test environment identified (see Figure 2(b)) using the methods outlined in O'Kelly et al. (2019). The second source of error, sensor model fidelity, especially the synthetic camera is a more pernicious factor. We utilize natural image statistics (Kundu, 2016), a measure of the spatial complexity and color content of an image, to quantify the distributional shift between simulation and reality. In order to minimize the gap, we adjust the renderer's lighting temperature. In addition, our plugin implements a variety of domain randomization techniques (Tobin et al., 2017) which include renderer settings, texture replacement, and procedural


LiDAR scanner
Depth camera
Nvidia Jetson TX2 GPGPU Compute Platform
F1/10 Sensor and Power board
Electronic Speed Controller
Sensor chassis
Single 5000mAH LiPo battery
Ackermann steering

(a) F1/10 components



(b) System Identification



(c) Onboard Gym API

Figure 2: Major components of the F1/10 platform and SystemID procedure.

generation of obstacles in order to improve generalization in the real-world . Finally, issues of the environment geometry mismatch are addressed by including a state-of-the-art SLAM front end which enables the capture of maps Hess et al. (2016). Figure 1.(a) compares the same hallway in reality and in the simulation; (b), (c), and (d) in Figure 1 shows different strategies of domain randomization (DR), and Figure 1.(e) details the statistics of real images collected on the vehicle relative to a variety of simulation settings.

4. Hardware Specification and Middleware

The hardware platform, shown in Figure 2(a) F1/10 (2020), consists of two halves: a widely available 1/10-th scale RC car modified to accommodate autonomous control inputs as well as a sensor and compute platform which enables autonomous decision making. The first half, known as the base chassis has 1. A high torque brushless DC motor. 2. An electronic speed controller (VESC) based on an open source design (Vedder, 2015) to convert speed input into motor RPM. 3. A servo motor for controlling the Ackermann steering, 4. A Lithium Polymer (LiPo) battery pack.

The second half is called the sensor and compute platform; it consists of the sensor payload, power management board, and the on-board computer.

The power management board provides a stable voltage source for the compute unit and its peripherals since the battery voltage varies as the vehicle is operated. The sensor suite includes a Hokuyo UST-10LX LIDAR, selected for its durability and high maximum scan frequency of 40Hz. The platform includes options to mount a Intel RealSense D435i which provides RGBD images and an integrated a IMU. Odometry is provided by the VESC. The default compute unit is the Jetson TX2, and there is additional support for the Xavier and Nano variant of the Jetson. The TX2 is mounted on a carrier board to reduce the form factor.

Finally, the platform also includes a open source middleware stack comprising: a driver that provides the interface between on-board ROS nodes and the hardware in order to obtain sensor information and actuate the vehicle; configuration files that define the lower level motor actuation and odometry estimation; and an OpenAI Gym interface as shown in Figure 2(c) that enables quasi-deterministic stepping and resetting of the physical car.

5. Benchmark Tasks and Competitions

The autonomous driving task encompasses a subset of general navigation and locomotion problems. Through the lens of autonomous racing (AR) we further partition the navigation problem into three distinct settings of increasing complexity. Note that our benchmark environments allow the use of alternate rewards for training, but submitted solutions are measured by a simple, sparse reward: lap time.

Benchmark 1: Static Kinodynamic Planning: The first benchmark environment involves a known race-track and a single vehicle; the objective of the autonomous vehicle is to navigate the track in a minimal amount of time; the task is considered successful only if the trajectory is collision free. Many planning algorithms assume that it is possible to observe the vehicle's full state and identify accurate polyhedral representations of obstacles. In practice, an accurate map and localization system can justify the assumption in this task. We describe a baseline solution based on evolution strategies implemented in Section 6.1.

Benchmark 2: Online Kinodynamic Planning: The importance of hardware-based benchmarks (at least as motivation for assumptions) is made more clear if we consider a relaxation of **Benchmark 1**; the map or equivalently the location of obstacles (still polyhedral) is not known prior to runtime. The agent must detect and avoid static obstacles in real time. Furthermore, due to occlusions and sensor properties the vehicle's pose restricts a precise representation of the full map. We describe an imitation learning-based baseline for this problem in Section 6.2.

Benchmark 3: Multi-agent Dynamic Games: Another advantage of the AR setting is that it naturally extends to multi-agent robotics problems. In the third benchmark task we consider two or more agents driving simultaneously on the same map. Interestingly, a subtle variation of the training reward signal induces widely different solutions. If the reward is given as finishing a set number of laps before the opponent, rather than minimizing the time to finish a set number of laps the optimal strategy exhibits blocking behavior not present under the timing based reward (Liniger and Lygeros, 2019). Since we allow arbitrary training reward signals users are free to explore this modality. We detail a suite of control theoretic baselines in Section 6.3.

We curate hardware-tested baseline solutions to the described benchmarks by organizing biannual competitions (Mangharam et al., 2018a,b, 2019a,b). Each competition includes races (**Benchmark 1,3**) and qualification rounds (**Benchmark 2**). We support entrants by providing tools (described in this paper) and teaching material (Agnihotri et al., 2020).[1]

6. Baseline Solutions

6.1. Benchmark 1: Kinodynamic planning

In **Benchmark 1**, the environment consists of the racetrack encoded as a map, a random starting position, and a reward function which measures lap-time. The agent, through interactions with the environment, tunes a parameterized model of its dynamics, strategy, and controller, with the objective of finding an assignment of these parameters that minimizes

1. See `https://github.com/f1tenth/f1tenth_baselines` for baselines and `https://github.com/f1tenth/f1tenth_gym` for simulation tools.

the lap-time. We employ the covariance matrix adaptation evolutionary strategy (CMA-ES) Hansen et al. (2003) optimizer due to the non-smooth objective function landscape. This baseline straddles continuous control and RL methods; the parameters of the vehicle's model predictive controller are tuned based on the results of rollouts which evaluate a candidate solution's quality. A detailed discussion of the learning problem and approach [2] can be found in O'Kelly et al. (2019).

The results comparing simulated lap times and the actual lap times are shown in Table 1. In all the experiments, the lap times measured on hardware rollouts are higher than the simulated lap times. Several factors contribute to this discrepancy: the friction coefficient between the driving surface and the tires is not constant, the steering delay is time-varying, and the accuracy of localization varies depending on the vehicle's location in the map. Each of these factors causes the ego-vehicle to deviate from the optimal trajectory found in simulation. Importantly, however, the local planning and feedback control methods which are presented in greater detail within Section 6.3 allow the solutions found in simulation to remain feasible in the real-world. Moreover, the difference in lap times between sim and real is small, 0.61% for the test track, demonstrating the accuracy of the simulator.

Method	**Lap Time (s, sim)**	**Lap Time (s, real)**
TunerCar	16.2376 ± 0.2161	16.19
Pure Pursuit	17.2307 ± 0.0620	17.00

Table 1: Best Solution Lap Times in Sim vs. Real

6.2. Benchmark 2: Online kinodynamic planning

In **Benchmark 2**, a random number of static obstacles are dispersed around the track which the agent will navigate. The agent does not have prior knowledge of the static obstacles' positions in the map before being deployed. Our baseline implementation for this benchmark illustrates a method to transfer a reactive LIDAR based obstacle avoidance policy to a vision based policy via self-supervised imitation learning (SSIL). The oracle policy, based on the follow the gap method (FGM) (Sezer and Gokasan, 2012) uses the most recent LIDAR scan to avoid the nearest obstacle and steer toward the largest gap. We define I as the stack of input images, S as the snapshot of the current laser scan, a_o as the steering angle output from the oracle policy at time t, and a_v as the steering angle output generated by the vision-based agents at time t. We utilize the network architecture described in (Bojarski et al., 2016) and trained on tuples (I, a_o). Given a trajectory $(I_0, S_0, a_{v,0}), (I_1, S_1, a_{v,1}) \dots (I_T, S_T, a_{v,T})$ we use S_t and FGM to relabel each $a_{v,t}$ to $a_{o,t}$. The relabeled trajectories are used to iteratively correct, via DAgger (Ross et al., 2011), the policy as the vehicle explores the state space.

In the simulated training phase, every rollout occurs in a domain randomized environment where obstacle placement, simulator asset textures, and lighting temperature are varied. Following the simulation-based training phase; the vehicle is deployed in the environment and trained in the same manner *in situ*. After the agent is capable of completing

2. See https://youtu.be/7rOZ1ZgRGqE

the real track without any collision, it is evaluated in the same environment with the learned vision only policy[3]. We show the resulting average lap times of the imitation policy and the oracle policy on the same Philadelphia Track in 6.1 in Table 2. SSIL DAgger produces an average lap time 5% slower than that of the FGM Oracle Policy, indicating that the vision-based policy mimics the LIDAR based policy after sufficient DAgger iterations. Future baselines should explore the ability of imitation learning approaches to exceed the performance of oracle policies.

Method	Avg. Lap time (s, sim)	Avg. Lap time (s, real)
SSIL DAgger	22.625 ± 0.1092	26.71
FGM (Oracle)	21.55 ± 0.0967	18.24

Table 2: Performance of SSIL and DAgger on **Benchmark 2**

6.3. Benchmark 3: Multi-agent dynamic games

For **Benchmark 3** we evaluate reactive, randomized, and model predictive approaches to multi-agent racing. The reactive method uses the same FGM policy described in Section 6.2. The probabalistic planning method is based on rapidly-exploring random trees (RRT) (LaValle, 1998). The RRT planner randomly samples points in the workspace of the vehicle and then expands a tree to find a path between the car's current position and goal position. We also demonstrate a variant of RRT, RRT* (Karaman and Frazzoli, 2011), which rebuilds the tree structure based on the path traversal cost such that the selected path asymptotically approach the global minimum cost. The solutions provided by RRT and RRT* are not inherently dynamically feasible; thus, an additional trajectory tracking controller (Coulter, 1992) is utilized. The model predictive control approach, lattice planning, and has been demonstrated on full scale autonomous vehicles (Urmson et al., 2008). The formulation of the MPC trajectory generation component in the planner is described in McNaughton (2011). Online, the lattice planner samples a set of goal poses in the ego-vehicle's moving reference frame, for each sample a trajectory, parameterized a cubic polynomial, which drives the vehicle from its current state to the goal is computed. Given the set of sampled trajectories, each is evaluated for feasibility and progress. This process repeats in a receding horizon fashion.

Table 3 reports the average lap times, crash rate, and win rate of baseline solutions on **Benchmark 3**. Unlike the baseline provided in the previous section, the algorithms investigated all originate from motion planning and continuous control. In these methods the opponent is treated as a dynamic obstacle rather than an adversarial agent. Thus, although all the baselines have win rate higher than 50%, they all have non-zero crash rates due to their relatively unsophisticated prediction mechanisms relative to RL and game-theoretic approaches (Liniger and Lygeros, 2019). Further work (Sinha et al., 2020) extends the multi-agent benchmark to balance safety and performance within our evaluation environment, providing a suite of diverse adversarial agents.

3. See `https://youtu.be/qq9I2lBAxgI`

Method	**Avg. Lap time (s)**	**Crash Rate**	**Win Rate**
FGM	18.647 ± 2.65	50%	50%
RRT	17.14 ± 2.11	45%	55%
RRT*	13.91 ± 0.64	35%	65%
Lattice Planner	15.59 ± 1.22	30%	70%

Table 3: Performance of Local Planners on Multi-agent benchmark

7. Conclusion

This paper details the F1TENTH platform, an open-source evaluation framework with virtual environments and a low-cost hardware counterpart which enables safe and rapid experimentation suitable for laboratory research settings. We present three benchmark tasks and baselines in the setting of autonomous racing, demonstrating the flexibility and features of our evaluation environment. A limitation of this work is the relatively small sample size of algorithms included in our baseline examples; however, the goal of releasing this environment is to generate a wider variety of community submitted solutions. In the future, we will explore more efficient methods for generating test cases suitable for high-performance agents which fail rarely (*e.g* O'Kelly et al., 2018; Uesato et al., 2018).

Acknowledgments

The authors would like to thank the many contributors to the F1TENTH community. In particular Madhur Behl, Houssam Abbas, Varundev Sukhil, Jack Harkins, Chris Kao, Yash Vardhan Pant, Paolo Burgio, Marko Bertogna, Kim Luong, Kuk Jang, Siddharth Singh, Joe Auckley, Abhijeet Agnihotri, Nischal K.N., Paril Jain, Diana Hoang, Kaiying Guo, Susan Xie, Paritosh Kelkar, Thejas Kesari, Nitesh Singh, Matthew Brady, and Matthew Lebermann have been instrumental in developing and supporting the platform.

References

Abhijeet Agnihotri, Matthew O'Kelly, Rahul Mangharam, and Houssam Abbas. Teaching Autonomous Systems at 1/10th-Scale: Design of the F1/10 Racecar, Simulators and Curriculum. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 657–663, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367936. doi: 10.1145/3328778.3366796. URL https://doi.org/10.1145/3328778.3366796.

Matthias Althoff, Markus Koschi, and Stefanie Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726. IEEE, 2017.

Baidu Apollo Team. Apollo: Open source autonomous driving, 2017. URL https://github.com/ApolloAuto/apollo.

Bharathan Balaji, Sunil Mallya, Sahika Genc, Saurabh Gupta, Leo Dirac, Vineet Khare, Gourav Roy, Tao Sun, Yunzhe Tao, Brian Townsend, et al. Deepracer: Educational autonomous racing platform for experimentation with sim2real reinforcement learning. *arXiv preprint arXiv:1911.01562*, 2019.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

R Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. *GitHub repository*, 2016.

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

F1/10. F1/10 Autonomous Racing: Community, Course and Competition. http://f1tenth.org/, 2020. [Online; accessed 09-March-2020].

Andrew Fishberg, Trevor Henderson, Ken Gregson, Zachary Dodds, Don Krawciw, Christine Nicholls, and Sertac Karaman. Autonomous racecar: 1/10-scale autonomous car platform for research and education in robotics. In *2019 IEEE International Conference on Robotics and Automation (ICRA) Tutorial*. IEEE, 2019.

Brian Goldfain, Paul Drews, Changxi You, Matthew Barulic, Orlin Velev, Panagiotis Tsiotras, and James M Rehg. Autorally: An open platform for aggressive autonomous driving. *IEEE Control Systems Magazine*, 39(1):26–55, 2019.

Jon Gonzales, Ugo Rosolia, and Greg Marcil. Barc: Berkeley autonomous race car. URL http://www.barc-project.com/.

Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278. IEEE, 2016.

Pieter Hintjens. *ZeroMQ: messaging for many applications.* " O'Reilly Media, Inc.", 2013.

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 287–296. IEEE, 2018.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *arXiv preprint arXiv:2002.00444*, 2020.

Debarati Kundu. *Subjective and objective quality evaluation of synthetic and high dynamic range images.* PhD thesis, 2016.

Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

II Lewis, W Cannon, Mark Moll, and Lydia E Kavraki. How much do unstated problem constraints limit deep robotic reinforcement learning? *arXiv preprint arXiv:1909.09282*, 2019.

LG Electronics, Inc. Lgsvl simulator, 2019. URL https://www.lgsvlsimulator.com/.

Alexander Liniger and John Lygeros. A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology*, 2019.

Rahul Mangharam, Madhur Behl, Houssam Abbas, Marko Bertonga, and Matthew O'Kelly. 2018 Italian Grand Prix, 2018a.

Rahul Mangharam, Madhur Behl, Houssam Abbas, and Matthew O'Kelly. 2018 Portuguese Grand Prix, 2018b.

Rahul Mangharam, Madhur Behl, Houssam Abbas, Marko Bertonga, and Matthew O'Kelly. 2019 New York City Grand Prix, 2019a.

Rahul Mangharam, Madhur Behl, Houssam Abbas, and Matthew O'Kelly. 2019 Canadian Grand Prix, 2019b.

Matthew McNaughton. Parallel algorithms for real-time motion planning. 2011.

M. O'Kelly, H. Zheng, J. Auckley, A. Jain, K. Luong, and R. Mangharam. Technical Report: TunerCar: A Superoptimization Toolchain for Autonomous Racing. Technical Report UPenn-ESE-09-15, University of Pennsylvania, September 2019. https://repository.upenn.edu/mlab_papers/122/.

Matthew O'Kelly, Aman Sinha, Hongseok Namkoong, Russ Tedrake, and John C Duchi. Scalable end-to-end autonomous vehicle testing via rare-event simulation. In *Advances in Neural Information Processing Systems*, pages 9827–9838, 2018.

Weichao Qiu, Fangwei Zhong, Yi Zhang, Siyuan Qiao, Zihao Xiao, Tae Soo Kim, Yizhou Wang, and Alan Yuille. Unrealcv: Virtual worlds for computer vision. *ACM Multimedia Open Source Software Competition*, 2017.

Craig Quiter and M Ernst. Deepdrive, 2018. URL `https://deepdrive.voyage.auto/`.

W Roscoe. Donkey car: An opensource diy self driving platform for small scale cars, 2019.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

Volkan Sezer and Metin Gokasan. A novel obstacle avoidance algorithm:"follow the gap method". *Robotics and Autonomous Systems*, 60(9):1123–1134, 2012.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL `https://arxiv.org/abs/1705.05065`.

Aman Sinha, Matthew O'Kelly, Hongrui Zheng, Rahul Mangharam, John Duchi, and Russ Tedrake. Formulazero: Distributionally robust online adaptation via offline population synthesis. *arXiv preprint*, 2020.

Siddhartha S. Srinivasa, Patrick Lancaster, Johan Michalove, Matt Schmittle, Colin Summers, Matthew Rockett, Joshua R. Smith, Sanjiban Chouhury, Christoforos Mavrogiannis, and Fereshteh Sadeghi. MuSHR: A low-cost, open-source robotic racecar for education and research. *CoRR*, abs/1908.08031, 2019.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Jonathan Uesato, Ananya Kumar, Csaba Szepesvari, Tom Erez, Avraham Ruderman, Keith Anderson, Nicolas Heess, Pushmeet Kohli, et al. Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures. *arXiv preprint arXiv:1812.01647*, 2018.

Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

Benjamin Vedder. Vedder electronic speed controller, 2015. URL `https://vesc-project.com/documentation`.

Boling Yang, Patrick Lancaster, Siddhartha Srinivasa, and Joshua R Smith. Benchmarking robot manipulation with the rubik's cube. *IEEE Robotics and Automation Letters*, 2020.