Project plan (COMP4801)

AUTONOMOUS DRIFTING RC CAR WITH SIMULATION-AIDED REINFORCEMENT LEARNING

October 1, 2017

Supervisor: Dr. D. Schnieders
Sourav Bhattacharjee (3035123796)
Kanak Dipak Kabara (3035164221)
Rachit Jain (3035134721)

Contents

Abstract

Our project is aimed at studying autonomous drifting of an RC car. To do so, we use Reinforcement Learning (RL), that learns control policies from trial-and-error, much like how humans learn and solve various problems by interacting with the environment. However, learning agents that use RL typically require many interactions with the environment before learning any useful policy, which is something we cannot afford to do with a physical robot because it will lead to wear of the robot itself. So, instead we use a framework that uses policies learned from multiple simulations to initialize the policy in our RC car and in doing so, we make our RL learning agent as data-efficient as possible. The field of RL is rapidly evolving, and we try and use some of the cutting-edge technologies, like PILCO, that are addressed in the paper.

1 Background

Hydroplaning refers to a situation which occurs when a layer of water builds between car tires and the road. The tires of the car encounter more water than they can scatter, which results in the loss of traction. This can cause the car to slip, leading to loss of control and many accidents. According to the United States' Department of Transportation, wet and icy roads are responsible for 15.97% of all vehicle crash fatalities in the United States [1]. In this plan, drifting refers to the intentional oversteering of a vehicle where the rear wheels slide out and the its combined slide angle and slip angle is greater than the slip angle of the front wheels.

Passenger vehicles usually implement stability control in a number of ways like differential braking [2], active steering [3] [4] or integrated chassis control [5] [6] [7]. Other methods, based on independent wheel torque, have also been developed to make passenger vehicles more stable. However, all of these methods function by making sure that the tires avoid slipping and in doing so, essentially restricts the operation of the vehicle. Similarly, control algorithms in current self-driving car systems (ABS, ESC etc.) try and mitigate the chances of slipping due to its unpredictable nature [8]. Sufficiently lowering the speed of the car and making

turns that are not too tight will mostly prevent slipping, but this does not consider cases where the system must make evasive moves (in which case the speed and turn angle will most likely be sharp) or when a car is already in a slipping state due to the driver's fault. To ensure that these systems are as robust and safe as possible, it is paramount to study drifting, and eventually deduce how systems can respond quickly to unintentional slipping states as those encountered due to hydroplaning. In addition, avoiding accidents in such emergencies might involve taking full advantage of the capabilities of a vehicle, which is why we think it is important to study drifting.

Another reason behind our motivation to study drifting is that Paul Frère [9] pointed out the usefulness of drifting to turn fast around sharp bends. Since high-speed stability is of greater importance to ordinary touring vehicles and competition cars, they tend to understeer, and for a competition car average circuit time is improved by going fast through fast bends while slowing down through sharp ones [9]. However, a car is able to turn sharp bends faster by drifting because the yaw angle formed by the drift brings the vehicle in line with the straight path following the bend even before the vehicle completes the turn [9].

2 Objective and Scope

The objective of this project is to study a real world problem of high speed cars skidding when trying to turn during rain/wet roads. The area of drifting falls into two categories – sustained drift and transient drift. Due to the vast breadth of the two categories, our project will mainly focus on sustained drift, and more specifically steady state circular drift. We are only looking to handle the 'studying' aspect of drifting through this project - by teaching the car how to drift autonomously, we ensure that the car is able to understand the concept of drifting, and hence cope with the unpredictability that is inherent to the drifting state, as described above. Time permitting, we will also try to get the car to learn how to safely escape a drift state and return to a safe stop or straight-line motion. Furthermore, if time allows, we will try to experiment with getting the car to drift in a particular pattern like the pattern "8", for instance, to see how much more difficult it is compared to having the car in a state of steady circular drift.

3 Literature Review

3.1 Autonomous Sustained Drifting

Sustained drift has been achieved with various control techniques by researchers around the world. Velenis et al. [10] described a simple single-track vehicle model using equations of motion to design a 'sliding' control policy to stabilize steady state conditions using basic acceleration/braking applied to the wheels. Hindiyeh and Gerdes [11] developed an open-loop control policy using nested feedback loops to achieve stable drift equilibrium. They too developed a complex model of the vehicle, chassis and wheels to form the basis of their control policy. Wu and Yao [12] created a control algorithm to stabilize a RC drifting car by balancing the tail sliding with counter-steering measures to prevent slipping during circular motion. Their system is based on understanding the dynamics of the car, including the planar force and moment generated by the car's wheels during drifting. These modeled approaches work well in cases where the model encapsulates the various dynamics of the real-world, but do not work when the dynamics of the world are not understood completely to be modelled by equations of motion. The open-loop approach of the optimization is not implementable in the presence of uncertainties [13], and hence a better approach, one that is independent of the underlying models, is needed.

This is the perfect use case for learning-based methods, specifically Reinforcement Learning. Since RL learns policies by directly interacting with the environment, the policies are dependent on the real-world instead of a model.

3.2 Using Probabilistic Interference for Learning COntrol (PILCO)

Since we are using model based reinforcement learning, one of the biggest challenge is that it suffers from model biased. They assume that the learned dynamics models resembles the real environment [14]. One of the most common techniques while designing adaptive controllers is to add an extra term in the cost function of a minimum-variance controller, when accounting for uncertainty of the model

parameters. The model parameter estimation was improved by penalizing the uncertainty of the model parameters. The model bias was solved by explicitly modeling and averaging over model uncertainty [14]. In contrast, PILCO's approach is to treat the model uncertainty as temporally uncorrelated noise and it doesn't require sampling methods for planning.

There are multiple algorithms that use Gaussian Process models with Reinforcement Learning but as Deisenroth and Rasmussen pointed out, the shortcomings of these algorithms were that the dynamics models were learned by motor babbling [14]. But this approach was data inefficient and value function models had to be maintained which didn't scale well to high dimensions [14]. We use PILCO as it does not require an explicit value function model [14].

4 Methodology

Our project involves learning the optimal policy to drift a car autonomously using reinforcement learning (RL) methods. Traditional RL methods can be used to obtain the optimal policy by interacting with the environment and trying to maximize a reward function, but this process involves obtaining many samples from the physical robot before learning anything useful [4]. These samples are easy to obtain in some scenarios, like finding an optimal policy to play Atari games, but this is not preferred for robotic systems as it would lead to physical wear of the system. To overcome this problem, we use simulation-aided reinforcement learning [15] which minimizes the number of samples needed from the physical car.

Our method works by first using simple models to obtain an optimal policy using optimal control software like GPOPS. The simple models are deterministic and use closed-form, expressible equations of motion [15]. But, as pointed out by Cutler and How [15], these simple models often fail to model the true system, and the resultant policies are too tightly coupled with the underlying model. Thus, more complex simulations are needed.

To build on the policy learned from the simple models, the complex simulation is initialized with the simple policy. The simulator will be sufficiently stochastic to model the physical world, and hence, will help eliminate some of the problems from the simple model. The simulator will be created on Gazebo, an open-source simulation engine. Probabilistic Inference for Learning COntrol (PILCO), a recently developed model-based policy search RL algorithm that carefully handles the uncertainty of the learned model dynamics [16], is then used to obtain a better policy in the complex simulation. The reason we will be using PILCO is that it handles the uncertainity in the model and the model-based learning algorithm will not suffer from model bias [9].We will use Gaussian processes [17] as the regression tool to model the observed data and build a probabilistic model of transition dynamics. This model will be optimized over a long term to update the policy parameters and obtain a better policy. Once we have a policy learned in the complex simulation, it will be tested on the physical car. We will run PILCO in the

real physical system. The information gathered from the physical car will be fed back into the complex simulation. With real data from the physical world, we will reevaluate the previously learned policy in the simulation to move towards the optimal policy. This cycle of simulation and real world will be run until the learned policy is optimal in the simulation and the real world. Algorithm 1 below summarizes the methodology of our project.

Algorithm 1 Continuous State-Action Reinforcement Learning using Multi-Direction Information Transfer [15]

Input: Simple, deterministic simulation Σ_s, more complex, stochastic simulator Σ_c, and real world Σ_{rw}
Use optimal control methods to find policy π_s^* in Σ_s
Use k-means to approximate π_s^* from π_c^{init} as initial policy
while 1 **do**
    Run PILCO in Σ_{rw}
    RUN PILCO in Σ_c to get π_c^{init}, combining GP predictions from Σ_{rw}
    if $\|\pi_c^{new} - \pi_c\|$ **then**
        break
    else
        $\pi_c = \pi_c^{new}$
    end if
end while

5 Project schedule and milestones

5.1 Schedule

- **Week 1 - 4 (September 1 - 30, 2017)**
  - Research on project: Gaussian processes, PILCO, reinforcement learning, simulation-aided reinforcement learning, inverse reinforcement learning, RBF networks.
  - **Phase 1 deliverable on October 1, 2017**: Project plan.
- **Week 5 - 6 (October 1 - 15, 2017)**
  - **1st to 7th:** Figure out how to work with an Arduino.
  - **8th to 15th:** Add accelerometer and gyroscope to the car and learn to use them.
- **Week 7 - 9 (October 16 - 29, 2017)**
  - **16th to 22nd:** Get familiar with ROS and OpenAI Gym.
  - **23rd to 29th:** Integrate with Gazebo and learn to define robots and worlds.
- **Week 10 (October 30 - November 5, 2017)**
  - Create the simulator.
  - Get more familiar with PILCO.
- **Week 11 - 12 (November 6 - 19, 2017)**
  - Implement the RL algorithm.
- **Week 13 (November 20 - 26)**
  - Buffer for overruns in plan.
  - Start first iteration on the simulator by end of week to obtain initial simulated optimal policy.

- **Week 14 - 17 (November 27 - December 24, 2017)**
  - Preparation for exams
  - Transfer optimal policy learnt from simulator to physical car.
- **Week 18 - 19 (December 25, 2017 - January 7, 2018)**
  - Obtain data from the physical car and start another iteration on the simulator.
  - Buffer for overruns in plan.
  - Complete interim report.
- **Week 20 - 22 (January 8 - 21, 2018)**
  - Iterate on implementation of the RL algorithm.
  - Presentation on **January 12th, 2018**.
  - **Phase 2 deliverable on January 21st, 2018**.
- **Week 23 (January 22 - 28, 2018)**
  - Buffer for additional reading, if required.
- **Week 24 - 28 (January 29 - February 25, 2018)**
  - Further iterations of the learning process.
  - Try out the results on the physical RC car.
- **Week 29 - 31 (February 26 - March 11, 2018)**
  - Tests and finalize implementation.
  - Record finalized demo at the end of **week 31**.

- **Week 32 - 36 (March 12 - April 15, 2018)**
  - Work on final report.
  - Keep testing on RC car.
  - **Phase 3 deliverable on April 15th, 2018**.
  - Buffer for overruns in plan.
- **End of Week 37:** Presentation on **April 20th, 2018**.

5.2 List of deliverables

Deliverable	**Due**
Phase 1 deliverable: Project plan	October 1, 2017
RC car fitted with sensors	October 20, 2017
Simulator set up	November 5, 2017
RL algorithm preliminary implementation	November 30, 2017
Phase 2 deliverable: Interim report	January 21, 2018
Implementation on the RC car (preliminary demo)	February 15, 2018
Final demo of drifting RC car	March 11, 2018
Phase 3 deliverable: Implementation and final report	April 15, 2018

6 Materials required

Material	**Description**	**Cost (HK$)**
RC Car	A 4 wheel drive RC car with rear plastic wheels and front wheels with rubber tires. The rear plastic wheels are the main cause of drifting.	1000 - 2000
IMU	MPU 6050 sensor which contains an accelerometer and a gyroscope which will be used to maintain a reference direction of the car.	50 - 60
On Board Sensors	Sensors used to calculate the turn rate and wheel speed.	100 - 200
Motion Capture System	External sensors to capture the body frame velocities, and relative position of the car.	100 - 200
Voltage Regulator	The car uses a high current switching voltage regulator to keep the battery voltage from affecting the car's learning algorithm [15]	100 - 200
Arduino kit	The 'brain' of the RC car. Steering commands will be send to the Arduino from a PC, and the Arduino will then send control signals to the car.	800 - 850
Zigbee	A Zigbee module will be used for the communication between the Arduino and the program on our PC. The car's sensor data will be sent to the PC using the Zigbee module.	300 - 350
GPOPS	Optimal control software to learn the optimal control policy and initialize the policy in our simulator.	940
GPU	To train our model faster.	-

7 Separation of Responsibilities

We are three members in the group. After considering for a while, we have come to a consensus that each member will be equally responsible for each and every aspect of the project. Our motivation for doing so rather than setting out exclusive responsibilities is that this project is fairly complex and there are quite a few components to it, which are interrelated. It is thus imperative that each of us is equally involved with the different parts rather than making somebody entirely responsible for something. We feel we can work on the project more efficiently if each of us has an idea of the different parts. The exact parts of the code that each of us will implement will be figured out as the project progresses.

References

[1] S. Saha, P. Schramm, A. Nolan, and J. Hess, "Adverse weather conditions and fatal motor vehicle crashes in the united states, 1994-2012," *Environmental Health*, vol. 15, 2016.

[2] A. T. van Zanten, R. Erhardt, G. Landesfeind, and K. Pfaff, "Vehicle stabilization by the vehicle dynamics control system esp," *IFAC Mechatronic Systems, Darmstadt, Germany*, pp. 95–102, 2000.

[3] J. Ackermann, "Robust control prevents car skidding," *IEEE Control Systems Magazine*, vol. 17, pp. 23–31, 1997.

[4] K. Yoshimoto, H. Tanaka, and S. Kawakami, "Proposal of driver assistance system for recovering vehicle stability from unstable statesby automatic steering," in *Proceedings of the IEEE InternationalVehicle Electronics Conference*, 1999.

[5] A. Hac and M. Bodie, "Improvements in vehicle handling through integrated control of chassis systems," *International Journal of Vehicle Design*, vol. 29, no. 1, 2002.

[6] J. Wei, Y. Zhuoping, and Z. Lijun, "Integrated chassis control system for improving vehicle stability," in *Proceedings of the IEEE Inter- national Conference on Vehicular Electronics and Safety,*, 2006.

[7] A. Trachtler, "Integrated vehicle dynamics control using active brake steering and suspension systems," *International Journal of Vehicle Design*, vol. 36, no. 1, pp. 1–12, 2004.

[8] F. Zhang, J. Gonzales, K. Li, and F. Borrelli, "Autonomous drift cornering with mixed open-loop and closed-loop control," in *Proceedings IFAC World Congress*, 2017.

[9] P. Frère, *Sports Car and Competition Driving*. Bentley, 1969.

[10] E. Velenis, D. Katzourakis, E.Frazzoli, P.Tsiotras, and R.Happee, "Steady-state drifting stabilization of rwd vehicles," *Control Engineering Practice*, vol. 19, 2011.

[11] R. Hindiyeh and J. Gerdes, "A controller framework for autonomous drifting: Design, stability, and experimental validation," *Journal of Dynamic Systems, Measurement, and Control*, vol. 136, 2014.

[12] S.-T. Wu and W.-S. Yao, "Design of a drift assist control system applied to remote control car," *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, vol. 10(8), 2016.

[13] E. Velenis, E. Frazzoli, and P. Tsiotras, "On steady-state cornering equilibria for wheeled vehicles with drift," *Institute of Electrical and Electronics Engineers*, 2009.

[14] M. P. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," 2011.

[15] M. Cutler and J. P.How, "Autonomous drifting using simulation-aided reinforcement learning," *2016 IEEE International Conference on Robotics and Automation(ICRA)*, 2016.

[16] M.Deisenroth, D.Fox, and C. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 99, 2014.

[17] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.