Autonomous Driving and Undergraduates: an Affordable Setup for Teaching Robotics

N. Arnaldi, C. Barone, F. Fusco, F. Leofante and A. Tacchella

DIBRIS — Università degli Studi di Genova
Via all'Opera Pia, 13 — 16145 Genova — Italia
francesco.leofante@edu.unige.it – armando.tacchella@unige.it

Abstract. This work proposes an affordable setup for the application of machine learning to autonomous driving. We discuss the hardware details of the platform, and propose a scenario wherein learning from demonstration is leveraged to teach a vehicle how to drive. Preliminary results are discussed, together with potential targets enabled by the setup.

1 Introduction

Autonomous vehicles have been attracting the attention of researchers in a variety of engineering fields, and the first successful attempts at navigating cars autonomously can be traced back to more than thirty years ago — see, *e.g.*, [9, 14]. The interest in the topic is well motivated by the anticipated benefits of automated cars, such as the reduction in deaths from traffic accidents. Researchers in robotics have taken up this challenge focusing on different applications that range from low-level control of the vehicle [7, 5], to high level tasks such as safe navigation through city streets [11, 12, 2]. Prize competitions have been funded by major research organization, such as the DARPA Grand Challenge[1], created to promote the development of technologies for fully autonomous ground vehicles. Many companies have also joined the race to self-driving cars: Google X launched an ambitious project to develop driverless cars[2] [8], Tesla is already commercializing cars with autopilot [10], while Apple, Microsoft and Amazon allegedly made important investments in the same technology.

Given the current state of affairs, universities ought to introduce students to Artificial Intelligence (AI) themes that are useful in autonomous driving, as this addition enhances their curricula and helps them to keep up with the state of the art (see, *e.g.*, [4, 15, 13, 1] for similar initiatives). However, full-scale autonomous cars are a daunting challenge when it comes to teaching for many reasons, *e.g.*, limited budgets, demanding logistics, lack of qualified technical support. Mobile robots could be devoted to this purpose, but their kinematic models are often different from those of real cars. This abstract proposes an affordable setup to engage undergraduate students in AI techniques for autonomous driving, involving both hardware and software elements.

[1] `http://archive.darpa.mil/grandchallenge/`
[2] `https://www.google.com/selfdrivingcar/`

More specifically, our setup considers the problem of designing a controller for the NXP-Cup challenge[3], a competition which aims at introducing undergraduate students to embedded programming through the solution of a (simplified) autonomous driving problem for 1/18th scale battery-powered cars. Instead of using traditional control approaches, *i.e.*, manual synthesis and tuning of a controller, we propose to synthesize a controller using machine learning techniques. Our proposal is based on Learning from Demonstration (LfD) — see, *e.g.*, [3] for a survey. Accordingly, the vehicle learns how to drive using a set of reference trajectories provided by a human controlling the car along a track. The elements of the setup are affordable and the techniques involved are relatively easy to grasp, so that undergraduates can keep up with the main challenge of the NXP-Cup, which consists in having the model car drive faster than its competitors while staying on track. A companion site containing datasets, preliminary experimental results and software can be found at `www.aimslab.org/teaching`. The remainder of this abstract is organized as follows. Section 2 details the main features and requirements of the proposed benchmark, while Section 3 presents a preliminary evaluation. Finally, we discuss challenges and future directions in Section 4.

2 The setup

Context. The setup we propose is meant to compete in the NXP-Cup, an annual context organized by NXP where students are required to build and program a model car, fully equipped with a range of sensors, so that it can run autonomously on a given track. The implementation of the car requires different skills, including embedded software programming, control theory, and some basic electronics.

Hardware. The basic hardware required for the competition is provided by NXP, which also dictates the rules teams must adhere to. The basic kit includes a battery-powered model car, a line scan camera used for sensing and a microcontroller to control the car. Additional sensors can be employed, such as wheel encoders or cameras, up to the limits defined by the NXP rules. In our setup, the only additional sensor is a second line scan camera which enables a better reconstruction of the track unraveling in front of the vehicle. Given our proposal to teach the car how to drive using learning from demonstration, further components are required which will be then removed during the competition. In particular, we operate the car using a 2.4GHz hobby-grade remote controller coupled with a receiver. The signals from the controller are decoded to provide throttle and steering inputs, so that samples for our machine learning algorithm can be gathered under the form of reference trajectories. An Arduino board provides throttle and steering decoding, and stores demonstration data to be transferred

[3] Previously known as Freescale cup `https://community.nxp.com/docs/DOC-1284?tid=vanFREESCALECUP`.

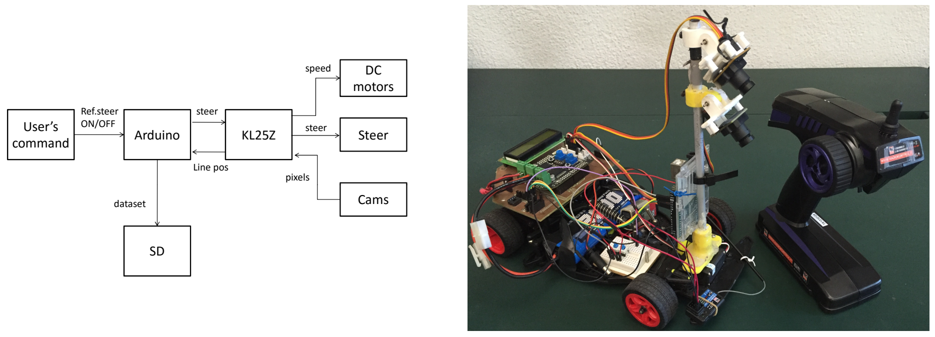

Fig. 1. Functional diagram of the setup (left) and a snapshot of the model car with the remote control for the teaching phase (right).

via Wi-Fi for further processing. In Figure 1 we show both a functional diagram of the components involved and a snapshot of the current prototype[4].

Control software. While most NXP-Cup competitors design and implement controllers manually, we propose to leverage state-of-the-art machine learning techniques. By doing so, we give students the chance to explore applications of AI solutions to a simple, yet challenging, autonomous driving problem. Students are required to formalize the learning problem as a LfD task and gather demonstration data by teleoperating the model car. After this initial phase, they have to choose the specific learning technique — *e.g.*, Neural Networks, SVRs, Gaussian Processes — and assemble the training dataset accordingly. This means they will have to reason and experiment about which sensory inputs are to be considered to obtain the best learning performances from a given learning algorithm. Ideally, students should be able to implement a simple state machine that includes the learned mapping from sensors to actuators in order to successfully drive the car along the track.

Simulator. A simulation of the model car running on a track has also been implemented in Matlab. Using the simulator, students can focus on dataset and algorithm design, avoiding hardware-related issues. A Simulink block containing the kinematic model of the car has been implemented. This allows the user to drive the car by specifying reference values for steering angle and speed. The simulator shows the car displacing on a user-defined track and also allows to change the geometric parameters of the line-scan camera, obtaining different field of views, and therefore different performances. The code of the simulator can be found in the companion site of this project. After data has been collected from the simulator the learning process can take place. In our preliminary implementation, demonstration data is fed to a Python script which implements Gaussian Process Regression using the GPy library [6] developed by the University of Sheffield. Once a solution for the problem is learned — *e.g.*, under

[4] The total cost of the platform as depicted in Figure 1 is approximately 350 euros.

the form of policy or mapping function — it can be tested in the simulator to evaluate its performances.

3 Preliminary Evaluation

Preliminary experiments on the setup shown in Figure 1 have been carried out to investigate the feasibility of our proposal when it comes to teaching undergraduate students. In the following experiments, the car is not equipped with wheel encoders — which are a common addition in NXP competitions — and data about the steering angles is read directly from the motors. Other than this, the configuration of the car is NXP-Cup-legal, so it reflects intended usage scenarios. An instance of the LfD problem is considered where the objective is to learn a function mapping from camera readings at time t to steering angles to be applied at time $t+1$. Demonstration data was collected using the on-board architecture of Figure 1, learning was done off-board. Trials were run on two different tracks: (*i.*) a loop with two straights and two 180-degrees turns, and (*ii.*) a more complex track, including crossings of different shapes. The first three authors, who are graduate students at the University of Genoa, tested the framework as if they were tackling the problem for the first time. As a result, they managed to have the car learn how to drive on the first track in a reliable way. However, the car could only move at a limited speed. The second track proved even more challenging: the car dealt badly with crossings, suggesting that improvements in sensory data and/or learning setup should be considered. While analyzing the dataset gathered during this experimental campaign we noticed that large parts of the input space where never explored during teaching. This was mostly due to sensors not being able to provide useful feedbacks for learning, *e.g.*, line-scan cameras giving blank readings most of the time as the borders of the track were hardly seen during teaching.

4 Conclusion

We believe that, once some aspects related to reliable sensory acquisition are solved in the current prototype, the proposed setup will provide an excellent testbed for teaching undergraduate students basic principles in AI and Robotics. The hardware setup is affordable and can be extended on a limited budget. Students can thus experiment easily how different sensors could affect the outcome of the learning process. At the same time, the setup is standardized as the car must comply to the NXP-Cup rules. Furthermore, working on this problem allows to deal with a complete robotics project, from low-level control to high-level decision-making. On the AI side, the problem of teaching a car how to drive is not trivial, even in this simplified setup. Moreover, students have to deal with limited hardware resources, *e.g.*, memory shortage, limited sensor feedbacks. On top of this, the controller obtained not only has to drive the car along the track, but it also has to be fast enough to compete with traditional controllers.

References

1. AAAI Spring Symposium on Educational Advances in Artificial Intelligence. `www.aaai.org/Symposia/EAAI/eaai-symposia.php`, accessed: 2016-11-02
2. Althoff, M., Stursberg, O., Buss, M.: Model-based probabilistic collision detection in autonomous driving. IEEE Transactions on Intelligent Transportation Systems 10(2), 299–310 (2009)
3. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and autonomous systems 57(5), 469–483 (2009)
4. Beux, S., Briola, D., Corradi, A., Delzanno, G., Ferrando, A., Frassetto, F., Guerrini, G., Mascardi, V., Oreggia, M., Pozzi, F., Solimando, A., Tacchella, A.: Computational thinking for beginners: A successful experience using prolog. In: Proceedings of the 30th Italian Conference on Computational Logic, Genova, Italy, July 1-3, 2015. pp. 31–45 (2015)
5. Falcone, P., Borrelli, F., Asgari, J., Tseng, H.E., Hrovat, D.: Predictive active steering control for autonomous vehicle systems. IEEE Transactions on control systems technology 15(3), 566–580 (2007)
6. GPy: GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy` (since 2012)
7. Gray, A., Gao, Y., Hedrick, J.K., Borrelli, F.: Robust predictive control for semi-autonomous vehicles with an uncertain driver model. In: Intelligent Vehicles Symposium (IV), 2013 IEEE. pp. 208–213. IEEE (2013)
8. Guizzo, E.: How googles self-driving car works. IEEE Spectrum Online, October 18 (2011)
9. Kanade, T., Thorpe, C., Whittaker, W.: Autonomous land vehicle project at cmu. In: Proceedings of the 1986 ACM fourteenth annual conference on Computer science. pp. 71–80. ACM (1986)
10. Kessler, A.M.: Elon musk says self-driving tesla cars will be in the us by summer. The New York Times p. B1 (2015)
11. Kümmerle, R., Ruhnke, M., Steder, B., Stachniss, C., Burgard, W.: A navigation system for robots operating in crowded urban environments. In: Robotics and Automation (ICRA), 2013 IEEE International Conference on. pp. 3225–3232. IEEE (2013)
12. Kümmerle, R., Ruhnke, M., Steder, B., Stachniss, C., Burgard, W.: Autonomous robot navigation in highly populated pedestrian zones. Journal of Field Robotics 32(4), 565–589 (2015)
13. Sintov, N., Kar, D., Nguyen, T.H., Fang, F., Hoffman, K., Lyet, A., Tambe, M.: From the lab to the classroom and beyond: Extending a game-based research platform for teaching AI to diverse audiences. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA. pp. 4107–4112 (2016)
14. Wallace, R., Stentz, A., Thorpe, C.E., Maravec, H., Whittaker, W., Kanade, T.: First results in robot road-following. In: IJCAI. pp. 1089–1095 (1985)
15. Wollowski, M., Selkowitz, R., Brown, L.E., Goel, A.K., Luger, G., Marshall, J., Neel, A., Neller, T.W., Norvig, P.: A survey of current practice and teaching of AI. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA. pp. 4119–4125 (2016)