The AI Driving Olympics at NIPS 2018

Andrea Censi,[1,2] Liam Paull,[3*] Jacopo Tani,[1] Julian Zilly,[1] Thomas Ackermann,[1] Oscar Beijbom,[2] Berabi Berkai,[1] Gianmarco Bernasconi,[1] Anne Kirsten Bowser, Simon Bing,[1] Pin-Wei David Chen,[4] Yu-Chen Chen,[4] Maxime Chevalier-Boisvert,[3] Breandan Considine,[2,3] Andrea Daniele,[9] Justin De Castri,[5] Maurilio Di Cicco,[2] Manfred Diaz,[3] Paul Aurel Diederichs,[1] Florian Golemo,[3,6] Ruslan Hristov,[2] Lily Hsu,[4] Yi-Wei Daniel Huang,[4] Chen-Hao Peter Hung,[4] Qing-Shan Jia,[7] Julien Kindle,[1] Dzenan Lapandic,[1] Cheng-Lung Lu,[1,4] Sunil Mallya,[5] Bhairav Mehta,[3] Aurel Neff,[1] Eryk Nice,[2] Yang-Hung Allen Ou,[4] Abdelhakim Qbaich,[3] Josefine Quack,[1] Claudio Ruch,[1] Adam Sigal,[3] Niklas Stolz,[1] Alejandro Unghia,[1] Ben Weber,[1] Sean Wilson,[8] Zi-Xiang Xia,[4] Timothius Victorio Yasin,[4] Nivethan Yogarajah,[1] Yoshua Bengio,[3] Tao Zhang,[7] Hsueh-Cheng Wang,[4] Matthew Walter,[9] Stefano Soatto,[5] Magnus Egerstedt,[8] Emilio Frazzoli[1,2]

Abstract

Deep learning and reinforcement learning have had dramatic recent breakthroughs. However, the ability to apply these approaches to control real physically embodied agents remains primitive compared to traditional robotics approaches.

To help bridge this gap, we are announcing the AI Driving Olympics (AI-DO), which will be a live competition at the Neural Information Processing Systems (NIPS) in Dec. 2018. The overall objective of the competition is to evaluate the state of the art of machine learning and artificial intelligence on a physically embodied platform. We are using the Duckietown [14] platform since it is a simple and well-specified environment that can be used for autonomous navigation.

The competition comprises five tasks of increasing complexity - from simple lane following to managing an autonomous fleet. For each task we will provide tools for competitors to use in the form of simulators, logs, low-cost access to robotic hardware and live ”Robotariums” where submissions will be evaluated automatically.

Keywords Robotics, safety-critical AI, self-driving cars, autonomous mobility on demand.

1 Introduction

Machine Learning (ML) methods have shown remarkable success for tasks in both disembodied dataset settings and virtual environments, yet still have to prove themselves in embodied robotic task settings. The requirements for real physical systems are much different: (a) the robot must make actions based on sensor data in realtime, (b) in many cases all computation must take place onboard the platform and resources may be limited, and (c) many physical systems are safety-critical which imposes a necessity for performance guarantees and a good understanding of uncertainty. All of these requirements present a stress on current ML methods that are built on Deep Learning (DL), are resource hungry and don't provide performance guarantees “out of the box”.

“Classical” robotics systems are built as a composition of blocks (perception, estimation, planning, control, etc.), and an understanding of the interplay between these different components is necessary for robots to be able to achieve complex and useful tasks. However, with ML methods it is unclear whether

[1] ETH Zürich, Zürich, Switzerland
[2] nuTonomy, an Aptiv company, Boston, USA
[3] Mila, Université de Montréal, Montréal, Canada
[4] National Chiao Tung University, Taiwan
[5] Amazon AWS, Seattle, USA
[6] INRIA Bordeaux, France
[7] Tsinghua, Beijing, China
[8] Georgia Tech, Atlanta, USA
[9] Toyota Technological Institute of Chicago, Chicago, USA
* Corresponding author: paulll@iro.umontreal.ca

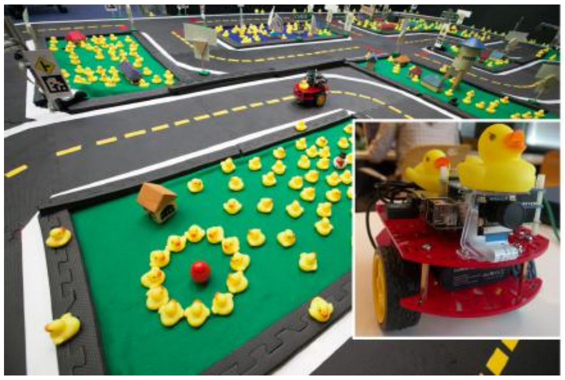


Figure 1: *Duckietown is a scaled-down autonomous city where the inhabitants are duckies. The platform comprises autonomous vehicles (Duckiebots) and a well-specified environment in which the Duckiebots operate. This platform is used as the basis for the AI Driving Olympics competition which will take place at the NIPS 2018 conference.*

these abstractions are necessary and whether it is more effective to learn chains of these components in an ”end-to-end” fashion or to separate the components out and learn them in isolation.

In order to address these issues, we have developed a new benchmark for evaluating ML on physically embodied systems, called the ”AI Driving Olympics” or AI-DO. The first AI-DO event will be a live competition at the Neural Information Processing Systems (NIPS) 2018 conference in Montréal, Québec.

A main consideration for casting this research as a competition is that solutions to embodied robotic tasks are notoriously difficult to compare [2, 4]. As argued by Behnke [4], competitions are therefore a suitable way of advancing robotics by making solutions comparable and results reproducible. Without a clear benchmark of performance, progress is difficult to measure. To drive home this message, we recall the story of the drunkard who has lost his house keys and then searches for them under the street light. When asked why he searches close to the street light he answers because *it is lighter there* [9]. Learning from this experience, we aspire to cast this competition in a way that encourages solving the problems that need to be solved rather than the ones we know how to solve.

Furthermore, as argued in [18] in the context of computer vision, large scale endeavors with relevant real-world applications have the potential to push the envelope of what is possible further in a process of *productive failure* [10]. *Productive failure* is a concept pioneered by Kapur which emphasizes the importance of productive struggle in the context of learning.

1.1 Novelty and related work

We call this competition the “AI Driving Olympics” (AI-DO) because there will be a set of different trials that correspond to progressively more sophisticated behaviors for cars. These vary in complexity, from the reactive task of lane following to more complex and “cognitive” behaviors outlined in section 3.

The competition will be live at the Neural Information Processing Systems (NIPS) conference, but participants will not need to be physically present—they will just need to send their source code.

There will be qualifying rounds in simulation, similar to the recent DARPA Robotics Challenge [8], and we will make the use of “robotariums,” [16], facilities that allow remote experimentation in a reproducible setting more closely described in section 2.

Many competitions exist in the robotics field. One example is the long-running annual Robocup [12], originally thought for robot soccer (wheeled, quadruped, and biped), and later extended to other tasks (search and rescue, home assistance, etc.). Other impactful competitions are the DARPA Challenges, such as the DARPA Grand Challenges [13] in 2007-8 that rekindled the field of self-driving cars, and the recent DARPA Robotics Challenge for humanoid robots [8].

In ML in general, and at NIPS in particular, there exist no competitions that involve physical robots. Yet, the interactive, embodied setting is thought to be an

Table 1: Competition and Benchmark Comparison

Competition	DARPA [13]	KITTI [6]	Robocup [12]	RL comp. [11]	HuroCup [3]	AI-DO
Accessibility	-	✓	✓	✓	✓	✓
Diverse metrics	✓	-	✓	-	✓	✓
Modularity	-	-	-	-	-	✓
Resource constraints	✓	-	✓	-	✓	✓
Simulation/Realism	✓	✓	✓	-	✓	✓
ML compatibility	-	✓	-	✓	-	✓
Embodiment	✓	-	✓	-	✓	✓
Teaching	-	-	-	-	-	✓

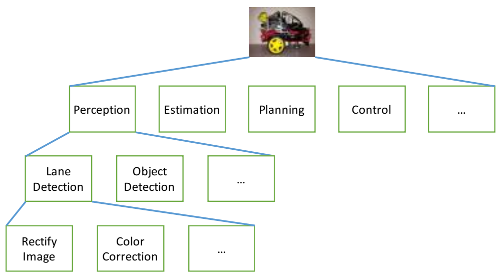


Figure 2: Modularity *- The full robotics problem is achieved through a hierarchical composition of blocks. Which are the right components and what is the right level of hierarchy to apply machine learning?*

essential scenario to study intelligence [15]. The application to robotics is often stated as a motivation in recent ML literature (e.g., [5, 7] among many others). However, the vast majority of these works only report on results *in simulation* [11] and on *very simple* (usually grid-like) environments.

To highlight what makes robotics and this competition unique, we list essential characteristics comparing existing competitions to the upcoming AI-DO as outlined in Tab. 1.

- **Accessibility:** No upfront costs other than the option of assembling a Duckiebot are required. Conceptually, classes teaching robotics and machine learning in the Duckietown setting will be made available online as described in section 5.
- **Resource constraints:** In robotics constraints on power, computation, memory and actuator constraints play a vital role.
- **Modularity:** More often than not, robotic pipelines can be decomposed into several modules (see Fig. 2).
- **Simulation/Realism:** Duckietowns will be made available both in simulation and reality posing interesting questions about the relationship to each other.
- **ML compatibility:** Additionally past data and ongoing data from cars in Duckietown will be made available to allow for training of ML algorithms.
- **Embodiment:** As any real robot, closed-loop and real-time interactive tasks await the participants of the competition.
- **Diverse metrics:** In most real-world settings, not one single number determines performance on a task. Similarly AI-DO employs multiple diverse performance metrics simultaneously.

2 The Platform

We make available a number of different resources to competitors to help them build, test, and finally evaluate their algorithms:

***Figure 3:** Currently 17 hours of logs are available from 38 different robots with more being continuously added.*

1. **The physical Duckietown platform [14, 17]** (Fig. 1): miniature vision-based vehicles and cities in which the vehicles drive. This is an affordable setup (∼\$200/robot), with rigorously defined hardware and environment specifications (e.g., the appearance specification, which enables repeatable experimentation in the online documentation [1]).

2. **Data** (Fig 3): Access to many hours of logs from previous use of the platform in diverse environments (a database that will grow as the competition unfolds).

3. **A cloud simulation and training environment**: for testing in simulation before trying on the physical fleet of robots.

4. **"Robotariums"**: remotely accessible, completely autonomous environments, continuously running the Duckietown platform. Robotariums will allow participants to see their code running in controlled and reproducible conditions, and obtain performance metrics on the tasks.

2.1 The Development Pipeline

Given the availability and relative affordability of the platform, we expect most participants to choose to build their own Duckietown and Duckiebots and develop their algorithms on the physical platform.

Alternatively, we provide a realistic cloud simulation environment. The cloud simulation also serves as a selection mechanism to access a robotarium.

The robotariums enable reproducible testing in controlled conditions. Moreover, they will produce the scores for the tasks by means of a network of street-level image sensors. The robotarium scores are the official scores for the leader board and they are used for the final selection of which code will be run at the live competition at NIPS. The participants will not need to be physically at NIPS — they can participate remotely by submitting a Docker container, which will be run for them following standardized procedures.

2.1.1 The physical Duckietown platform

The physical Duckietown platform comprises autonomous vehicles (*Duckiebots*) and a customizable model urban environment (*Duckietown*) [1].

The Robot Duckiebots are equipped with only one *sensor*: a front-viewing camera with 160° fish-eye lens, streaming 640×480 resolution images reliably at 30 fps.

Actuation is provided through two DC motors that independently drive the front wheels (differential drive configuration), while the rear end of the Duckiebot is equipped with a passive omnidirectional wheel. A minimum radius of curvature constraint is imposed, at software level, to simulate more car-like dynamics.

All the *computation* is done onboard on a Raspberry Pi 3 B+ computer, equipped with a quad-core 1.4 GHz, 64 bit CPU and 1 GB of RAM.

We might support other configurations for the purposes of deploying neural networks onto the robots.

Power is provided by a 10 Ah battery, which guarantees several hours of operation.

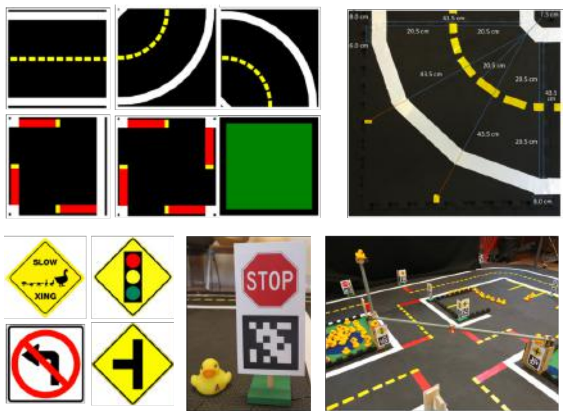


Figure 4: *The Duckietown environment is rigorously defined at road and signal level. When the appearance specifications are met, Duckiebots are guaranteed to navigate cities of any conforming topology.*

The Environment Duckietowns are modular, structured environments built on two layers: the *road* and the *signal* layers (Fig. 4).

There are six well defined *road segments*: straight, left and right 90 deg turns, 3-way intersection, 4-way intersection, and empty tile. Each is built on individual tiles, and their interlocking enables customization of city sizes and topographies. The appearance specifications detail the color and size of the lines as well as the geometry of the roads.

The signal layer comprises of street signs and traffic lights.

In the baseline implementation, *street signs* are AprilTags (in union with typical road sign symbols) that enable global localization and interpretation of intersection topologies by Duckiebots. The appearance specifications detail their size, height and positioning in the city. *Traffic lights* provide a centralized solution for intersection coordination, encoding signals in different LED blinking frequencies[1].

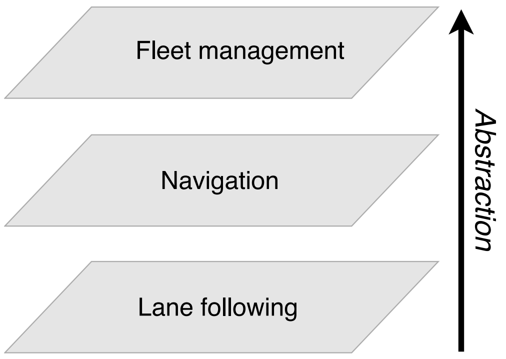


Figure 5: *The tasks are designed to be increasingly complex, and consequently investigate the autonomous mobility on demand problem at increasing levels of abstraction.*

3 Tasks

Many recent works in deep (reinforcement) learning cite robotics as a potential application domain [5, 7]. However, comparatively few actually demonstrate results on physical agents. This competition is an opportunity to properly benchmark the current state of the art of these methods as applied to a real robotics system.

Our experience thus far indicates that many of the inherent assumptions made in the ML community may not be valid on real-time physically embodied systems. Additionally, considerations related to resource consumption, latency, and system engineering are rarely considered in the ML domain but are crucially important for fielding real robots. We hope this competition can be used to benchmark the state of the art as it pertains to real physical systems and, in the process, spawn a more meaningful discussion about what is necessary to move the field forward.

The best possible outcome is that a larger proportion of the ML community redirects its efforts towards real physical agents acting in the real world, and helps to address the unique characteristics of the problem. The guaranteed impact is that we can establish a baseline for where the state of the art really is in this domain.

We propose a sequence of five challenges, in increasing order of difficulty. We briefly discuss the goal of each challenge; later, we will discuss the metrics in detail.

- **LF** **Lane following on a closed course**, with static obstacles (cones and parked vehicles). The robot will be placed on a conforming closed track (with no intersections) and should follow the track on the right-hand lane.
- **LFV** **Lane following** on a continuous closed course, with static obstacles **plus dynamic vehicles** controlled "by the matrix" sharing the road. Now the agent needs to deal with an environment populated by other intelligent agents.
- **NAVV** **Navigation plus vehicles** controlled by the matrix. Requires that the robot implement a coordination protocol for navigating the intersections.
- **FM** **Fleet management**: This scenario is meant to resemble a taxi-service. Customer request to go from location "A" to location "B" arrive sequentially and need to be served in an intelligent manner. Participants are asked to submit a central dispatcher that submits navigation tasks to Duckiebots to best serve customer requests.
- **AMOD** **Autonomous mobility on demand**: in addition to dynamic navigation, participants must implement a central dispatcher that provides goals to the individual robots. Points are given for the number of customers served.

3.1 Metrics and Judging

Each challenge will have specific objective metrics to quantify success. The success metrics will be based on measurable quantities such as speed, timing and automated detection of breaches of rules of the road. No human judges are necessary.

There are going to be three classes of metrics:

1. **Traffic laws compliance metrics**, to penalize the illegal behavior (e.g. not respecting the right of way);
2. **Performance metrics**, such as the average speed, to penalize inefficient solutions;
3. **Comfort metrics**, to penalize unnatural solutions that would not be comfortable to a passenger.

We will keep these metrics separate, and evaluate the solutions on a partial order, rather than creating a total order by collapsing the metrics in a single reward function to optimize. We will impose minimum objectives (minimum speed, maximum violation) to avoid degenerate solutions (e.g. a robot that does not move is very safe, but it does not have an acceptable performance). A more detailed mathematical description of these metrics is provided in the supplementary material (section 8).

We anticipate that there will be multiple winners in each competition (e.g. a very conservative solution, a more adventurous one, etc.).

4 Technical Components

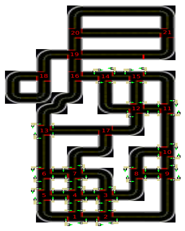
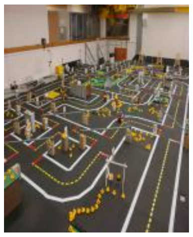
4.1 Localization

We are building infrastructure in our Robotariums system to automate the localization of the robots. This is essential for two reasons: 1) To evaluate the score of the agents, and 2) To robustly automate the process of resetting the agents at the their initial configurations in preparation for the evaluation of the next entry.

The approach for automatic localization is based on an system of cameras mounted on traffic lights and other elevated city structures, which are designed to locate the robots and report on their locations (Fig. 6).

Additional solutions might be used to increase overall robustness (e.g., ceiling mounted cameras).

Table 2: Overview of tasks in pictures. From left to right: Lane following, lane following with dynamic vehicles, navigation, fleet management and autonomous mobility on demand.

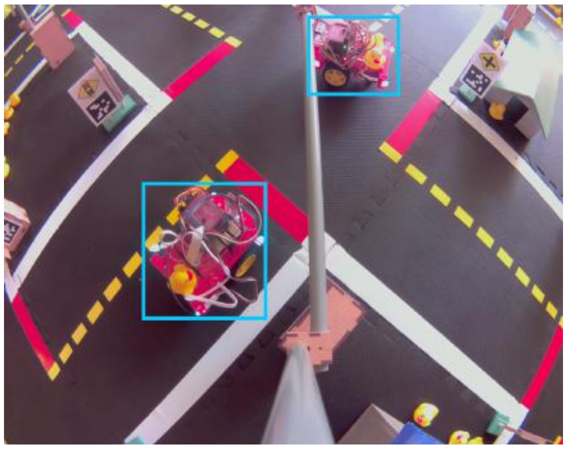


Figure 6: *The overhead cameras are used to provide ground truth localization of the robots inside the Robotarium environments.*

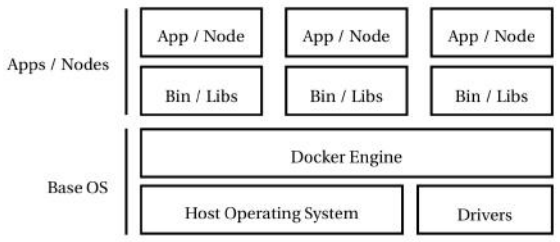


Figure 7: *Our Docker containerization system acts as an interface between the hardware/simulator and the competitors developed code.*

4.2 Containerization

In order to lower the barrier of entry for participants and minimize the amount of code refinements between platforms (simulators, robotarium, real robot), we are developing a state-of-the-art containerization infrastructure, which is also compatible with the cloud interface.

4.3 Simulation

We are developing a suite of simulators that are available to the competitors for use in development. These simulators scale in fidelity with the complexity of the tasks:

- we have a very lightweight ”machine-learning friendly” simulator with minimal dependencies and a low-level API based on OpenGL (shown in Fig. 8) that is used for lane following tasks;
- for navigation tasks, we provide a higher fidelity simulator based on Unity;
- for the AMOD task, we provide a fleet-level simulator.

4.4 Baselines

We are providing baseline “strawman” solutions, based both on “classical” and learning-based approaches. These solutions are fully functional (they will get a score if submitted) and contain all the necessary components, but should be easily beatable.

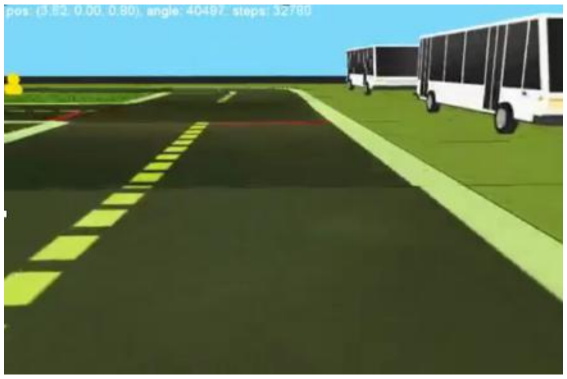


Figure 8: *Lightweight OpenGL based simulator used for lane following tasks.*

Additionally, the baseline solutions will be modular, i.e., they will contain a composition of containers, where competitors can choose to replace single containers or combinations (there need not be a one to one mapping between the containers in the baseline solutions and the competitor entries) as long as the well-specified APIs are preserved.

5 Pedagogical

It is envisioned that there will be two different groups of competitors: (a) independent learners not affiliated with any institution and (b) participants within the framework of a university class. A second version of the globally synchronized class will culminate this year with students submitting entries to the competition. We have provided a full suite of online learning materials in the ”Duckiebook” [1] to support both groups. These materials include slides, theory units, exercises, and demonstration instructions. For more details about the educational components of the projects please see [17].

6 Outreach

One of the key design criteria for this project and competition is the low barrier of entry, both in terms of effort and cost. This is very important since we want to encourage participation from all geographical and demographic categories.

To achieve this objective we have:

- designed the containerization framework to allow for rapid development;
- striven to keep the hardware components of the platform accessible (mostly off-the-shelf items) and low cost;
- provided access to cloud infrastructure, to ensure a level playing field for all participants.

7 Conclusion

We are excited to announce a new competition - the AI Driving Olympics, which will take place at the NIPS 2018 conference. The main objective is to evaluate the performance of state-of-the-art machine learning systems in the real physically embodied robot setting. The challenges inherent in deploying robots in the real world are quite different than most other applications where machine learning has had recent breakthroughs. We believe this is an opportunity to inspire the members of the ML community to focus more efforts on the physical embodied scenario, and to provide a benchmark for realistic comparison of algorithms. The platform is designed have a very low barrier for entry, both in terms of cost and in terms of effort, and we therefore hope to attract participants from diverse geographical regions and underrepresented demographics.

References

[1] The Duckietown Book (Duckiebook). docs.duckietown.org.

[2] John Anderson, Jacky Baltes, and Chi Tai Cheng. Robotics competitions as benchmarks for AI research. *The Knowledge Engineering Review*, 26(1):1117, 2011. doi: 10.1017/S0269888910000354.

[3] Jacky Baltes, Kuo-Yang Tu, Soroush Sadeghnejad, and John Anderson. HuroCup: competition for multi-event humanoid robot athletes. *The Knowledge Engineering Review*, 32, 2017.

[4] Sven Behnke. Robot competitions-ideal benchmarks for robotics research. In *Proc. of IROS-2006 Workshop on Benchmarks in Robotics Research*. Institute of Electrical and Electronics Engineers (IEEE), 2006.

[5] Devendra Singh Chaplot, Emilio Parisotto, and Ruslan Salakhutdinov. Active Neural Localization. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=ry6-G_66b`.

[6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[7] Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. {DARLA}: Improving Zero-Shot Transfer in Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 1480–1490, 2017.

[8] Karl Iagnemma and Jim Overholt. Introduction. *Journal of Field Robotics*, 32(2):187–188, 2015.

[9] Abraham Kaplan. *The conduct of inquiry: Methodology for behavioural science*. Routledge, 2017.

[10] Manu Kapur. Productive failure. *Cognition and instruction*, 26(3):379–424, 2008.

[11] Ł. Kidziński, S. P. Mohanty, C. Ong, J. L. Hicks, S. F. Carroll, S. Levine, M. Salathé, and S. L. Delp. Learning to Run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. *ArXiv e-prints*, 3 2018.

[12] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347. ACM, 1997.

[13] Keoni Mahelona, Jeffrey Glickman, Alexander Epstein, Zachary Brock, Michael Siripong, and K Moyer. Darpa grand challenge. 2007.

[14] Liam Paull, Jacopo Tani, Heejin Ahn, Javier Alonso-Mora, Luca Carlone, Michal Cap, Yu Fan Chen, Changhyun Choi, Jeff Dusek, Yajun Fang, and others. Duckietown: an open, inexpensive and flexible platform for autonomy education and research. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1497–1504. IEEE, 2017.

[15] Rolf Pfeifer and Christian Scheier. *Understanding intelligence*. MIT press, 2001.

[16] Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. The Robotarium: A remotely accessible swarm robotics research testbed. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1699–1706. IEEE, 2017.

[17] Jacopo Tani, Liam Paull, Maria T Zuber, Daniela Rus, Jonathan How, John Leonard, and Andrea Censi. Duckietown: an innovative way to teach autonomy. In *International Conference EduRobotics 2016*, pages 104–121. Springer, 2016.

[18] Julian Zilly, Amit Boyarski, Micael Carvalho, Amir Atapour Abarghouei, Konstantinos Amplianitis, Aleksandr Krasnov, Massimiliano Mancini, Hernn Gonzalez, Riccardo Spezialetti, Carlos Sampedro Pérez, and others. Inspiring Computer Vision System Solutions. *arXiv preprint arXiv:1707.07210*, 2017.

8 Appendix: Performance metrics

Measuring performance in robotics is less clear cut and more multidimensional than traditionally encountered in machine learning settings. Nonetheless, to achieve reliable performance estimates we assess submitted code on several *episodes* with different initial settings and compute statistics on the outcomes. We denote $\mathcal{J}$ to be an objective or cost function to optimize, which we evaluate for every experiment. In the following formalization, objectives are assumed to be minimized.

In the following we summarize the objectives used to quantify how well an embodied task is completed. We will produce scores in three different categories: *performance objective*, *traffic law objective* and *comfort objective*. Note that the these objectives are not merged into one single number.

8.1 Performance objective

Lane following (LF / LFV)

As a performance indicator for the "lane following task" and the "lane following task with other dynamic vehicles", we choose the speed $v(t)$ along the road (not perpendicular to it) over time of the Duckiebot. This then in turn measures the moved distance per episode, where we fix the time length of an episode. This encourages both faster driving as well as algorithms with lower latency. An *episode* is used to mean running the code from a particular initial configuration.

$$\mathcal{J}_{P-LF(V)}(t) = \int_0^t -v(t)dt$$

The integral of speed is defined over the traveled distance of an episode up to time $t = T_{eps}$, where T_{eps} is the length of an episode.

Navigation (NAVV)

Similarly, for the "navigation with dynamic vehicles task" (NAVV), we choose the time it takes to go from point A to point B within a Duckietown map as performance indicator. A trip from A to B is *active* as soon as it is received as long as it has not been completed.

This is formalized in the equation and integral below.

$$\mathcal{J}_{P-NAVV}(t) = \int_0^t \mathbb{I}_{AB-active}dt$$

The indicator function $\mathbb{I}_{AB-active}$ is 1 if a trip is *active* and 0 otherwise. Again the integral of an episode is defined up to time $t = T_{eps}$, where T_{eps} is the length of an episode.

Fleet management (FM)

As performance objective on task FM, we calculate the sum of trip times to go from A_i to B_i. This generalizes the objective from task NAVV to multiple trips. The difference to task NAVV is that now multiple trips (A_i, B_i) may be active at the same time. A trip is *active* as soon as it is requested and as long as it has not been completed. Likewise, multiple Duckiebots are now available to service the additional requests. To reliably evaluate the metric, multiple pairs of points A, B will be sampled at different time points within an episode.

$$\mathcal{J}_{P-FM}(t) = \sum_i \int_0^t \mathbb{I}_{i-active}dt$$

The indicator function $\mathbb{I}_{i-active}$ is 1 if a trip is *active* and 0 otherwise. Again the integral of an episode is defined up to time $t = T_{eps}$, where T_{eps} is the length of an episode.

Autonomous mobility on demand (AMoD)

An AMoD system needs to provide the highest possible service level in terms of journey times and wait times, while ensuring maximum fleet efficiency. We have two scoring metrics representing these goals in a simplified manner. In order to normalize their contributions, we supply a baseline case $\mathcal{B}$ for every scenario. In the baseline case, a simple heuristic principle is used to operate the AMoD system and the performance is recorded.

We introduce the following variables:

d_T	total distance driven by fleet
d_E	empty distance driven by fleet without a customer on board
$R \in \mathbb{N}^+$	number of requests in scenario
$w_i \in \mathbb{R}$	waiting time of request i
$w_{i,\mathcal{B}} \in \mathbb{R}$	w_i in $\mathcal{B}$ case
$N \in \mathbb{N}^+$	number of taxis
$N_{\mathcal{B}} \in \mathbb{N}^+$	N in $\mathcal{B}$ case

The first performance metric is for cases when the same number of vehicles as in the benchmark case $N_{\mathcal{B}}$ is used:

$$\mathcal{J}_{P-AMOD-1} = 0.5 \cdot \frac{d_E}{d_T} + 0.5 \cdot \frac{\sum_{i=1}^{K} w_i}{\sum_{i=1}^{K} w_{i,\mathcal{B}}}$$

The second performance metric allows the designer to reduce the number of vehicles, if possible, or increase it if deemed useful:

$$\mathcal{J}_{P-AMOD-2} = \mathcal{J}_{AMOD-1} + 0.5 \cdot \frac{N}{N_B}$$

For the AMoD task, only a performance metric will be evaluated. Robotic taxis are assumed to already observe the rules of the road as well as drive comfortably. Through the abstraction of the provided AMoD simulation, these conditions are already enforced.

8.2 Traffic law objective

The following are a list of rule objectives the Duckiebots are supposed to abide by within Duckietown. All individual rule violations will be summarized in one overall traffic law objective $\mathcal{J}_T$. These penalties hold for the lane following, navigation and fleet management tasks (LF, LFV, NAVV, FM).

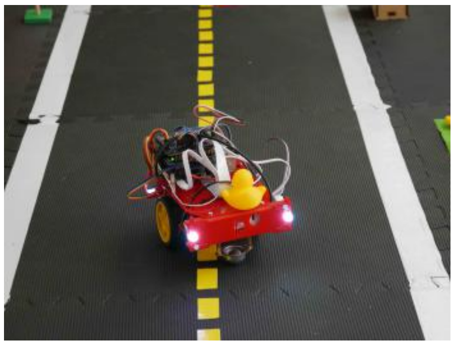


Figure 9: *Picture depicting a situation in which the* staying-in-the-lane rule *applies.*

Quantification of "Staying in the lane" The Duckietown traffic laws say:

"The vehicle must stay at all times in the right lane, and ideally near the center."

We quantify this as follows: let $d(t)$ be the absolute perpendicular distance of the center of mass the Duckiebot-body from the middle of the right lane, such that $d(t) = 0$ corresponds to the robot being in the center of the right lane at a given instant. While $d(t)$ stays within an acceptable range no cost is incurred. When the safety margin d_{safe} is violated, cost starts accumulating proportionally to the square of $d(t)$ up to an upper bound d_{max}. If even this bound is violated a lump penalty α is incurred.

The "stay-in-lane" cost function is therefore defined as:

$$\mathcal{J}_{T-LF}(t) = \int_0^{T_{eps}} \begin{cases} 0 & d(t) < d_{safe} \\ \beta d(t)^2 & d_{safe} \leq d(t) \leq d_{max} \\ \alpha & d(t) > d_{max} \end{cases}$$

An example situation where a Duckiebot does not stay in the lane is shown in Fig. 9.

Quantification of "Stopping at red intersection line" The Duckietown traffic laws say:

"Every time the vehicle arrives at an intersection with a red stop line, the vehicle should come to a complete stop in front of it, before continuing."

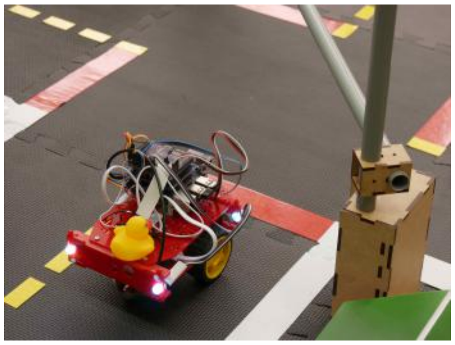


Figure 10: *Picture depicting a Duckiebot stopping at a red intersection line.*

During each intersection traversal, the vehicle is penalized by γ if there was not a time t when the vehicle was at rest ($v(t) = 0$) in the stopping zone defined as the rectangular area of the same width as the red line between 3 and 10 cm distance from the start of the stop line perpendicular to the center of mass point of the Duckiebot. This situation is demonstrated in Fig. 10. The condition that the position $p(t)$ of the center of mass of the Duckiebot is in the stopping zone is denoted with $p(t)_{\text{bot}} \in \mathcal{S}$. Then we write the objective as the cumulative sum of stopping at intersection rule infractions.

$$\mathcal{J}_{T-SI}(t) = \sum_{t_k} \gamma \mathbb{I}_{\nexists t \text{ s.t. } v(t)=0 \wedge p(t)_{bot} \in S_{zone}}$$

Here the sum over time increments t_k denote the time intervals in which this conditions is checked. The rule penalty is only applied once the Duckiebot leaves the stopping zone. Only then is it clear that it did not stop within the stopping zone.

To measure this cost, the velocities $v(t)$ are evaluated while the robot is in the stopping zone $\mathcal{S}$.

Quantification of "Keep safety distance" The Duckietown traffic laws say:

"Each Duckiebot should stay at an adequate distance from the Duckiebot in front of it, on the same lane, at all times."

We quantify this rule as follows: Let $b(t)$ denote the distance between the center of mass of the Duckiebot and the center of mass of the closest Duckiebot in front of it which is also in the same lane. Furthermore let b_{safe} denote a cut-off distance after which a Duckiebot is deemed "far away". Let δ denote a scalar positive weighting factor. Then

$$\mathcal{J}_{T-SD}(t) = \int_0^t \delta \cdot \max(0, b(t) - b_{\text{safe}})^2.$$

Quantification of "Avoiding collisions" The Duckietown traffic laws say:

At any time a Duckiebot shall not collide with another object or Duckiebot.

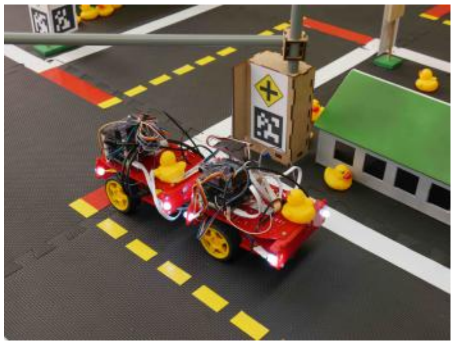


Figure 11: *Picture depicting a collision situation.*

The vehicle is penalized by ν if within a time a time interval of length t_k $t \in [t, t+t_k)$, the distance $\ell(t)$ between the vehicle and a nearby object or other vehicle is zero or near zero. $\ell(t)$ denotes the perpendicular distance between any object and the Duckiebot rectangular surface. The collision cost objective therefore is

$$\mathcal{J}_{T-AC}(t) = \sum_{t_k} \nu \mathbb{I}_{\exists t \in [t-t_k, t) \ell(t) < \epsilon}$$

Time intervals are chosen to allow for maneuvering after collisions without incurring further costs.

An illustration of a collision is displayed in Fig. 11.

Quantification of "Yielding the right of way" The Duckietown traffic laws say:

Every time a Duckiebot arrives at an intersection with a road joining on the right, it needs to check whether there are other Duckiebots on the right-hand lane of the joining road. If so, these vehicles shall traverse the intersection first.

Mathematically we accumulate penalties μ whenever the Duckiebot moves at an intersection while there is a Duckiebot (DB) on the right hand joining lane (RHL).

$$\mathcal{J}_{T-YR}(t) = \sum_{t_k} \mu \mathbb{I}_{v(t)>0 \wedge \exists \text{ DB in RHL}}$$

The yield situation at an intersection is depicted in Fig. 12.

Hierarchy of rules To account for the relative importance of rules, the factors $\alpha, \beta, \gamma, \delta, \nu, \mu$ of the introduced rules will be weighted relatively to each other.

Letting $>$ here denote "more important than", we define the following rule hierarchy:

$$\mathcal{J}_{T-AC} > \mathcal{J}_{T-SI} > \mathcal{J}_{T-YR} > \mathcal{J}_{T-SD} > \mathcal{J}_{T-LF}$$

I.e.:

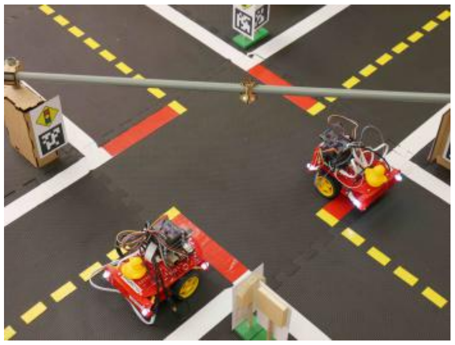


Figure 12: *Picture depicting a situation in which the* yield rule *applies.*

Collision avoidance $>$ Stop line $>$ Yielding $>$ Safety distance $>$ Staying in the lane.

This constrains the factors $\alpha, \beta, \gamma, \delta, \nu, \mu$ whose exact values will be determined empirically to enforce this relative importance.

While the infractions of individual rules will be reported, as a performance indicator all rule violations are merged into one overall traffic law objective $\mathcal{J}_T$. Let T denote a particular task, then the rule violation objective is the sum of all individual rule violations $\mathcal{J}_i$ which are an element of that particular task.

$$\mathcal{J}_T = \sum_i \mathbb{I}_{\mathcal{J}_i \in T} \mathcal{J}_{T-i},$$

where $\mathbb{I}_{\mathcal{J}_i \in T}$ is the indicator function that is 1 if a rule belongs to the task and 0 otherwise.

8.3 Comfort metric

Lane following and navigation (LF, LFV, NAVV)

In the single robot setting, we encourage "comfortable" driving solutions. We therefore penalize large accelerations to achieve smoother driving. This is quantified through smoothed changes in Duckiebot position $p_{bot}(t)$. Smoothing is performed by convolving the Duckiebot position $p_{bot}(t)$ with a smoothing filter k_{smooth}.

As a comfort metric, we measure the smoothed absolute changes in position $\Delta p_{bot}(t)$ over time.

$$\mathcal{J}_{C-LF/LFV/NAVV}(t) = \int_0^t k_{smooth} * \Delta p_{bot}(t)dt$$

Fleet management (FM)

In the fleet management setting ”customer experience” is influenced greatly by how fast and dependable a service is. If it is known that a taxi arrives quickly after ordering it, it makes the overall taxi service more convenient.

We therefore define the comfort metric as the maximal waiting time T_{wait} until customer pickup. Let T_{wait} denote the time beginning at the reception of a ride request until when the ride is started.

Let $S_{\text{wait}}(t) = \{T_{\text{wait}_1}, \dots\}$ denote the set of waiting times of all started ride requests $A_i \to B_i$ up to time t. Then the comfort metric of the fleet management task is the maximal waiting time stored in the set S_{wait}.

$$\mathcal{J}_{C-FM}(t) = \max_{T_{\text{wait}}} S_{\text{wait}}$$

This concludes the exposition of the rules of the AI Driving Olympics. Rules and their evaluation are subject to changes to ensure practicability and fairness of scoring.