An open framework for human-like autonomous driving using Inverse Reinforcement Learning

Dizan Vasquez*, Yufeng Yu† Suryansh Kumar‡ and Christian Laugier*
* Inria Rhne-Alpes, France
alejandro-dizan.vasquez-govea@inria.fr
christian.laugier@inria.fr
† Beijing University, China
yufeng.yu@cis.pku.edu.cn
‡ IIIT Hyderabad, India
suryansh.kumar@iiit.ac.in

***Abstract*—Research on autonomous car driving and advanced driving assistance systems has come to occupy a very significant place in robotics research. On the other hand, there are significant entry barriers (*eg* cost, legislation, logistics) that make it very difficult for small research groups and individual researchers to have access to a real autonomous vehicle for their experiments. This paper proposes to leverage an existing driving simulator (Torcs) by developing a ROS communication bridge for it. We use is as the basis for an experimental framework for the development and evaluation of Human-like autonomous driving based on Inverse Reinforce Learning (IRL). Based on an extensible and open architecture, this framework provides efficient GPU-based implementations of state-of the art IRL algorithms, as well as two challenging test environments and a set of evaluation metrics as a first step toward a benchmark.**

***Index Terms*—Autonomous Driving, Inverse Reinforcement Learning, Experimental Frameworks.**

I. Motivation, Problem Statement, Related Work

Research on autonomous car driving and advanced driving assistance systems has been advancing at an accelerated pace during the last ten years. A clear sign of the progress is the rate at which developed technologies are being transferred to the industry, leading many major car makers to promise some degree of autonomous driving functionalities by 2020.

On the other hand, to progress, research depends on field experiments which, in general, are considerably difficult to realise due to many factors, such as:

- *Cost*. A fully-equipped vehicle often costs several times the value of the vehicle itself.
- *Legislation and logistics*. Realising autonomous navigation experiments in public spaces is dangerous and, in many countries, forbidden by the law. This forces experiments to be realised in test grounds which are not available for most researchers.
- *Technical complexity*. An autonomous vehicle is a highly complex system requiring, in addition to the research group, a significant technical team for development, testing and maintenance.

These difficulties effectively constitute a major obstacle for small research teams and individual researchers willing to contribute to the field. This paper aims to lower this entry barrier by introducing a framework that leverages the realistic and open Torcs simulator [1], integrating it with ROS, the Robot Operating System [2], a popular open source robotic software framework comprising communication middleware, drivers and algorithms, among other functionalities.

Although a simulator like Torcs is not well suited to study all the problems faced by autonomous navigation (*eg* perception, localization) we believe that it may be prove invaluable in developing algorithms for motion planning, control, and motion prediction. Specifically, we have focused our framework in the study, development and evaluation of algorithms for human-like autonomous driving based on Inverse Reinforcement Learning (IRL).

IRL based approaches to autonomous driving make the hypothesis that, when they drive, human beings act like planners optimizing a complex cost function which balances a number of often contradictory desiderata (*eg* maintaining a desired speed while maximizing tire adherence, maximizing distance to obstacles while respecting driving-side priorities, etc.). The task of IRL is to learn, from human demonstrations, how people balance these trade-offs, thus estimating the cost function they optimize.

The development of a realistic simulation-based experimental platform for IRL approaches is further motivated by the fact that, although these techniques are very general, applications to autonomous driving have been proposed since the early papers on the topic [3]. However, most of the experiments in the literature have been conducted on unrealistic simulations dismissing kynematics, dynamics, slipping, etc[1]. We hope that this framework will help to improve this situation and constitute a solid step towards more realistic benchmarking for IRL-based algorithms in the context of autonomous driving applications.

II. Technical Approach

As illustrated by Fig. 1, our framework comprises four groups of hardware modules: two platforms (the simulator and

[1]To the best of our knowledge, the only exception is the work of Abbeel et al [4], which worked with a real platform on a car parking environment, where no other mobile objects were present.

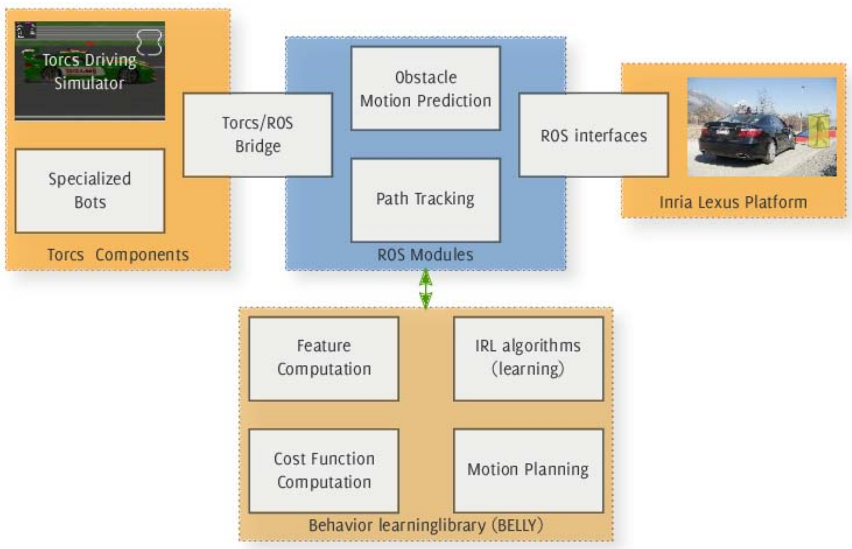

Fig. 1. Framework overview: *orange*) platforms; *brown*) IRL and planning libraries; *blue*) additional ROS modules. Modules described on this paper are indicated by grey boxes.

a real vehicle); a learning and planning library; and additional ROS modules for motion prediction and path tracking. The rest of this section describes the groups in more detail.

A. *Platforms*

The individual platforms are described in Sec. IV while here, we focus on their common elements. Both platforms connect through ROS to the motion prediction and tracking modules. We assume that every platform has the following built-in capabilities:

1) *Lane detection or localization and mapping* allowing the vehicle to situate itself and the other vehicles with respect to the road/track.
2) *Detection and tracking of mobile objects (DATMO)* to estimate the relative position and velocity of other cars around the ego-vehicle.
3) *Proprioceptive information* such as the car's odometry and engine state (*eg* revolutions per minute, current gear).
4) *Control*. For autonomous driving, the vehicle should expose interfaces to steering angle, acceleration, brake and gear changing. Currently, this is not implemented in our real vehicle.

B. *Behavior learning and planning*

The heart of our IRL learning and planning algorithms is a stand-alone library called *BEhavior Learning LibrarY* (Belly), which also takes care of motion planning –which is often used as a sub-routine by IRL algorithms. The library incorporates different learning algorithms, feature sets and planning algorithms using an extensible GPU-based plug-in architecture. As described in the following subsections, the Belly library is comprised of four modules: (a) feature computation; (b) cost-function computation; (c) motion planning algorithms; (d) the learning algorithms themselves.

1) Feature computation.: Most IRL approaches assume that the cost is a function of a set of task-specific features, representing the context associated with every possible system state. Currently implemented features include: (a) lateral displacement with respect to track center; (b) absolute speed; (c) relative speed with respect to traffic limitations; and (d) collision distance to an obstacle.

It is important to note that features are computed not only for the current time, but also for the future, based on the output of the prediction module. To the best of our knowledge, ours is the first IRL approach for autonomous driving to use such predicted features.

2) Cost-function computation.: From the obtained features, this module computes a discrete cost map, assigning a cost to every discrete state and time. In the same way as [5], [6], the only type of cost function currently supported by our framework is a linear combination of features where the relative weights have been estimated off-line by one of the implemented learning algorithms (see below).

3) Trajectory planning.: This module has pluggable algorithms which use the computed cost maps to find a trajectory which minimizes the incurred cost. The trajectory consists of a list of time-stamped waypoints to be used by the path-tracking algorithm. Currently, we have implemented the Dijkstra algorithm and a variation of the Forward-Backward algorithm proposed by Ziebart [7].

4) Learning: The goal of an IRL algorithm is to estimate the cost function parameters from exhibited behavior (*ie* a log file of human-driven trajectories). Currently, we have implemented two IRL algorithms: *max margin* IRL [3], [4] and *max entropy* IRL [7], the interested reader is referred to the respective papers for more information on them.

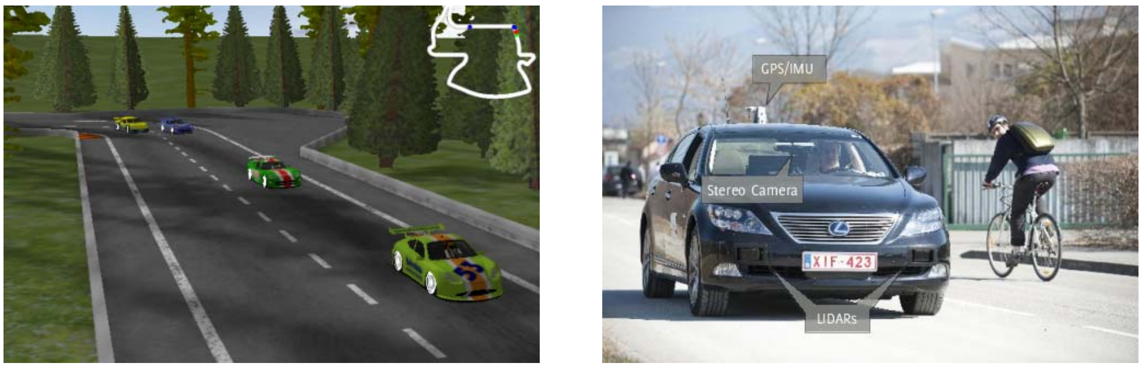

Fig. 2. Experimental platforms: *left*) Torcs racing simulator; *right*) sensor-equipped Lexus vehicle.

C. Other modules

This group includes two modules whose functionality is not related to IRL *per se*, namely motion prediction and path tracking.

1) Motion prediction.: The motion prediction module receives tracking data (*ie* position, orientation and velocity) and outputs H grids, representing the posterior probability of the space being occupied at times $\{t_1, \cdots, t_K\}$ in the future. At this point, prediction is accomplished through a standard Kalman Filter, which is then "rasterized" into the grids. This representation makes it easy to later introduce more advanced prediction algorithms.

2) Path tracking.: The output of the planning algorithm is a series of time-stamped waypoints containing nominal pose and velocity information for the ego-vehicle. The path tracking module is responsible of controlling the vehicle along those waypoints. Currently, it is implemented as two submodules:

1) A *velocity controller* that computes the required gear, acceleration and break commands required to keep the nominal velocity. Implemented as a Markov Decision Process.
2) A *pose controller* that computes the required steering angle in order to keep the nominal position and orientation. This is currently implemented as a pure pursuit controller.

III. Results

This paper presents an ongoing effort, and our results are preliminary. At this point, we have implemented all of the modules shown in Fig. 1, but we are still refining them and exploring alternative solutions. For instance, we are experimenting with different feature sets, so that the ones presented in Section II may be completed and fully compared on a further paper. Because of this, we do not have yet quantitative results to present here. We have, nonetheless, obtained a number of promising achievements:

- *ROS bridge*. The Torcs-ROS bridge is now very stable and suitable for its application, even outside the proposed IRL framework.
- *Navigation*.Our planning algorithm is able to produce smooth and safe plans at 2 Hz. In this sense, the integration of feature prediction as described in Sec. II-C, has made a big difference in both smoothness and reliability.
- *Generalization*. Our system has been able to learn different driving styles for both two-way roads and one-way racing tracks, based only on the use of different data sets. However, for a quantitative comparison with actual human driving, we need to finish implementing our evaluation metrics (see Sec. IV).

IV. Experiments

In this section we describe in further detail our two experimental platforms; our experimental test scenarios and methodology; and the quantitative metrics we intend to use to evaluate the performance of our algorithms.

A. Experimental platforms

Our IRL framework currently works with two experimental platforms, a realistic race simulator and Inria/Toyota instrumented Lexus car.

1) Simulator: We have developed a ROS bridge for the Torcs [1] simulator. Besides being open source and freely available, this racing simulator has a solid developer community and is often used in research. It features a rich physics engine which models collisions, tires (*eg* drifting), gear box, suspension (*eg* springs, dampers), aerodynamics and so on. This makes it a clear step ahead with respect to the simple simulations found in the IRL literature[8], [9]. Although originally intended as a racing simulation we have also developed "bots" to simulate a regular two-way road. Also, an additional bot has been developed to allow for human-driving logging to be used by the learning algorithms.

2) Real car: We are also conducting preliminary experiments with Inria/Toyota instrumented Lexus vehicle (Fig. 2). The results are limited, however, due to the lack of rear-facing sensors in the vehicle and the fact that the control interfaces are not yet available (we do log the computed commands). These provisional limitations have been a major motivation for us to start working in a simulated platform.

B. Test scenarios

Our first experiments aim to verify that the implemented algorithms are able to learn how to drive in two very different environments when using the exact same set of features. The first environment is a two-way road in which vehicles move at moderate speeds. The second one is a regular one-way racing track.

Experiments themselves simply consist of data gathering in both environments by having a human pilot to drive the car. The resulting data logs are fed into our learning algorithms and the resulting cost function parameters tested in autonomous mode.

C. Evaluation metrics

Since the overall goal of our experiments is to evaluate to which degree different features and IRL algorithms are able to replicate human behavior, we need to define appropriate evaluation metrics. For this, we have explored the literature, selecting pre-existing task-specific metrics, such as the average speed and the average time passed closely tailing or leading another vehicle [9]. We have also included some features of our own, such as velocity and steering angle smoothness, number of overtaken vehicles, etc.

V. Main experimental insights

Although preliminary, our experiments and the experience of putting together the experimental platform has already provided some important insights:

- *Motion prediction is crucial for autonomous driving*. Without feature prediction, the obtained motion was discontinuous and reactive, with the car often getting stuck in local minima. Even a nave motion prediction improved the situation considerably.
- *Even with a simulator, gathering training data is difficult*. An unexpected difficulty we have found is that the simulator is somewhat hard to drive for humans, requiring additional training in our test subjects.
- *Compared learning algorithms behave similarly, but features make a huge difference*. The obtained behavior did not perceptibly change within different IRL algorithms. Modifying the feature set, on the other hand, lead to widely different behavior on the vehicle.
- *Simulation is indeed a challenging research environment*. In our experiments, we have found that the simulator is already a very challenging environment. For instance, we have dedicated a lot of effort to develop a robust and reliable path tracker that avoids tire drifting while being responsive enough to follow up the high dynamics of the environment.

References

[1] B. Wymann, E. Espi, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "TORCS, The Open Racing Car Simulator," http://www.torcs.org, 2013.

[2] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.

[3] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. of the 21st Int. Conf. on Machine Learning*, Banff (USA), 2004.

[4] P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[5] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *Proc. of the Int. Conf. on Robotics and Automation*, Anchorage, USA, 2010.

[6] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," in *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[7] B. D. Ziebart, A. L. Maas, A. K. Dey, and A. J. Bagnell, "Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior," in *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008, pp. 322–331.

[8] S. Levine, Z. Popovic, and V. Koltun, "Feature construction for inverse reinforcement learning," in *Advances in Neural Information Processing Systems 23*, 2010, pp. 1342–1350.

[9] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," in *Proc. of the 29th Int. Conf. on Machine Learning (ICML)*, 2012.