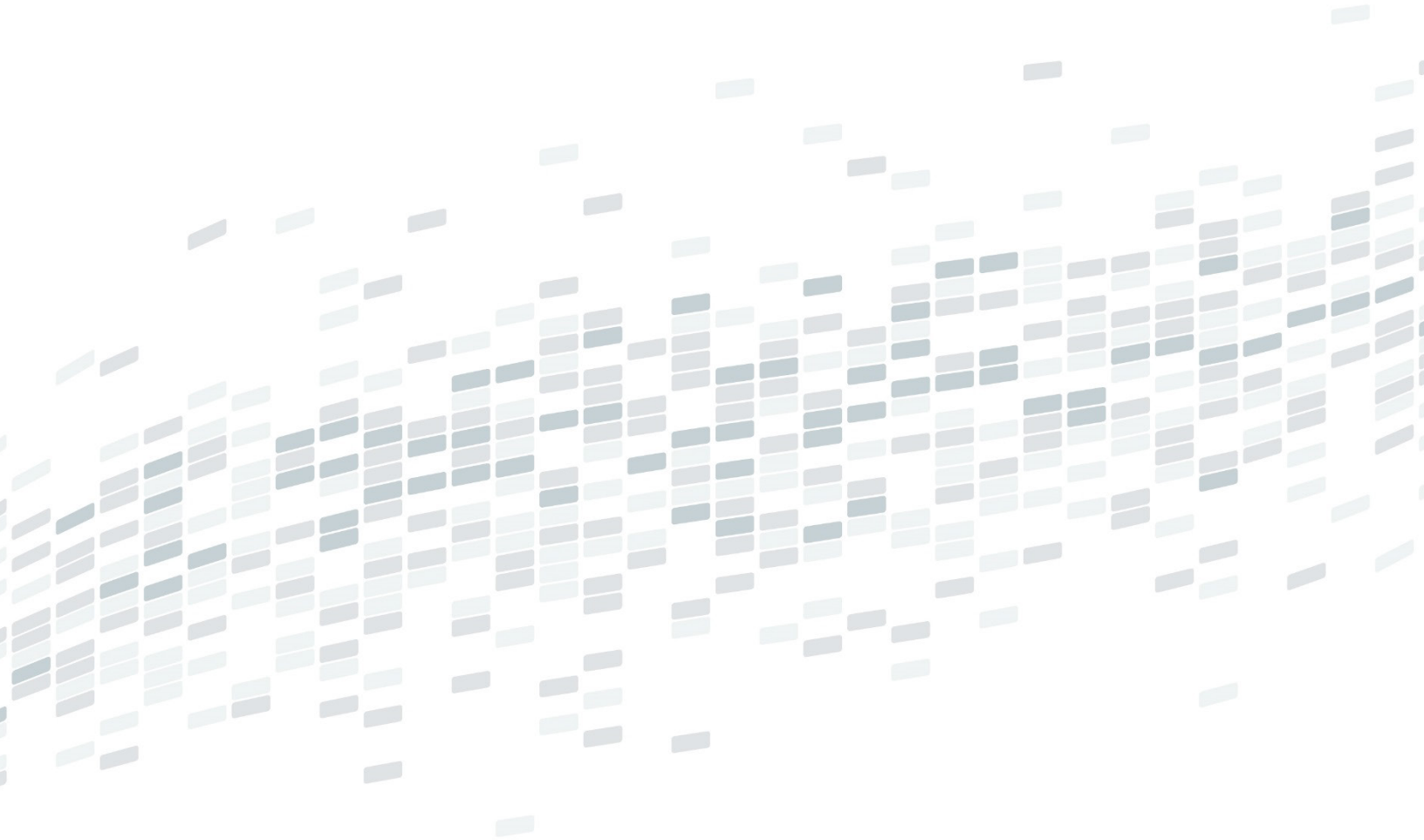
WHITE PAPER
www.visteon.com

Visteon®

A Generalized Method for Adaptive Longitudinal Control Using Reinforcement Learning



A generalised method for adaptive longitudinal control using reinforcement learning

Shashank Pathak[1] and Suvam Bag[2] Vijay Nadkarni[2]

[1] Visteon Electronics GmbH, An der RaumFabrik 33b 76227 Karlsruhe, Germany,
[2] Visteon Corporation, 2901 Tasman Drive, Santa Clara, CA, USA 95054
{name.surname}@visteon.com

Abstract. Adaptive cruise control (ACC) seeks intelligent and adaptive methods for longitudinal control of the cars. Since more than a decade, high-end cars have been equipped with ACC typically through carefully designed model-based controllers. Unlike the traditional ACC, we propose a reinforcement learning based approach – `RL-ACC`. We present the `RL-ACC` and its experimental results from the automotive-grade car simulators. Thus, we obtain a controller which requires minimal domain knowledge, is intuitive in its design, can accommodate uncertainties, can mimic human-like behaviour and may enable human-trust in the automated system. All these aspects are crucial for a fully autonomous car and we believe reinforcement learning based ACC is a step towards that direction.

1 Introduction

Recent advancements in machine learning has fuelled numerous applications – both in the academia and in the industries. In particular, supervised learning problems such image classification and object detection have matured to a product-level performance. On the downside, these approaches are limited by the requirements of well specified labelled data, which are either too expensive or too complex to obtain. Reinforcement learning[19], unlike supervised learning, is not limited to classification or regression problems, but can be applied to any learning problem under uncertainty and lack of knowledge of the dynamics. The approach indeed has been applied to numerous such cases where the environment model is unknown e.g - humanoids[18], in games[14], in financial markets[15] and many others.

Adaptive Cruise Control (ACC) is an important feature of autonomous driving where the car is designed to cruise at a speed both efficiently and safely over a wide range of scenarios and contexts. Numerous approaches for ACC have been proposed over the years. They can be divided in two major classes - one which primarily considers the physical signals of the environment and the other which considers the human perspective as the main design principle. We shall call them *environment-centric* and *driver-centric* ACC models.

Environment-centric models: One of the first environment-centric models proposed was Gazis-Herman-Rothery (GHR) model[7]. This was a linear-

proportional control where the desired acceleration was some factor of the velocity difference between the ego car and the car ahead. Many variations of this simple model were suggested later on to address the limitation such as asymmetry between acceleration and deceleration, the non-zero reaction time of the driver, and interaction with multiple vehicles. We refer the reader to [6] for a thorough discussion. In order to address the fact that each driver may have a notion of safe-distance (that might vary with the speed of the ego car), a different class of models was proposed such as [10]. The most well-known model of this category is the Intelligent-driver model (`IDM`)[20]. Another approach in this class of models is where the driver reacts to the distance between the cars rather than the relative speed, such as in [8]. In yet another approach, an assumption of optimal safe velocity is made such as in [4]. Each of these three sub-classes have seen various – some even fairly complex –enhancements being proposed. Consequently, there have also been models that prioritised simplicity, such as [17] that identified that ACC should only consider two *modes* of driving; one in congestion and the other when the ego car is free of any obstacles ahead. A related idea is to consider ACC behaviour as an automaton where the switching happens at desired contexts; this stream of approaches can be traced back to the seminal work of Nagel and Schreckenberg. [16]

Driver-centric models: One of the biggest flaws with the environment-centric models is the pre-assumption that the driver is perfectly rational, time-invariant, and can perceive all signals without failure and always seeks to optimise the control. None of these is actually true in the real-world. See [5] for the detailed criticism. Thus, there are classes of ACC design where the driver's perspective is central such as when a threshold to sensory perception is assumed[21] or the size of the objects[13] or visual angles[9].

Contributions : In this work, we propose a general approach to ACC which is significantly different from these two classes of ACC design. We identify ACC design as a highly context-dependent control problem. We incorporate changes due to both the environment and the driving styles. The key idea here is that instead of aiming for an optimal control for ACC, we strive for a human-like cruise driving that could adapt over time. Incorporating such sources of uncertainty increases the model complexity, hence in order to address that, we harness model-free approaches of the artificial intelligence research where exact modelling of the environment is skirted through efficient use of data. Additionally, the approach allows for truly adaptive cruise control that adapts not just to the environment but also the changing preferences of the driver over time.

2 Problem Formulation

The longitudinal control of an autonomous car has to cope with two major conflicting goals, viz., to maintain as close to the set speed as possible and to have as much safe distance from the preceding car as possible. There are other albeit less significant objectives besides this such as maintaining an optimally 'smooth'

trajectory. Note that in actual deployment of ACC in the car, these additional objectives are very crucial too; ACC is after all a comfort feature. Also, in a real-world scenario, longitudinal control of the car is seldom, if ever, de-coupled from other aspects such as lateral control, traffic situations and behaviour of the other cars. However, these factors when considered together make the problem insurmountable.

Let us denote the ego car with $\mathcal{E}$ while the one ahead of the ego car as $\mathcal{A}$. Also, the position of the car is denoted by x whereas velocity and acceleration by $\dot{x}$ and $\ddot{x}$ respectively. Furthermore, separation between these cars is denoted by Δx i.e., $\Delta x \coloneqq x_{\mathcal{A}} - x_{\mathcal{E}}$. The ego car would typically be controlled by some action; let $u_{\mathcal{E}}$ be such a control action. For example, in the simplest case, this control action could be the new position of the car. Since in the real world, longitudinal control is done through acceleration (gas pedal) and deceleration (brakes), we would assume the control action to be acceleration i.e., $u_{\mathcal{E}} \in [-b_{max}, a_{max}]_{\mathbb{Q}}$ where b_{max} is the maximum possible deceleration and a_{max} is the maximum acceleration that could be achieved given the physics of the car. There are couple of things to note here. First, the precision of actual control of acceleration (or deceleration) is governed by the dynamics of the car; we make this fact explicit through defining the control action range over rational numbers i.e., $u_{\mathcal{E}} \in \mathbb{Q}$. Secondly, the comfortable acceleration and deceleration limits are usually a strict subset of this range. We assume that the function $f^{\theta}(\Delta x) : \mathbb{R} \mapsto \mathbb{R}$ denotes the objective of maintaining a safe distance. Usually such a function would be parametrised by requirements like what is considered to be a safe distance; these parameters are denoted by θ. Similarly, we let the function $g^{\theta}(\dot{x}_{\mathcal{E}}) : \mathbb{R} \mapsto \mathbb{R}$ stand for the objective of maintaining the set speed. Typically, a given situation or driver preference would determine such a set speed; this is subsumed in the parameters θ. Finally, we assume that the dynamics of the environment is a black-box function $D(t)$ which at any instant $t > 0$ provides all necessary kinematics values i.e., $x_{\mathcal{E}}$, $\dot{x}_{\mathcal{E}}$, $\ddot{x}_{\mathcal{E}}$, $x_{\mathcal{A}}$, $\dot{x}_{\mathcal{A}}$ and $\ddot{x}_{\mathcal{A}}$; we shall denote it by $\mathbf{X}$. Equipped with these symbols, we are ready to define precisely what kind of control problem ACC is.

Definition 1 *Adaptive Cruise Control: Given the positions of the cars and possibly their time derivatives,* `classical-ACC` *is defined as a control problem of finding the optimal acceleration action that maintains a desired safe distance and also obeys a set speed. Mathematically,* `classical-ACC` *is equivalent to this problem:*

$$\begin{aligned} \underset{u_{\mathcal{E}}}{\textit{minimize}} \quad & f^{\theta}(\Delta x) + \lambda g^{\theta}(\dot{x}_{\mathcal{E}}) \\ \textit{subject to} \quad & u_{\mathcal{E}} \in [-b_{max}, a_{max}] \\ & \forall t > 0,\ \mathbf{X}_t = D(t) \end{aligned} \tag{1}$$

Here, $\lambda \in \mathbb{R}$ is a factor to facilitate convex combination of the conflicting objectives of maintaining a set-speed and a safe distance. In general, such a combination could also be some non-linear function; we however consider only linear combination for sake of simplicity. Note that `classical-ACC` does not

consider the driver preferences explicitly. In fact, as is known already (e.g., see [5]), a crucial limitation of such models is that the parameters do not have intuitive meaning when actual traffic data are considered. In other words, the parameters of these models can not be set through observing actual traffic data and human behaviour.

3 Approach

3.1 Generalised approach to longitudinal control

Like the approaches mentioned in the previous section, we consider the longitudinal control as a problem separate from other factors such as lateral control, traffic conditions and behaviours of the other drivers. The objective to be minimised in 1 is therefore a function of only longitudinal variables.

In the context of this work, the control problem involves controlling the ego car in the longitudinal direction when the position of the car ahead is known through some sensory input. Unlike the model-based approach, we do not consider the physical model of the problem and this position may even be provided in non-metric terms such as through viewing angle or the image itself or a laser scan. In reality, we would use the radar measurements as the inputs. To make this point explicit, we consider that the input variables are represented by the vector $\mathbf{Z}$. In the simple scenario, this would be all kinematics variables provided through ideal sensors i.e., $\mathbf{Z} = \mathbf{X}$. As is common in the learning community, we shall call the control strategy as *policy* which maps a given observation to the desirable action under that behaviour i.e., $\pi(\mathbf{Z}) = u_{\mathcal{E}}$.

Another aspect that we consider in our work is regarding intuitive behaviour model of the controller. This is similar to those car following models (see the previous section 2) which consider human aspects of driving. However, unlike those approaches, we seek to model all the human-like preferences and biases under a single framework of *reward schema*. For example, a conservative driver would put a higher negative reward for an unsafe close distance than a usual driver. Similarly, in a highway of high-speed cars, faster speeds would carry bigger rewards.

As has been noted recently, longitudinal (and lateral) control of the car by human drivers are typically in the order of few seconds. E.g., once a driver commits to accelerating towards the car ahead, he is not likely to change it within the next few milliseconds. Considering the abstract-level of human-like control of the car greatly signifies the learning problem. This is similar to hybrid control strategy where the high-level controller is responsible for discrete jumps between low-level continuous state controllers. Similar to such approaches, we assume that such a low-level controller is provided to us. In the industrial setting, such controllers are designed based on specifics of the car and its numerous physical parameters.

In order to obtain a generalised longitudinal controller, in this work we have applied an artificial intelligence approach to the controller design. In particular,

we use reinforcement learning to learn the controller through various active simulations. On one hand, this allows us to harness the reward schema mentioned before while on the other hand, it is a general approach which degenerates into known model-based controllers under suitable conditions. We shall now delve briefly into the learning approach.

To summarise, our approach is to generalise the `classical-ACC` problem into one where control is learned through the interaction with the world. To be more precise, we re-formulate the problem as:

Definition 2 *Learned Adaptive Cruise Control: Given the observations of the world* $\mathbf{Z}$*,* `RL-ACC` *is defined as a learning problem of finding the near-optimal policy* π *(of acceleration control) under a given set of rewards specified for different scenes and actions i.e.,* $R^{\pi}(\mathbf{Z}_0)$*; here* $\mathbf{Z}_0 \in \mathcal{Z}$ *is the initial state at time* $t = 0$ *where after the policy* π *is followed.* $\mathcal{Z}$ *is set of all possible initial states. Mathematically,* `RL-ACC` *is equivalent to this problem:*

$$\begin{aligned} \underset{u_{\mathcal{E}}}{\textit{maximise}} \quad & \mathop{\mathbb{E}}_{\mathbf{Z}_0 \in \mathcal{Z}} \{R^{\pi}(\mathbf{Z}_0)\} \\ \textit{subject to} \quad & u_{\mathcal{E}} \in [-b_{max}, a_{max}] \\ & \forall t > 0,\ \mathbf{Z}_t = \hat{D}(t) \end{aligned} \tag{2}$$

Note that this approach does not require to set the functions $f^{\theta}(\Delta x)$ and $g^{\theta}(\dot{x}_{\mathcal{E}})$ explicitly. In the situations where these functions are known, such as from the previous models, we can readily incorporate them in our rewarding schema.

3.2 Least Square Policy Iteration

Markovian Decision Problem (MDP) involves a probabilistic system represented by the tuple $(\mathcal{S}, \mathcal{U}, \mathcal{T}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ is the countable set of states of the system while $\mathcal{U}$ is the action space. The uncertain nature of the environment is encoded in the transition system $\mathcal{T} : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}_{[0,1]}$. The rewards in each state (and action) are determined by $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$ and $\gamma \in \mathbb{R}_{[0,1)}$ is the discounting factor. MDP is an NP-hard problem[12] of finding an optimal policy $\pi^* : \mathcal{S} \mapsto \mathcal{U}$ which maximises the total discounted reward. One of the approaches to achieve this approximately is to define *Q-value* as:

$$Q^{\pi}(\hat{s}, \hat{u}) = \mathop{\mathbb{E}}_{u_t \sim \pi, s_t \sim \mathcal{T}} (\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = \hat{s}, u_o = \hat{u})$$

By applying the Bellman optimality criteria – i.e., Q-value at any step t with the state s and the action u returned by an optimal policy is same as the optimal reward obtained at this step and following the optimal policy for all subsequent steps ($t+1$ onward) – we can compute the Q-value under the current policy. This is *policy evaluation*. In order to improve a policy, we may simply select those actions that maximises the Q-value thus computed. This step is *policy improvement*. When performed iteratively over the space of deterministic

policies, policy evaluation and policy improvement result in optimal policy. This is the essence of *policy iteration*. However, under this algorithm, convergence is guaranteed only when the representation of the Q-values as well as the policy is exact such as with tabular form. For large (or infinite) state space, both these representations need to be parametrised and hence only approximate.

Least Square Policy Iteration – or LSPI – is a class of approximate policy iteration reinforcement learning algorithms[11]. The crucial idea here is to project the actual policy to an approximate space such that the iterative operator would still yield Q-values that are near-optimal (w.r.t. actual fixed point) in the sense of L_2-norm. We refer the interested reader to the seminal paper work on LSPI[11].

3.3 Learning the ACC through LSPI

We are now ready with all ingredients to describe our approach for learning the ACC. We first generate the data of the interaction of the car with the environment. This is done through the third-party simulators described in 4. Once the required data of interaction is available, we apply the RL algorithm as described in Alg. 1. Here, we devise rewarding schema based on the control and the environment. Another domain specific aspect is the choice of gaussians. Typically, we would like the gaussians to approximate the actual Q-value as close as possible hence it is always better to choose the centres of these gaussians to be at the interesting parts of the state-space. Based on the number of these gaussians, we construct a random weight vector. Note that once the problem design is accomplished, the purpose of the `RL-ACC` is to obtain optimum weight vector. Rest of the algorithm is simply the LSPI.

Algorithm 1 `RL-ACC`

Require: reliable simulator `sim` for the environment, rewarding schema `reward`, ϕ set of gaussians, discounting factor γ

Construct sample set $D \Leftarrow$ (`sim`,`reward`)
Initialise ϕ and weights w randomly
$A = 0$, $b = 0$
while no convergence **do**
  for $(z, u, r, z') \in D$ **do**
    $A = A + \phi(z, u)(\phi(z, u) - \gamma\phi(z', \pi(u')))$
    $b = b + \phi(z, u)r$
  end for
  $w = A^{-1}b$
end while

3.4 Cruise Control mode

A cruise control functionality is designed for the model in order to maintain a constant velocity when the ego vehicle can drive freely. This feature is primarily

based on a proportional-integral-derivative (PID) controller. Through multiple simulations, the values of the proportional tuning constant (K_p) and the derivative constant (K_d) were set to **10.0** each. This value was found to be stable for the optimal smoothness at different velocities of the ego vehicle. For practical purposes, the output of the PID controller is limited to the maximum and minimum acceleration values. The desired acceleration is calculated through the equation 3.

$$\Delta^{cte} \coloneqq \dot{x}_{cruise} - \dot{x}_{curr},\ \ddot{x}_{PID} = K_p \times CTE + K_d \times \frac{\Delta^{cte}_{i+1} - \Delta^{cte}_{i}}{dt} \tag{3}$$

where subscripts *cruise* and *curr* stand for kinematics of the ego car for cruising and current state respectively. Note that the acceleration control is obtained via a PID controller. Δ^{cte} stands for *cross track error* while the time-step is $dt = 40ms$

3.5 ACC and Cruise Control modes combined

Our model has the ability to switch between the cruise mode and the ACC mode based on the threshold ACC distance (TAD) and the ACC mode and the safety mode based on the threshold safety distance (TSD) between the ego vehicle and the lead vehicle set by the user (1a) This inter vehicle distance (IVD) is a function of the long-range radar output from the ego vehicle.

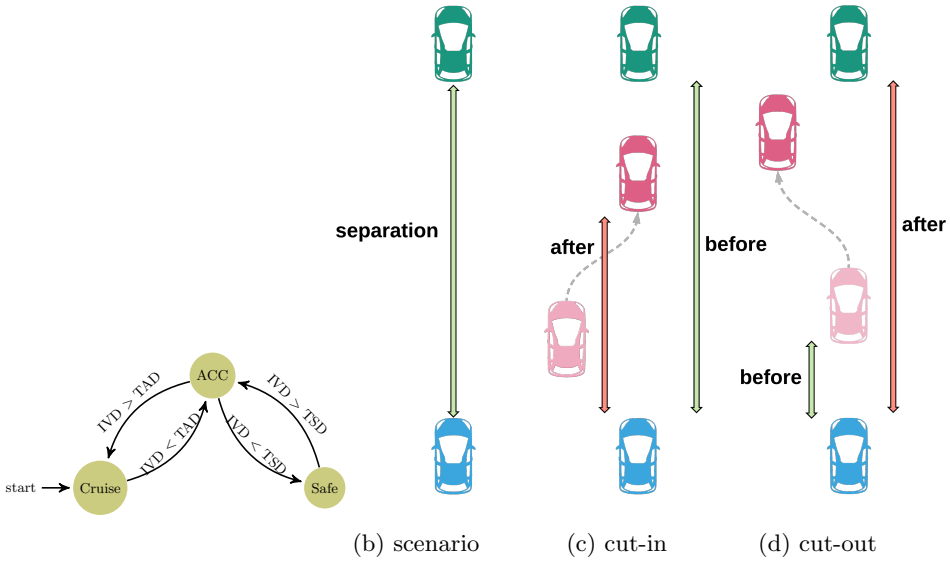
4 Implementation

For sake of simplicity, we first consider the `classical-ACC`. Here, we have the ego vehicle equipped with the radar.[3] Recall that in this setting, ACC is a purely longitudinal control problem and we assume that the vehicle is already controlled appropriately in the lateral direction so as to maintain the lane it is driving in. Another aspect to notice is that the cut-ins and cut-outs are allowed to happen which may result in the separation between the vehicles vary discontinuously as is evident in the Fig. 1c and Fig. 1d.

4.1 Experimental setup

In this section, we describe the extensive experiments that were carried out to evaluate `RL-ACC` empirically. For example, we first consider a simple scenario where there is only one leading vehicle with certain well-defined trajectory. This is named as `single-car`. While later on, we consider more realistic scenarios

[3] The vehicle is equipped with both long and short range radars with limits [80, 240] and [0.2 100] meters respectively. Unlike the problems like parking, the ACC does not require very close range detection. Here, sensor fusion is used to homogenise the readings from both the radars.



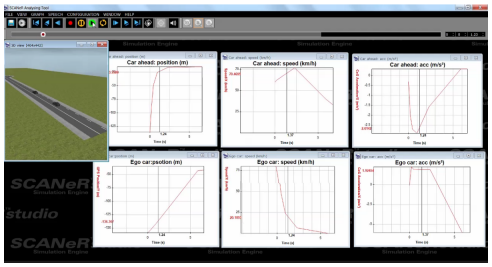
(a) state machine (b) scenario (c) cut-in (d) cut-out

Fig. 1: Finite state machine regarding the safe control. Cut-in and cut-outs are shown in the scenario of ACC.

where there are multiple vehicles in various lanes, all of which exhibit some sort of intelligent and rule-following behaviour. This is denoted as `multi-car`. Thus, we have considered two automotive-grade simulators, albeit for different scenarios of evaluations. In this section, we describe very briefly, each of these simulators and their importance. We also describe the overall software and hardware pipeline utilised for this work. In the later section to follow, we shall describe how the overall architecture and these components were used to perform the experiments.

Simulators: The development phase experiments are conducted primarily in two simulators besides other mathematical software tests and closed loop tests. The simulators used are the Oktal SCANeR^{TM}studio[2] and Vires[3]. A model test vehicle with similar dynamics to the real vehicle is created for these tests. In order to test different functionalities of the vehicle including more sophisticated algorithms, multiple scenarios are created with different number of vehicles, trucks etc. as well as lanes. Sensor fusion from radars on the ego vehicle provide the desired input for the ACC. The set-up is capable of running the entire test vehicle's distributed system in simulation. Hence, ACC being a closed-loop feature can be tested on top of other components like domain controller, sensor fusion etc.

Software: The learning algorithm itself is implemented in MATLAB for ease of prototyping whereas during the actual deployment the learned policy is in native C++ code on a distributed system set-up on ROS. The software stack is composed of several ROS nodes inter-communicating with each other at differ-



(a) Oktal SCANeR^{TM}studio

(b) DriveCore^{TM} Studio

Fig. 2: Simulation and the actual DriveCore Studio platform

ent levels based on their individual functionalities. Unfortunately, the detailed description of the system architecture is out of scope of this paper. The ROS nodes communicate to the simulators through a TCP/IP network configuration.

Hardware: The entire software stack runs on the DriveCore^{TM} platform (Fig 2b. It was designed as a centralised domain controller consisting of highly scalable hardware, in-vehicle middle-ware and PC-based software toolset all on a single platform handled by its three primary components - Compute, Runtime and the Studio. The detailed description of the platform is out of scope at the time of writing this paper, as the product will be released in CES 2018[1].

Overall architecture We have considered both the cases of ACC applications – on highway with high set-speed and in urban environment with low set-speed. Even in this simple set-up of one leading car, many interesting situations arise for `RL-ACC`. We call these as *scenarios*; e.g., we may start with a close-by car ahead which then cuts-out increasing the separation immensely or with different maximum-allowed speeds in the lane. There are other features like driver behaviour – a conservative driver in the car ahead might accelerate (and decelerate) slower than a sporty one – which we encapsulate in the module *environment*. This module along with the given scenario is sufficient for implementing a theoretical MATLAB model on top of which learning can be performed. Environment module also affects how the Oktal/Vires simulator generates the next instance of the world. The result of RL algorithm applied on these models is a policy which can then be deployed in the Oktal/Vires simulator for visualisation. In order to ensure safety of the learned policy, it is rigorously checked against desirable specification using the simulator and the scenarios. Once found satisfactorily safe, the policy is deployed in the real car, but not before passing a few other levels of software and closed loop checks as per the safety guidelines implemented by the Systems and the Validation teams. Overall pipeline is shown in Fig. 3.

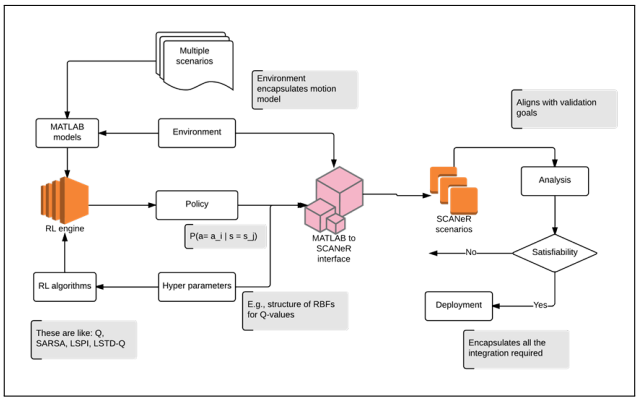


Fig. 3: Overall pipeline of learning-based ACC using SCANeR. Use of Vires is analogous.

5 Experiments

We conduct a variety of experiments to test `RL-ACC` in different scenarios. Like any other model of ACC, our primary criterion of success is for the ego vehicle to follow the lead vehicle at a safe distance while maintaining a smooth velocity profile. Besides that, we also compare velocity profiles with empirical models of ACC like the `IDM`. Finally, we try to explain our argument of the dynamic nature of `RL-ACC` through comparison of aggressive vs conservative behaviours of the controller in the same scenario.

5.1 Single-car scenario: `single-car`

The first set of experiments are conducted in Oktal in simple scenarios with relatively ideal conditions. There are 3 lanes (only 2 lanes shown in Fig 2a) designed in the shape of a racetrack, with each non-curvy side greater than 3000 m, where the `RL-ACC` is primarily tested. The ego vehicle is modelled after the Lincoln MKZ (used in US road testing by Visteon ADAS) and the single lead vehicle is generated by the simulator. We leverage the powerful feature of setting checkpoints on the road provided by Oktal. These checkpoints are useful for altering the environment agents and other parameters.

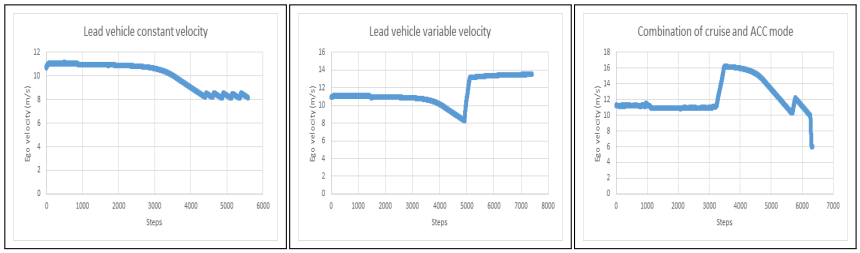
Lead vehicle with constant velocity In this experiment the lead vehicle maintains a constant velocity throughout its journey. The main objective here is to test the very basic expectation of any ACC model i.e - for the ego vehicle to be able to adjust its velocity based on the preceding vehicle in the simplest possible scenario with no other agents involved. From the velocity profile in Fig 4a, we can observe the drop in velocity from 11.11 m/s to 8.33 m/s between steps 3000-4000. Although we consider this outcome successful to an extent, it should be noted here that the model was still learning and hence the ACC velocity is not

perfectly smooth after stabilising. For the exact parameter values, please refer to Table 1.

Table 1: Agent/parameter values in `single-car` experiments (see Sec. 5.1)

Experiment	Parameter	Initial Conditions	Checkpoint A	Checkpoint B
Exp. 1	Ego vehicle	5.55 m/s	N/A	N/A
	Lead vehicle	8.33 m/s	N/A	N/A
	Distance between vehicles	> 200 m	N/A	N/A
	Cruise speed	11.11 m/s	N/A	N/A
Exp. 2	Ego vehicle	5.55 m/s	N/A	N/A
	Lead vehicle	8.33 m/s	13.89 m/s	N/A
	Distance between vehicles	< 150 m	N/A	N/A
	Cruise speed	11.11 m/s	13.89 m/s	N/A
Exp. 3	Ego vehicle	11.11 m/s	N/A	N/A
	Lead vehicle	8.33 m/s	13.89 m/s	11.11 m/s
	Distance between vehicles	< 150 m	N/A	N/A
	Cruise speed	11.11 m/s	16.67 m/s	N/A

Lead vehicle with variable velocity By this time we already know that `RL-ACC` is capable of following vehicles at a constant velocity while maintaining a safe distance. Hence, we step up a notch in the scenario complexity in the second experiment. The lead vehicle here changes its velocity after crossing a preset checkpoint on the simulator track, resulting in the ACC model to adapt to multiple velocities over the course of the journey. We can see from Fig 4b that the ego vehicle initially accelerates to reach the cruise speed (cruise mode), followed by decelerating once the ACC mode gets triggered and then accelerating again after the lead vehicle has changed its velocity post checkpoint A. Compared to Fig 4a the velocity profile is smoother irrespective of the drop or rise in the velocity. In Fig 4b, we can also observe that the ego vehicle decelerates over several steps in the first phase, but also has the ability to accelerate faster in a short span of time in order to match the velocity of the vehicle ahead. This is necessary as the lead vehicle changes its velocity to 13.89 m/s immediately after crossing checkpoint A due to simulator constraints. However it is a testament of the `RL-ACC` s flexibility to replicate human like driving, which always prefers smooth deceleration over hard breaking but is less conservative when it comes to accelerating.



(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

Fig. 4: Velocity profile in `single-car` scenarios

Test of uncertainty - sharp rise/drop in velocity of the lead vehicle in short ranges We often witness vehicles breaking away from their steady motion and either accelerating/decelerating abruptly in both highways and country roads. It is imperative that our model be able to adapt to these situations. Obviously, the model will jump back and forth between the cruise and the ACC modes frequently. Here, the lead vehicle changes its velocity to 13.89 m/s after crossing checkpoint A, moving away from the ego vehicle rapidly. As the inter-vehicle distance becomes greater than 140 m, the ego vehicle switches back to the cruise mode like initially and starts accelerating. Post checkpoint B, as the lead vehicle slows down, the ACC starts decelerating. The important test here is whether the model can handle a sharp drop in velocity (5.55 m/s) over a very short distance while maintaining a safe distance with the lead vehicle. The Fig 4c shows that it does adjust pretty well in this situation between steps 3500-5800. The final drop in velocity is at the end of the test track where the ego vehicle hard breaks in order to gradually slow down and stop eventually. This proves that our model can handle uncertain conditions well without manual human intervention, a key requirement in autonomous vehicles.

5.2 Multi-cars scenario: `multi-car`

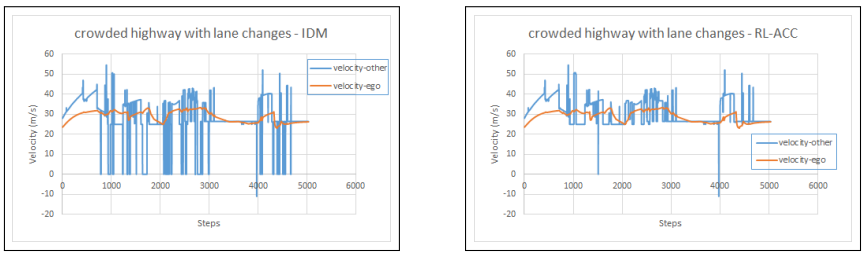
We use Vires for conducting experiments in more complex scenarios like realistic highway and residential roads with congested traffic while comparing with empirical models of ACC like the `IDM`. We designed two different scenarios for these experiments - 1) Crowded highway in the German Autobahn with a relatively higher congestion of vehicles (Fig 5a) and 2) Country side roads where the average speed is much lower besides other traffic restrictions (Fig 5b). The former helps us in testing our model on higher velocities as well as its adaptability to lane changes or other vehicles cut in/out, whereas the later creates many corner cases automatically because of the shorter inter-vehicle distances. We also test the conservative vs aggressive behaviour of our model on the country side roads. The ego vehicle is modelled on an Audi-A6 (used in German road testing by Visteon ADAS) here, whereas the other vehicles are generated from the simulator. There are no checkpoints unlike the experiments in Oktal, but instead we opt for smooth traffic flow with some hard-coded intelligent behaviour from other vehicles while following traffic rules of the respective scenarios.

Crowded highway with lane changes While it is relatively easier for human drivers to react to lane changes/merges with other vehicles cutting in/out, it might be more challenging for the ACC to adapt in real time, being dependent sensor fusion of outputs of both radars. Instead of reacting to this output directly, we consider it over consecutive time steps which enables smoother control and also makes the learner context-aware. Hence, when the lead vehicle changes or is absent in the new lane, the ACC resets to either the cruise mode (lead vehicle absent) or adjusts its velocity based on the new lead vehicle's velocity in a short span of time. We test our model on a long stretch of the Autobahn while manually changing lanes a number of times. In Fig 6, we can see that the ego

(a) highway scenario (b) residential country side scenario

Fig. 5: Two contexts of validating ACC

vehicle adjusts well, even in situations where the lead vehicle is out of range of the radar (blue line hitting zero). Our approach being model-free, `RL-ACC` can act in a more aggressive or conservative manner depending on the inter vehicle distance or average speed of the other vehicles in the respective lane and tuning of the rewards.

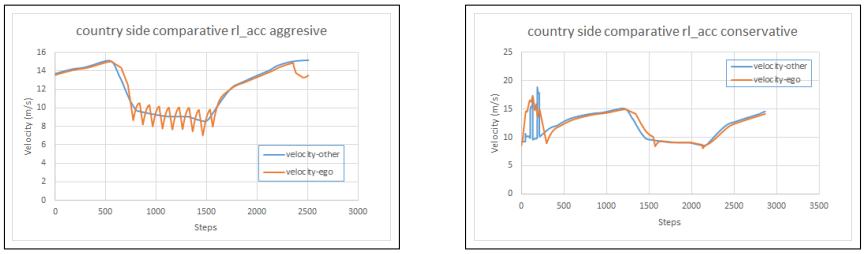


(a) IDM (b) Rl-ACC

Fig. 6: RL-ACC vs IDM in crowded highway with lane changes

Conservative vs aggressive RL-ACC on country side roads As discussed in 1 and 3, the primary advantage of the `RL-ACC` over `IDM` is its ability to adapt to different environments and driving conditions. Ideally, ACC should behave differently in highway scenarios and city roads. In the former, it should behave more conservatively i.e - the distance between vehicles should always be relatively large because of the average speed, whereas this distance would be a lot less in the later, resulting in a more aggressive behaviour of the ACC. In this experiment, we present our model behaving in a conservative vs aggressive manner on country side roads. The two sub-plots in Fig 7 are produced from different episodes of the journey, thus having different velocity trajectories. Looking carefully at the plots, we can see that the conservative `RL-ACC` is able to maintain a smoother velocity profile than the aggressive model. The aggressive behaviour of the car

here is the result of impatient driving style. Since our controller is designed for highway auto pilot we have set a high reward for optimal separation. When penalised appropriately for acceleration and deceleration, a smoother conservative behaviour is obtained.



(a) RL-ACC aggressive (b) RL-ACC conservative

Fig. 7: RL-ACC aggressive vs conservative behaviour

6 Conclusion

Typically, adaptive cruise control (ACC) is approached as a model-based controller design. In this paper, we have presented our approach to the control problem of ACC by using artificial intelligence techniques; in particular, we used reinforcement learning and termed it as `RL-ACC`. While the classical approach – `IDM`– which is widely used as the go-to ACC model in the automotive industry performs decently on the highways, we argue that due to its model-based approach, it lacks the ability to adapt to the environments or driving preferences. Since the `RL-ACC` does not require domain knowledge, it can be trained irrespective of the environment. Moreover this approach also makes the controller design intuitive from the perspective of human-driving. Complex control *arises* out of rather simple rewarding strategies. This enhances both the explain-ability as well as trust of human passengers when the autonomous car is utilising ACC. We believe this is one concrete step towards human-like autonomous driving.

This work opens several avenues of scientific research and application. The two most prominent ones are ensuring the safety of learned control and devising a general human-like controller for autonomous driving. In order to achieve the former, systematic verification of the controller can be performed using state-of-the-art formal methods. For achieving the latter goal, one of the approaches is to invoke the deep learning methods that can ingest the input images and learn ACC as some sort of optimal behaviour in different contexts. In future, we shall be pursuing both these avenues.

References

1. Drivecore - ces2018. http://wardsauto.com/technology/visteon-looks-play-big-role-autonomous-vehicles-drivecore/. Accessed:

Feb, 2018.

2. Oktal - simulation in motion. http://www.oktal.fr/en/automotive/range-of-simulators/software. Accessed: Feb, 2018.
3. Vtd - virtual test drive. https://vires.com/vtd-vires-virtual-test-drive/. Accessed: Feb, 2018.
4. Masako Bando, Katsuya Hasebe, Akihiro Nakayama, Akihiro Shibata, and Yuki Sugiyama. Dynamical model of traffic congestion and numerical simulation. *Physical review E*, 51(2):1035, 1995.
5. Erwin R Boer. Car following from the driver's perspective. *Transportation Research Part F: Traffic Psychology and Behaviour*, 2(4):201–206, 1999.
6. Mark Brackstone and Mike McDonald. Car-following: a historical review. *Transportation Research Part F: Traffic Psychology and Behaviour*, 2(4):181–196, 1999.
7. Denos C Gazis, Robert Herman, and Richard W Rothery. Nonlinear follow-the-leader models of traffic flow. *Operations research*, 9(4):545–567, 1961.
8. Peter G Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981.
9. R Gray and D Regan. Accuracy of estimating time to collision using binocular and monocular information. *Vision research*, 38(4):499–512, 1998.
10. W Helly. Simulation of bottlenecks in single lane traffic flow, presentation at the symposium on theory of traffic flow. *Research laboratories, General Motors, New York, pp207–238*, 1959.
11. Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec):1107–1149, 2003.
12. Michael L Littman, Thomas L Dean, and Leslie Pack Kaelbling. On the complexity of solving markov decision problems. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 394–402. Morgan Kaufmann Publishers Inc., 1995.
13. RM Michaels. Perceptual factors in car following. In *Proceedings of the 2nd International Symposium on the Theory of Road Traffic Flow (London, England), OECD*, 1963.
14. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
15. John Moody and Matthew Saffell. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, 12(4):875–889, 2001.
16. Kai Nagel and Michael Schreckenberg. A cellular automaton model for freeway traffic. *Journal de physique I*, 2(12):2221–2229, 1992.
17. Gordon Frank Newell. A simplified car-following theory: a lower order model. *Transportation Research Part B: Methodological*, 36(3):195–205, 2002.
18. Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Reinforcement learning for humanoid robotics. In *Proceedings of the third IEEE-RAS international conference on humanoid robots*, pages 1–20, 2003.
19. Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
20. Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.
21. R Wiedemann. Simulation des straßenverkehrsflusses. schriftenreihe heft 8. *Institute for Transportation Science, University of Karlsruhe, Germany*, 1994.