Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles

Szilárd Aradi, *Member, IEEE*

***Abstract*— Academic research in the field of autonomous vehicles has reached high popularity in recent years related to several topics as sensor technologies, V2X communications, safety, security, decision making, control, and even legal and standardization rules. Besides classic control design approaches, Artificial Intelligence and Machine Learning methods are present in almost all of these fields. Another part of research focuses on different layers of Motion Planning, such as strategic decisions, trajectory planning, and control. A wide range of techniques in Machine Learning itself have been developed, and this article describes one of these fields, Deep Reinforcement Learning (DRL). The paper provides insight into the hierarchical motion planning problem and describes the basics of DRL. The main elements of designing such a system are the modeling of the environment, the modeling abstractions, the description of the state and the perception models, the appropriate rewarding, and the realization of the underlying neural network. The paper describes vehicle models, simulation possibilities and computational requirements. Strategic decisions on different layers and the observation models, e.g., continuous and discrete state representations, grid-based, and camera-based solutions are presented. The paper surveys the state-of-art solutions systematized by the different tasks and levels of autonomous driving, such as car-following, lane-keeping, trajectory following, merging, or driving in dense traffic. Finally, open questions and future challenges are discussed.**

***Index Terms*— Machine learning, motion planning, autonomous vehicles, artificial intelligence, reinforcement learning.**

I. Introduction

MOTION planning for autonomous vehicle functions is a vast and long-researched area using a wide variety of approaches such as different optimization techniques, modern control methods, artificial intelligence, and machine learning. This article presents the achievements of the field from recent years focused on Deep Reinforcement Learning (DRL) approach. DRL combines the classic reinforcement learning with deep neural networks, and gained popularity after the breakthrough article from Deepmind [1], [2]. In the number of research papers about autonomous vehicles and the DRL has been increased in the last few years (see Fig. 1.), and because of the complexity of the different motion planning problems, it is a convenient choice to evaluate the applicability of DRL for these problems.

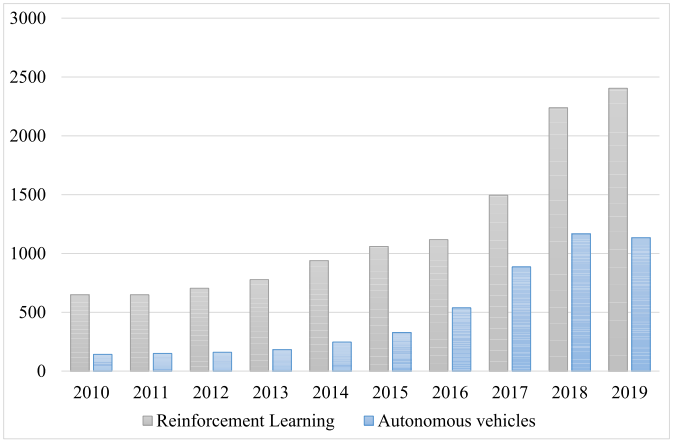


Fig. 1. Web of science topic search for "Deep Reinforcement Learning" and "Autonomous vehicles (2020.01.17.)".

A. The Hierarchical Classification of Motion Planning for Autonomous Driving

Using deep neural networks for self-driving cars gives the possibility to develop "end-to-end" solutions where the system operates like a human driver: its inputs are the travel destination, the knowledge about the road network and various sensor information, and the output is the direct vehicle control commands, e.g., steering, torque, and brake. However, on the one hand, realizing such a scheme is quite complicated, since it needs to handle all layers of the driving task, on the other hand, the system itself behaves like a black box, which raises design and validation problems. By examining the recent advantages in the field, it can be said that most researches focus on solving some sub-tasks of the hierarchical motion planning problem. This decision-making system of autonomous driving can be decomposed into at least four layers, as stated in [3] (see Fig.2.). Route planning, as the highest level, defines the waypoints of the journey based on the map of the road network, with the possibility of using real-time traffic data. Though optimal route choice has a high interest among the research community, papers dealing with this level do not employ

Manuscript received January 28, 2020; revised July 2, 2020; accepted September 8, 2020. Date of publication September 30, 2020; date of current version February 2, 2022. This work was supported in part by the Hungarian Government and in part by the European Social Fund through the project "Talent management in autonomous vehicle control technologies" under Grant EFOP-3.6.3-VEKOP-16-2017-00001. The Associate Editor for this article was J. W. Choi.

The author is with the Department of Control for Transportation and Vehicle Systems, Budapest University of Technology and Economics, 1111 Budapest, Hungary (e-mail: aradi.szilard@mail.bme.hu).

Digital Object Identifier 10.1109/TITS.2020.3024655

reinforcement learning. A comprehensive study on the subject can be found in [4].

The Behavioral layer is the strategic level of autonomous driving. With the given way-points, the agent decides on the short term policy, by taking into consideration the local road topology, the traffic rules, and the perceived state of other traffic participants. Having a finite set of available actions for the driving context, the realization of this layer is usually a finite state-machine having basic strategies in its states (i.e., car following, lane changing, etc.) with well-defined transitions between them based on the change of the environment. However, even with the full knowledge of the current state of the traffic, the future intentions of the surrounding drivers are unknown, making the problem partially observable [5]. Hence the future state not only depends on the behavior of the ego vehicle but also relies on unknown processes; this problem forms a Partially Observable Markov Decision Process (POMDP). Different techniques exist to mitigate these effects by predicting the possible trajectories of other road users, like in [6], where the authors used gaussian mixture models, or in [7], where support vector machines and artificial neural networks were trained based on recorded traffic data. Since finite action POMDPs are the natural way of modeling reinforcement learning problems, a high amount of research papers deal with this level, as can be seen in the sections of the paper. To carry out the strategy defined by the behavioral layer, the motion planning layer needs to design a feasible trajectory consisting of the expected speed, yaw, and position states of the vehicle on a short horizon. Naturally, on this level, the vehicle dynamics has to be considered, hence classic exact solutions of motion planning are impractical since they usually assume holonomic dynamics. It has long been known that the numerical complexity of solving the motion planning problem with nonholonomic dynamics is Polynomial-Space Algorithm (PSPACE) [8], meaning it is hard to elaborate an overall solution by solving the nonlinear programming problem in real-time [9]. On the other hand, the output representation of the layer makes it hard to directly handle it with "pure" reinforcement learning, only a few papers deal solely with this layer, and they usually use DRL to define splines as a result of the training [10], [11].

At the lowest level, the local feedback control is responsible for minimizing the deviation from the prescribed path or trajectory. A significant amount of papers reviewed in this article deals with the aspects of this task, where lane-keeping, trajectory following, or car following is the higher-level strategy. Though at this level, the action space becomes continuous, and classical approaches of RL can not handle this. Hence discretization of the control outputs is needed, or - as in some papers - continuous variants of DRL are used.

B. Reinforcement Learning

As an area of Artificial Intelligence and Machine Learning, Reinforcement learning (RL) deals with the problem of a learning agent placed in an environment to achieve a goal. Contrary to supervised learning, where the learner structure gets examples of good and bad behavior, the RL agent must discover by trial and error how to behave to get the most reward [12]. For this task, the agent must percept the state of the environment at some level and based on this information, and it needs to take actions that result in a new state. As a result of its action, the agent receives a reward, which aids in the development of future behavior. To ultimately formulate the problem, modeling the state transitions of the environment, based on the actions of the agent is also a necessity. This leads to the formulation of a POMDP defined by the functions of $(\mathcal{S}, \mathcal{A}, T, R, \Omega, O)$, where $\mathcal{S}$ is the set of environment states, $\mathcal{A}$ is the set of possible actions in that particular state, T is the transition function between the states based on the actions, R is the reward for the given $(\mathcal{S}, \mathcal{A})$ pair, while Ω is the set of observations, and O is the sensor model. The agent in this context can be formulated by any inference model whose parameters can be modified in response to the experience gained. In the context of Deep Reinforcement Learning, this model is implemented by neural networks.

The problem in the POMDP scenario is that the current actions affect the future states, therefore the future rewards, meaning that for optimizing the behavior for the cumulative reward throughout the entire episode, the agent needs to have information about the future consequences of its actions. RL has two main approaches for determining the optimal behavior: value-based and policy-based methods.

The original concept using a value-based method is the Deep-Q Learning Network (DQN) introduced in [1]. Described briefly, the agent predicts a so-called Q value for each state-action pair, which formulate the expected immediate and future reward. From this set, the agent can choose the action with the highest value as an optimal policy or can use the values for exploration during the training process. The main goal is to learn the optimal Q function, represented by a neural network in this case. This can be done by conducting experiments, calculating the discounted rewards of the future states for each action, and updating the network by using the Bellman-equation [13] as a target. Using the same network for value evaluation and action selection results in unstable behavior and slow learning in noisy environments. Meta-heuristics, such as experience replay, can handle this problem, while other variants of the original DQN exist, such as Double DQN [14] or Dueling DQN [15], separating the action and the value prediction streams, leading to faster and more stable learning.

Policy-based methods target at choosing the optimal behavior directly, where the policy π_Θ is a function of $(\mathcal{S}, \mathcal{A})$. Represented by a neural network, with a softmax head, the agent generally predicts a normalized probability of the expected goodness of the actions. In the most natural implementation, this output integrates the exploration property of the RL process. In advanced variants, such as the actor-critic, the agent uses different predictions for the value and the action [16]. Initially, RL algorithms use finite action space, though, for many control problems, they are not suitable. To overcome this issue in [17] introduced the Deep Deterministic Policy Gradients (DDPG) agent, where the actor directly maps states to continuous actions.

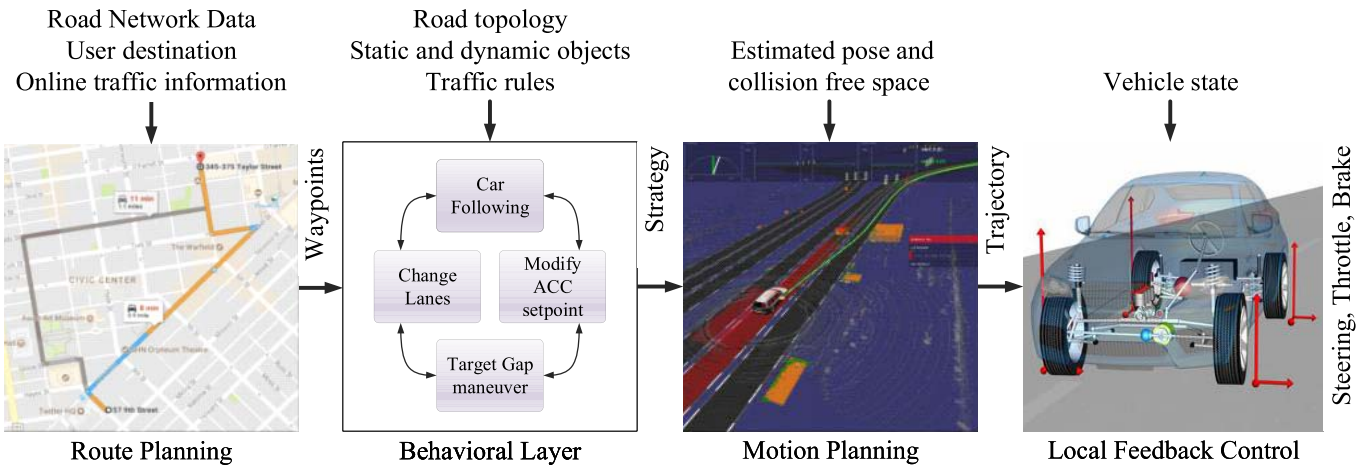


Fig. 2. Layers of motion planning.

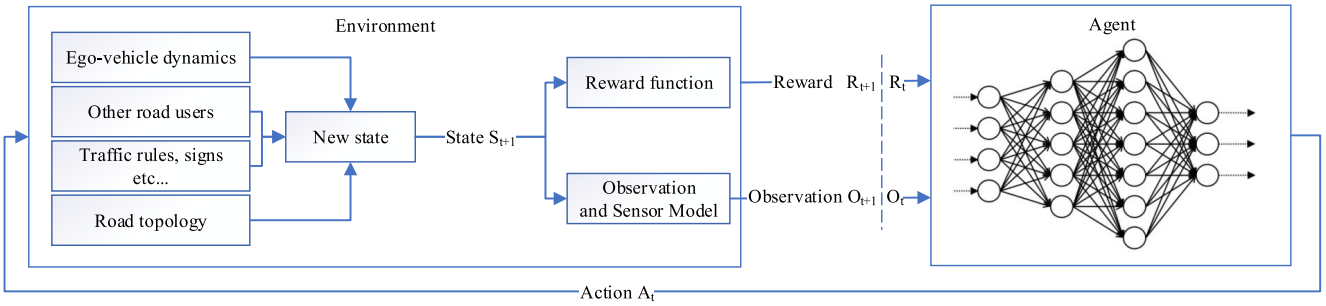


Fig. 3. The POMDP model for deep reinforcement learning based autonomous driving.

For complex problems, the learning process can still be long or even unsuccessful. It can be soluted in many ways:

- Curriculum learning describes a type of learning in which the training starts with only easy examples of a task and then gradually increase difficulty. This approach is used in [18]–[20].
- Adversarial learning aims to fool models through malicious input. Papers using variants of this technique are: [21], [22]
- Model-based action choice, such as the MCTS based solution of Alpha-Go, can reduce the effect of the problem of distant rewarding.

Since reinforcement learning models the problem as a POMDP, a discrete-time stochastic control process, the solutions need to provide a mathematical framework for this decision making in situations where outcomes are partly random and partly under the control of a decision-maker, while the states are also partly observable [23]. In the case of motion planning for autonomous or highly automated vehicles, the tuple $(\mathcal{S}, \mathcal{A}, T, R, O)$ of the POMDP is illustrated in Fig. 3 and can be interpreted as follows:

$\mathcal{S}, \mathcal{A}, T$, and R describe the MDP, the modeling environment of the learning process. It can vary depending on the goals, though in our case it needs to model the dynamics of the vehicle, the surrounding static and dynamic objects, such as other participants of the traffic, the road topology, lane markings, signs, traffic rules, etc. $\mathcal{S}$ holds the current actual state of the simulation. A is the possible set of actions of the agent driving the ego-car, while T, the so-called state-transition function updates the vehicle state and also the states of the traffic participants depending on the action of the vehicle. The different levels of abstraction are described in section II-A. Many research papers use different software platforms for modeling the environment. A brief collection of the used frameworks are presented in section II-B. R is the reward function of the MDP, section II-D gives a summary on this topic.

Ω is the set of observations the agent can experience in the world, while O is the observation function giving a possibility distribution over the possible observations. In more uncomplicated cases, the studies assume full observability and formulate the problem as an MDP, though in many cases, the vehicle does not possess all information. Another interesting topic is the representation of the state observation, which is a crucial factor for the architecture choice and performance of Deep RL agents. The observation models used in the literature are summarized in section II-E.

C. Multiagent Reinforcement Learning

As mentioned before, the lower layers of motion planning, like trajectory following or simple control tasks do not require interactions with agents, whose reaction depends

on the behavior of the ego-vehicle. However, on the higher levels, where the vehicle is placed in complex situations, like racing, passing intersections, merging, or driving in traffic, the other participants' reactions strongly affect the available choices and possible outcomes. This leads to the area of Multiagent Systems (MAS) [24], which if handled with RL techniques are called Multiagent (Deep) Reinforcement Learning (MARL or MDRL in different sources) [25]. One modeling approach to MARL is the generalization of the original POMDP, by extending it with multiple actions and observation sets for each agent, or even various rewards in case different agents have different goals. This approach is called decentralized partially observable Markov decision process (DEC-POMDP) [26], [27].

Naturally, some of the problems in this domain can still be handled by single-agent approaches, where one embeds all other agents in a previously defined model with predefined or rule-based behaviors and create an independent learning environment for a single agent; or even with totally independent learners, where all other agents are just part of the environment of the actual learner. This comes with the danger that the policies found can overfit to the environment agents' strategies and hence not generalize well [28].

Dealing with MARL raises multiple additional questions above single-agent RL problems, as it is numerically and technically more complex and has many conceptual issues to deal with [29]. The first is the nature of the "game", whether it is cooperative or competitive, which highly affects the credit assignment, i.e., the calculation and distribution of rewards. Zero-sum games usually lead to competitive scenarios, since one agent can only benefit the other's loss. Among the problems of vehicle motion planning, racing is one example of such a MAS problem. Also, there are clearly cooperative problems when only the success of all participants is acceptable. Some traffic scenarios can be considered from both perspectives. For example, in intersection or highway driving situations, one can train agents for shortest individual travel time, or the average travel time of all agents. Even though when the intentions are clear, credit assignment is not trivial and can lead to different learning dynamics or unintended results [30].

The heterogeneity of the agents' knowledge or tasks is also a design aspect. It is not trivial that all agents have to behave similarly, even if their individual goals are the same. Furthermore, in some scenarios, like merging into traffic, the agents have distinct tasks: the ones already traveling in the target lane should decide to adjust the gap for the merging vehicle, and its agent is supposed to navigate into this target gap.

This leads to the last significant difference compared to single-agent systems that in MAS, the agents have the opportunity to communicate through messages [31] or by memory sharing [32]. This setting usually assumes a partially observable environment and cooperative agents [33], and could serve two purposes: one is the transfer of observations, that are hidden from the other agents, and the second is to transfer intended behavior to achieve better joint performance. Both make sense in driving scenarios. For example, considering a highway platoon, the radar of each vehicle can only sense the closest car in front. However, they could react better, having information about all the cars ahead, and also the knowledge on their intended braking or acceleration could help. Communication in MAS is a relatively new field with promising results, though it has many open questions [34].

And finally, there are different training-schemes for MARL. Its main classes are the following. The first is the centralized controller approach, where there is a joint model for all of the observations and actions of all agents. In the first place, this could be an optimal approach, though it is a single agent controlling multiple agents. On the other hand, with the growth of the number of agents, the complexity of the action space grows exponentially, making exploration extremely hard [35]. An opposite to this is the concurrent learning, where each agent has an individual policy, with private observation and action space. Heterogeneous tasks would rationalize this approach, though it also has its drawbacks. The first is that each agent has its own learning process, hence the overall learning resource requirements (memory, calculations) grow linearly with the number of agents. The second is that since the agents adjust their policies to the behavior of the others, the dynamics of the learning can become cyclic, like in a naive rock-paper-scissors game [36]. The third and less resource-intensive approach is parameter sharing, where the agents develop a common policy, with all their unique experiences. This does not mean the same behavior since the state and observation of each agent can differ.

II. Modeling for Reinforcement Learning

A. Vehicle Modeling

Modeling the movement of the ego-vehicle is a crucial part of the training process since it raises the trade-off problem between model accuracy and computational resource. Since RL techniques use a massive number of episodes for determining optimal policy, the step time of the environment, which highly depends on the evaluation time of the vehicle dynamics model, profoundly affects training time. Therefore during environment design, one needs to choose from the simplest kinematic model to more sophisticated dynamics models ranging from 2 Degree of Freedom (2DoF) lateral model to the more and more complex models with a higher number of parameters and complicated tire models.

At rigid kinematic single-track vehicle models, which neglect tire slip and skip, lateral motion is only affected by the geometric parameters. Therefore, they are usually limited to low-speed applications. More details about the model can be found in [37]. The simplest dynamic models with longitudinal and lateral movements are based on the 3 Degrees of Freedom (3DoF) dynamic bicycle model, usually with a linear tire model. They consider $(V_x, V_y, \dot{\Psi})$ as independent variables, namely longitudinal and lateral speed, and yaw rate. A more complex model is the four-tire 9 Degrees of Freedom (9DoF) vehicle model, where amongst the parameters of the 3DoF, body roll and pitch $(\dot{\Theta}, \dot{\Phi})$ and the angular velocities of the four wheels $(\omega_{fl}, \omega_{fr}, \omega_{rl}, \omega_{rr})$ are also considered, to calculate tire forces more precisely. Hence the model takes

into account both the coupling of longitudinal and lateral slips and the load transfer between tires.

Though the kinematic model seems quite simplified, and as stated in [38], such a model can behave significantly different from an actual vehicle, though for the many control situations, the accuracy is suitable [37].

According to [38], using a kinematic bicycle model with a limitation on the lateral acceleration at around $0.5g$ or less provides appropriate results, but only with the assumption of dry road. Above this limit, the model is unable to handle dynamics. Hence a more accurate vehicle model should be used when dealing with higher accelerations to push the vehicle's dynamics near its handling limits.

Regarding calculation time, based on the kinematic model, the calculation of the 3DoF model can be $10 \dots 50$ times higher, and the precise calculation of a 9DoF model with nonlinear tire model can be $100 \dots 300$ times higher, which is the main reason for the RL community to use a low level of abstraction.

Modeling traffic and surrounding vehicles is often performed by using unique simulators, as described in section II-B. Some authors develop their environments, using cellular automata models [39]. Some use MOBIL, which is a general model (minimizing overall braking induced by lane change) to derive lane-changing rules for discretionary and mandatory lane changes for a broad class of car-following models [40]; the Intelligent Driving Model (IDM), a continuous microscopic single-lane model [41].

B. Simulators

Some authors create self-made environments to achieve full control over the model, though there are commercial and Open-source environments that can provide this feature. This section briefly identifies some of them used in recent researches in motion planning with RL.

In modeling the traffic environment, the most popular choice is SUMO (Simulation of Urban MObility), which is a microscopic, inter- and multi-modal, space-continuous and time-discrete traffic flow simulation platform [42]. It can convert networks from other traffic simulators such as VISUM, Vissim, or MATSim and also reads other standard digital road network formats, such as OpenStreetMap or OpenDRIVE. It also provides interfaces to several environments, such as python, Matlab,.Net, C++, etc. Though the abstraction level, in this case, is microscopic, and vehicle behavior is limited, its ease of use and high speed makes it an excellent choice for training agents to handle traffic, though it does not provide any sensor model besides the ground truth state of the vehicles.

Another popular microscopic simulator that has been used commercially and for research also is VISSIM [43]. In [44] it is used for developing car-following behavior and lane changing decisions.

Considering only vehicle dynamics, the most popular choice is TORCS (The Open Racing Car Simulator), which is a modern, modular, highly portable multi-player, multi-agent car simulator. Its high degree of modularity and portability render it ideal for artificial intelligence research [45]. Interfacing with python, the most popular AI research environment is comfortable and runs at an acceptable speed. TORCS also comes with different tracks, competing robots, and several sensor models.

It is assumed that for vehicle dynamics, the best choices would be the professional tools, such as CarSIM [46] or CarMaker [47], though the utilization of these softwares can not be found in the reinforcement learning literature. This may be caused by the fact that these are expensive commercial platforms, though more importantly, their lack of python interfaces and high precision, but resource-intensive models prevent them from running several episodes within a reasonable time.

For more detailed sensor models or traffic, the authors usually use Airsim, Udacity Gazebo/ROS, and CARLA:

AirSim, used by a recent research in [48], is a simulator initially developed for drones built on Unreal Engine now has a vehicle extension with different weather conditions and scenarios.

Udacity, used in [49], is a simulator that was built for Udacity's Self-Driving Car Nanodegree [50] provides various sensors, such as high quality rendered camera image LIDAR and Infrared information, and also has capabilities to model other traffic participants.

Another notable mention is CARLA, an open-source simulator for autonomous driving research. CARLA has been developed from the ground up to support the development, training, and validation of autonomous urban driving systems. In addition to open-source code and protocols, CARLA provides open digital assets (urban layouts, buildings, vehicles) that were created for this purpose and can be used freely. The simulation platform supports flexible specification of sensor suites and environmental conditions [51].

Though this section provides only a brief description of the simulators, a more systematic review of the topic can be found in [52].

C. Action Space

The choice of action space highly depends on the vehicle model and task designed for the reinforcement learning problem in each previous research. Though two main levels of control can be found: one is the direct control of the vehicle by steering braking and accelerating commands, and the other acts on the behavioral layer and defines choices on strategic levels, such as lane change, lane keeping, setting ACC reference point, etc. At this level, the agent gives a command to low-level controllers, which calculate the actual trajectory. Only a few papers deal with the motion planning layer, where the task defines the endpoints (x, y, θ), and the agent defines the knots of the trajectory to follow represented as a spline, as can be seen in [11]. Also, few papers deviate from vehicle motion restrictions and generate actions by stepping in a grid, like in classic cellular automata microscopic models [53].

Some papers combine the control and behavioral layers by separating longitudinal and lateral tasks, where longitudinal acceleration is a direct command, while lane changing is a strategic decision like in [54].

The behavioral layer usually holds a few distinct choices, from which the underlying neural network needs to choose, making it a classic reinforcement learning task with finite actions.

Though on the level of control, the actuation of vehicles, i.e., steering, throttle, and braking, are continuous parameters and many reinforcement learning techniques like DQN and PG can not handle this since they need finite action set, while some, like DDPG, works with continuous action space. To adapt to the finite action requirements of the RL technique used, most papers discretizes the steering and acceleration commands to 3 to 9 possibilities per channel. The low number of possible choices pushes the solution farther from reality, which could raise vehicle dynamics issues with uncontrollable slips, massive jerk, and yaw-rate, though the utilization of kinematic models sometimes covers this in the papers. A large number of discrete choices, however, ends up in an exponential growth in the possible outcomes in the POMDP approach, which slows down the learning process.

D. Rewarding

During training, the agent tries to fulfill a task, generally consisting of more than one step. This task is called an episode. An episode ends if one of the following conditions are met:

- The agent successfully fulfills the task;
- The episode reaches a previously defined steps
- A terminating condition rises.

The first two cases are trivial and depend on the design of the actual problem. Terminal conditions are typically situations where the agent reaches a state from which the actual task is impossible to fulfill, or the agent makes a mistake that is not acceptable. Vehicle motion planning agents usually use terminating conditions, such as: collision with other participants or obstacles or leaving the track or lane, since these two inevitably end the episode. There are lighter approaches, where the episode terminates with failure before the accident occurred, with examples of having a too high tangent angle to the track or reaching too close to other participants. These "before accident" terminations speed up the training by bringing the information of failure forward in time, though their design needs caution [55].

Rewarding plays the role of evaluating the goodness of the choices the agent made during the episode giving feedback to improve the policy. The first important aspect is the timing of the reward, where the designer of the reinforcement learning solution needs to choose a mixture of the following strategies all having their pros and cons:

- Giving reward only at the end of the episode and discounting it back to the previous $(\mathcal{S}, \mathcal{A})$ pairs, which could result in a slower learning process, though minimizes the human-driven shaping of the policy.
- Giving immediate reward at each step by evaluating the current state, naturally discount also appears in this solution, which results in significantly faster learning, though the choice of the immediate reward highly affects the established strategy, which sometimes prevents the agent from developing better overall solutions than the one that gave the intention of the designed reward.
- An intermediate solution can be to give a reward in predefined periods or travel distance [56], or when a good or bad decision occurs.

In the area of motion planning, the end episode rewards are calculated from the fulfillment or failure of the driving task. The overall performance factors are generally: time of finishing the task, keeping the desired speed or achieving as high average speed as possible, yaw or distance from lane middle or the desired trajectory, overtaking more vehicles, achieve as few lane changes as possible [57], keeping right [58], [59] etc. Rewarding systems also can represent passenger comfort, where the smoothness of the vehicle dynamics is enforced. The most used quantitative measures are the longitudinal acceleration [60], lateral acceleration [61], [62] and jerk [10], [63].

In some researches, the reward is based on the deviation from a dataset [64], or calculated as a deviation from a reference model like in [65]. These approaches can provide favorable results, though a bit tends from the original philosophy of reinforcement learning since a previously known strategy could guide the learning.

E. Observation Space

The observation space describes the world to the agent. It needs to give sufficient information for choosing the appropriate action, hence - depending on the task - it contains the following knowledge:

- The state of the vehicle in the world, e.g., position, speed, yaw, etc.
- Topological information like lanes, signs, rules, etc.
- Other participants: surrounding vehicles, obstacles, etc.

The reference frame of the observation can be absolute and fixed to the coordinate system of the world, though as the decision process focuses on the ego-vehicle, it is more straightforward to choose an ego-centric reference frame pinned to the vehicle's coordinate system, or the vehicle's position in the world, and the orientation of the road. It allows concentrating the distribution of visited states around the origin in both position, heading, and velocity space, as other vehicles are often close to the ego-vehicle and with similar speed and heading, reducing the region of state-space in which the policy must perform. [66]

1) Vehicle State Observation: For lane keeping, navigation, simple racing, overtaking, or maneuvering tasks, the most commonly used and also the simplest observation for the ego vehicle consists of the continuous variables of $(|e|, v, \theta_e)$ describing the lateral position from the center-line of the lane, vehicle speed, and yaw angle respectively. (see Fig. 4). This information is the absolute minimum for guiding car-like vehicles, and only eligible for the control of the classical kinematic car-like models, where the system implies the motion without skidding assumption. Though in many cases in the literature, this can be sufficient, since the vehicles remain deep in the dynamically stable region.

For tasks, where more complex vehicle dynamics is inevitable, such as racing situations, or where the stability of

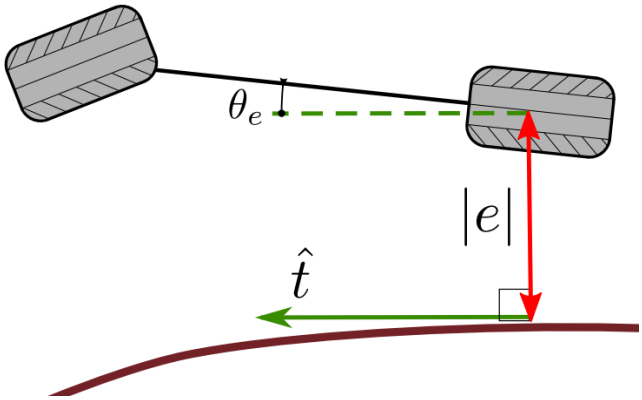


Fig. 4. Observation for basic vehicle state (source: [3]).

Fig. 5. Real images from the driving data and their semantic segmentations (source: [72]).

the vehicle is essential, this set of observable state would not be enough, and it should be extended with yaw, pitch, roll, tire dynamics, and slip.

2) Environment Observation: Getting information about the surroundings of the vehicle and representing it to the learning agent shows high diversity in the literature. Different levels of sensor abstractions can be observed:

- sensor level, where camera images, lidar or radar information is passed to the agent;
- intermediate level, where idealized sensor information is provided;
- ground truth level, where all detectable and non-detectable information is given.

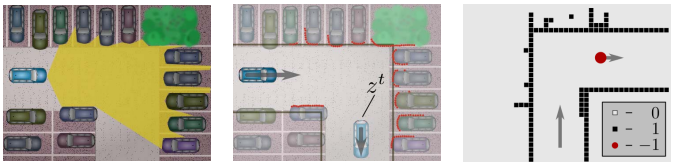
The structure of the sensor model also affects the neural network structure of the Deep RL agent since image like, or array-like inputs infer 2D or 1D CNN structures, while the simple set of scalar information results in a simple dense network. There are cases where these two kinds of inputs are mixed. Hence the network needs to have two different types of input layers.

Image-based solutions usually use front-facing camera images extracted from 3D simulators to represent the observation space. The data is structured in a (C x W x H) sized matrix, where C is the number of channels, usually one for intensity images and three for RGB, while W and H are the width and height resolution of the image. In some cases, for the detection of movement, multiple images are fed to the network in parallel. Sometimes it is convenient to down-sample the images - like (1x48x27) in [67] or (3x84x84) in [68], [69] - for data and network compression purposes. Since images hold the information in an unstructured manner, i.e., the state information, such as object positions, or lane information are deeply encoded in the data, deep neural networks, such as CNN, usually need large samples and time to converge [70]. This problem escalates, with the high number of steps that the RL process requires, resulting in a lengthy learning process, like $1.5M$ steps in [67] or $100M$ steps in [68].

Many image-based solutions propose some kind of pre-processing of the data to overcome this issue. In [70], the authors propose a framework for vision-based lateral control, which combines DL and RL methods. To improve the perception accuracy, an MTL (Multitask learning) CNN model is proposed to learn the critical track features, which are used to locate the vehicle in the track coordinate, and trains a policy gradient RL controller to solve the continuous sequential decision-making problem. Naturally, this approach can also be viewed as an RL solution with structured features, though the combined approach has its place in the image-based solutions also.

Another approach could be the simplification of the unstructured data. In [71] Kotyan *et al.* uses the difference image as the background subtraction between the two consecutive frames as an input, assuming this image contains the motion of the foreground and the underlying neural network would focus more on the features of the foreground than the background. By using the same training algorithm, their results showed that the including difference image instead of the original unprocessed input needs approximately 10 times less training steps to achieve the same performance. The second possibility is, instead of using the original image as an input, it can be driven through an image semantic segmentation network as proposed in [72]. As the authors state: "Semantic image contains less information compared to the original image, but includes most information needed by the agent to take actions. In other words, semantic image neglects useless information in the original image." Another advantage of this approach is that the trained agent can use the segmented output of images obtained from real-world scenarios, since on this level, the difference is much smaller between the simulated and real-world data than in the case of the simulated and real-world images. Fig. 5 shows the 640×400 resolution inputs used in this research.

2D or 3D Lidar like sensor models are not common among the recent studies, though they could provide excellent depth-map like information about the environment. Though the same problem arises as with the camera images, that the provided data - let them be a vector for 2D, and a matrix for 3D Lidars - is unstructured. The usage of this type of input only can be found in [73], where the observation emulates a 2D Lidar that provides the distance from obstacles in 31 directions within the field-of-view of 150°, and agent uses sensor data as its state. A similar input structure, though not modeling a Lidar,



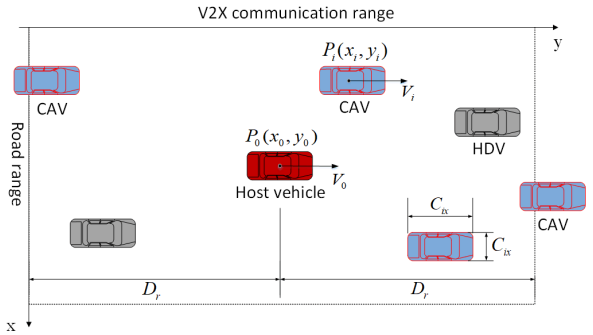
(a) Sensors (b) Target state z^t (c) Perception Ø

Fig. 6. The surrounding from the perspective of the vehicle can be described by a coarse perception map where the target is represented by a red dot (c) (source: [78]).

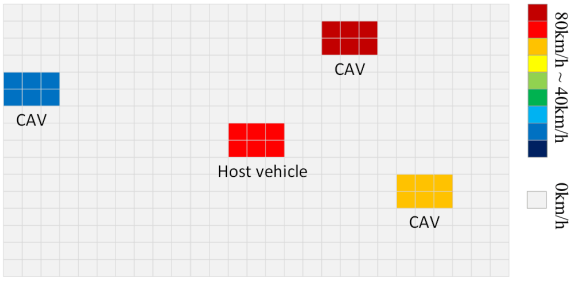
since there is no reflection, which is provided by TORCS and used in [20], is to represent the lane markings with imagined beam sensors. The agent in the cited example uses readings from 19 sensors with a 200m range, presenting at every 10° on the front half of the car returning distance to the track edge.

Grid-based path planning methods, like the A* or various SLAM (Simultaneous Localization and Mapping) algorithms exist and are used widespread in the area of mobile robot navigation, where the environment is represented as a spatial map [74], usually formulated as a 2D matrix assigning to each 2D location in a surface grid one of three possible values: Occupied, free, and unknown [75]. This approach can also be used representing probabilistic maneuvers of surrounding vehicles [76], or by generating spatiotemporal map from a predicted sequence of movements, motion planning in a dynamic environment can also be achieved [77]. Though the previously cited examples didn't use RL techniques, they prove that grid representation holds high potential in this field. Navigation in a static environment by using a grid map as the observation, together with position and yaw of the vehicle with an RL agent, is presented in [78] (See Fig.6). Grid maps are also unstructured data, and their complexity is similar to the semantically segmented images, since the cells store class information in both cases, and hence their optimal handling is using the CNN architecture.

Representing moving objects, i.e. surrounding vehicles in a grid needs not only occupancy, but other information hence the spatial grid's cell need to hold additional information. In [57] the authors used equidistant grid, where the ego-vehicle is placed in the center, and the cells occupied by other vehicles represented the longitudinal velocity of the corresponding car (See Fig. 7). The same approach can also be found in [62]. Naturally this simple representation can not provide information about the lateral movement of the other traffic participants, though they give significantly more than the simple occupancy based ones. Equidistant grids are a logical choice for generic environments, where the moving directions of the mobile robot are free, though, in the case of road vehicles, the vehicle mainly follows the direction of the traffic flow. In this case, the spatial representation could be chosen fixed to the road topology, namely the lanes of the road, regardless of its curvature or width. In these lane-based grid solutions, the grid representing the highway has as many rows as the actual lane count, and the lanes are discretized longitudinally. The simplest utilization of this approach can be found in [39], where the length of the cells is equivalent



(a) Mathematical model for the traffic



(b) Visualization of the Hyper Grid Matrix

Fig. 7. The visualization of the HDM mapping process (source: [57]).

to the unit vehicle length, and also, the behavior of the traffic acts similar to the classic cellular automata-based microscopic models [79].

This representation, similarly to the equidistant ones, can be used for occupancy, though they still do not hold any information on vehicle dynamics. [80] is to fed multiple consecutive traffic snapshots into the underlying CNN structure, which inherently extracts the velocity of the moving objects. Representing speed in grid cells is also possible in this setup, for that example can be found in [49], where the authors converted the traffic extracted from the Udacity simulator to the lane-based grid.

Besides the position and the longitudinal speed of the surrounding vehicles are essential from the aspect of the decision making, other features (such as heading, acceleration, lateral speed) should be considered. Multi-layer grid maps could be used for each vital parameter to overcome this issue. In [10] the authors processed the simulator state to calculate an observation tensor of size 4 x 3 x (2 x FoV + 1), where Fov stands for Field of View and represents the maximum distance of the observation in cell count. There is one channel (first dimension) each for on-road occupancy, relative velocities of vehicles, relative lateral displacements, and relative headings to the ego-vehicle. Fig.8 shows an example of the simulator state and corresponding input observation used for their network.

The previous observation models (image, lidar, or grid-based) all have some common properties: All of them are unstructured datasets, need a CNN architecture to process, which hardens the learning process since the agent simultaneously needs to extract the exciting features and form the policy for action. It would be obvious to pre-process the unstructured

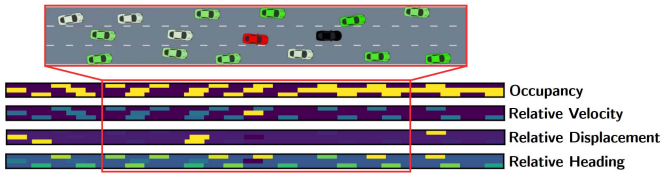


Fig. 8. The simulator state (top, zoomed in) gets converted to a 4 x 3 x (2 x FoV + 1) input observation tensor (bottom) (source: [10]).

data and feed structured information to the agents' network. Structured data refers to any data that resides in a fixed field within a record or file. As an example, for navigating in traffic, based on the task, the parameters of the surrounding vehicles are represented on the same element of the input. In the simplest scenario of car following, the agent only focuses on the leading vehicle, and the input beside the state of the ego vehicle consists of (d, v) as in [64] or (d, v, a) as in [81], where these parameters are the headway distance, speed, and acceleration of the leading vehicle. Contrary to the unstructured data, these approaches significantly reduce the amount of the input and can be handled with simple DNN structures, which profoundly affects the convergence of the agent's performance. For navigating in traffic, i.e., performing merging or lane changing maneuvers, not only the leading vehicle's, but the other surrounding vehicles' states also need to be considered. In a merging scenario, the most crucial information is the relative longitudinal position and speed 2x(dx, dv) of the two vehicles bounding the target gap, as used by [82]. Naturally, this is the absolute minimal representation of such a problem, but in the future, more sophisticated representations would be developed. In highway maneuvering situations, both ego-lane, and neighboring lane vehicles need to be considered, in [54] the authors used the above mentioned 6x(dx, dv) scalar vector is used for the front and rear vehicles in the three interesting lanes. While in [83] the authors extended this information with the occupancy of the neighboring lanes right at the side of the ego-vehicle (See Fig. 9). The same approach can be seen in [55], though extending the number of traced objects to nine. These researches lack lateral information, though, in [54], the lateral positions and speeds are also involved in the input vector resulting in a 6x(dx, dy, dvx, dvy) structure, logically representing longitudinal and lateral distance, and speed differences to the ego, respectively. In a special case of handling unsignalized intersection [84] the authors also used this formulation scheme where the other vehicle's Cartesian coordinates, speed and heading were considered.

III. Scenario-Based Classification of the Approaches

Though this survey focuses on Deep Reinforcement Learning based motion planning research, it is essential to mention that some papers try to solve some subtasks of automated driving through classic reinforcement techniques. One problem of these classic methods, that they can not handle unstructured data, such as images, mid-level radar, or lidar sensing.

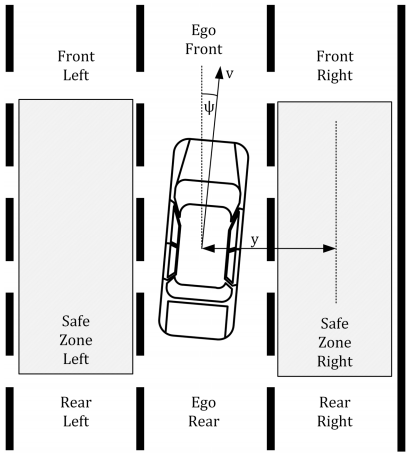


Fig. 9. Environment state on the highway [83].

TABLE I
State Representation Discretization in [85]

Name	Size	Class bounds
$dist_y[m]$	6	{0, 10, 20 ,30, 50, 100, 200}
$dist_x[m]$	10	{-25, -15, -5, -3 , -1, 0, 1, 3, 5, 15, 25}
$pos[m]$	8	{-10, -5, -2, -1, 0, 1, 2, 5, 10}
$\Delta speed[km/h]$	9	{-300, 0, 30, 60, 90, 120, 150, 200, 250, 300}

The other problem comes from the need of maintaining the Q-table for all $(\mathcal{S}, \mathcal{A})$ state-action pairs. This results in space complexity explosion, since the size of the table equals the product of the size of all classes both in state and action. As an example, the Q-learning made in [85] is presented. The authors trained an agent in TORCS, which tries to achieve a policy for the best overtaking maneuver, by taking advantage of the aerodynamic drag. There are only two participants in the scenario, the overtaking vehicle, and the vehicle in front on a long straight track. The state representation contains the longitudinal and lateral distance of the two vehicles and also the the lateral position of the ego-vehicle and the speed difference of the two. The authors discretized this state space to classes of size $(6, 10, 8, 9)$ respectfully (see table I); and used the minimal lateral action set size of 3, where the actions are sweeping $1m$ to the left or right and maintaining lateral position. Together, this problem generates a Q-table with $6 * 10 * 8 * 9 * 3 = 12960$ elements. Though a table of this size can be easily handled nowadays, it is easy to imagine that with more complex problems with more vehicles, more sensors, complex dynamics, denser state and action representation, the table can grow to enormous size. A possible reduction is the utilization of the Multiple-Goal Reinforcement Learning Method and dividing the overall problem to subtasks, as can be seen in [86] for overtaking maneuver. In a latter research, the authors widened the problem and separated the driving problem to the tasks of collision avoidance, target seeking, lane following, Lane choice, speed keeping, and steady steering [87]. To reduce problem size, the authors of [88] used strategic-level decisions to set movement targets for the vehicles concerning the surrounding ones, and left

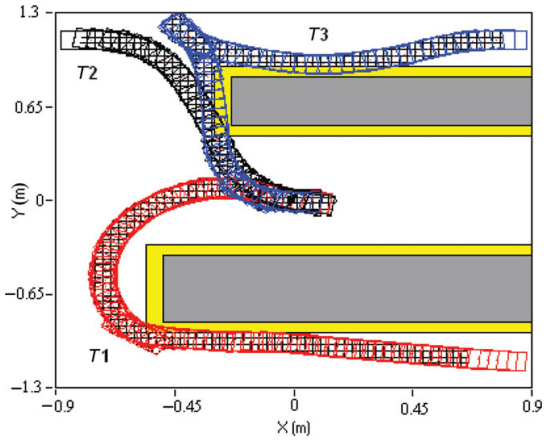


Fig. 10. Path planning results from [89].

Fig. 11. Ramp merge: (a) simulated scenario and (b) real-world location (source: [82]).

the low-level control to classic solutions, which significantly reduced the action space.

An other interesting example of classic Q-learning is described in [89] where the authors designed an agent for the path planning problem of a ground vehicle considering obstacles with Ackermann steering by using (v, x, y, θ) (speed, positions and heading) as state representation, and used reinforcement learning as an optimizer (See Fig. 10).

Though one would expect that machine learning could give an overall end-to-end solution to automated driving, the study of the recent literature shows that Reinforcement Learning research can give answers to certain sub-tasks of this problem. The papers in recent years can be organized around these problems, where a well-dedicated situation or scenario is chosen and examined whether a self-learning agent can solve it. These problem statements vary in complexity. As mentioned earlier, the complexity of reinforcement learning, and thus training time, is greatly influenced by the complexity of the problem chosen, the nature of the action space, and the timeliness and proper formulation of rewards. The simplest problems, such as lane-keeping or vehicle following, can generally be traced back to simple convex optimization or control problems. However, in these cases, the formulation of secondary control goals, such as passenger comfort, is more comfortable to articulate. At the other end of the imagined complexity scale, there are problems, like in the case of maneuvering in dense traffic, the efficient fulfillment of the task is hard to formulate, and the agent needs predictive "thinking" to achieve its goals. In the following, these approaches are presented.

A. Car Following

Car following is the simplest task in this survey, where the problem is formulated as follows: There are two participants of the simulation, a leading and the following vehicle, both keeping their lateral positions in a lane, and the following vehicle adjusts its longitudinal speed to keep a safe following distance. The observation space consists of the (v, dv, ds) tuple, representing agent speed, speed difference to the lead, and headway distance. The action is the acceleration command. Reward systems use the collision of the two vehicles as a failure naturally, while the performance of the agent is based on the jerk, TTC (time to collision) [63], or passenger comfort [44]. Another approach is shown in [64], where the performance of the car following agent is evaluated against real-world measurement to achieve human-like behavior.

B. Lane Keeping

Lane-keeping or trajectory following is still a simple control task, but contrary to car following, this problem focuses on lateral control. The observation space in these studies us two different approaches: One is the "ground truth" lateral position and angle of the vehicle in lane [22], [73], [90], while the second is the image of a front-facing camera [67], [70], [72]. Naturally, for image-based control, the agents use external simulators, TORCS, and GAZEBO/ROS in these cases. Reward systems almost always consider the distance from the center-line of the lane as an immediate reward. It is important to mention that these agents hardly consider vehicle dynamics, and surprisingly does not focus on joined longitudinal control.

C. Merging

The ramp merge problem deals with the on-ramp highway scenario (see Fig. 11), where the ego vehicle needs to find the acceptable gap between two vehicles to get on the highway. In the simplest approach, it is eligible to learn the longitudinal control, where the agent reaches this position, as can be seen in [19], [58], [91]. Other papers, like [82] use full steering and acceleration control. In [58], the actions control the longitudinal movement of the vehicle accelerate and decelerate, and while executing these actions, the ego vehicle keeps its lane. Actions "lane change left" as well as "lane change right" imply lateral movement. Only a single action is executed at a time, and actions are executed in their entirety, the vehicle is not able to prematurely abort an action.

An exciting addition can be examined in [19], where the surrounding vehicles act differently, as there are cooperative and non-cooperative drivers among them. They trained their agents with the knowledge about cooperative behavior, and

also compared the results with three differently built MTCS planners. Full information MCTS naturally outperforms RL, though they are computationally expensive. The authors used a curriculum learning approach to train the agent by gradually increasing traffic density. As they stated: "When training an RL agent in dense traffic directly, the policy converged to a suboptimal solution which is to stay still in the merge lane and does not leverage the cooperativeness of other drivers. Such a policy avoids collisions but fails at achieving the maneuver."

The most detailed description for this problem is given by [82], where "the driving environment is trained as an LSTM architecture to incorporate the influence of historical and interactive driving behaviors on the action selection. The Deep Q-learning process takes the internal state from LSTM as the input to the Q-function approximator, using it for the action selection based on more past information. The Q-network parameters are updated with an experience replay, and a second target Q-network is used to relieve the problems of local optima and instability." With this approach, the researchers try to mix the possibilities from behavior prediction and learning, simultaneously achieving better performance.

Multiagent merging scenarios usually use only longitudinal control, to find the safe gap, and leave the lateral movements to an underlying control scheme. From this aspect, on-ramp merging and some intersection passing problems have a lot in common. Therefore both on-ramp and intersection related MARL is discussed in this section.

The first example comes from [92], where the scenario is a roundabout, which is topologically similar to the on/off-ramp problems. The research used homogeneous non-communicating agents, with parameter sharing A3C learner. The observation space is both ego-vehicle state and a birds'eye view grid of the scene, representing the path-to-follow, the topology, and the dynamic objects in three channels. Naturally, this setup needs a heterogeneous-input NN, CNN for the grid, and DNN for the state values, and three discrete choices: accelerate, maintain speed and brake. An interesting comparison can be found in [93], where the effect of delay and the single/multi agent approaches are evaluated through multiple scenarios, from which one is an unsignalized intersection, with four agents turning to the left. Delay awareness was handled by expanding the POMDP with the set of previous actions. The research applied continuous longitudinal acceleration command, and the multiagent DDPG (MADDPG) from [94], with centralized critic and decentralized actor.

In [95], a multilane intersection was examined, where besides the longitudinal discrete action, a lane change action was also applied. The researched used "COIN" from [96], a parameter sharing table-based immediate reward reinforcement learning approach. Though, as mentioned before, such a problem is too complicated for a tabled-Q learner, hence the authors used the KNN technique for functional approximation to deal with occasional rare states whose actions have not all been trained. Another tabled-Q is presented in [97] for a two-agent merge scenario represented with a cell-transition model. This representation is small enough to solve, though its expansion and generalization are not possible.

Among the merge scenarios, the most complicated one is the double-merge. Two multilane highway join and separate afterward, having agents arriving from both entrances and also leaving on both exits. The first examination for this problem is an example of the proposed CM3 algorithm from [98], where two AI-controlled agents perform this merge among other surrounding vehicles in the SUMO simulator. As the double-merge problem is quite dangerous, it is hard to solve it with simple RL techniques. In [99], a PG based learner provides longitudinal and lateral desired targets, though a rule-based supervisory system ensures its safety.

D. Driving in Traffic

The most complicated scenario examined in the recent papers are those where the autonomous agent drives in traffic. Naturally, this task is also scalable by the topology of the network, the amount and behavior of the surrounding vehicles, the application of traffic rules, and many other properties. Therefore almost all of the current solutions deal with highway driving, where the scenario lacks intersections, pedestrians, and the traffic flow in one direction in all lanes. Sub-tasks of this scenario were examined in the previous sections, such as lane-keeping, or car following. In the following, two types of highway driving will be presented. First, the hierarchical approaches are outlined, where the agents act on the behavioral layer, making decisions about lane changing or overtaking and performs these actions with an underlying controller using classic control approaches. Secondly, end-to-end solutions are presented, where the agents directly control the vehicle by steering and acceleration. As the problem gets more complicated, it is important to mention that the agents trained this would only be able to solve the type of situations that it is exposed to in the simulations. It is, therefore, crucial that the design of the simulated traffic environment covers the intended case [65].

Making decisions on the behavioral layer consists of at least three discrete actions: Keeping current lane, Change to the left, and Change to the right, as can be seen in [55]. In this paper, the authors used the ground truth information about the ego vehicle's speed and lane position, and the relative position and speed of the eight surrounding vehicles as the observation space. They trained and tested the agents in three categories of observation noises: noise-free, mid-level noise (%5), and high-level noise (%15), and showed that the training environments with higher noises resulted in more robust and reliable performance, also outperforming the rule-based MOBIL model, by using DQN with a DNN of $64, 128, 128, 64$ hidden layers with $tanh$ activation. In a quite similar environment and observation space, [65] used a widened set of actions to perform the lane changing with previous accelerations or target gap approaching, resulting in six different actions as can be seen in table II. They also achieved the result that the DQN agent - using two convolutional and one dense layer - performed on par with, or better than, a reference model based on the IDM [41]. and MOBIL [40] model. In the other publication from the same author [100], the action space is changed slightly by changing

TABLE II
ACTION SPACE IN [65]

a_1	Stay in current lane, keep current speed
a_2	Stay in current lane, accelerate with $-2m/s^2$
a_3	Stay in current lane, accelerate with $-9m/s^2$
a_4	Stay in current lane, accelerate with $2m/s^2$
a_5	Change lanes to the left, keep current speed
a_6	Change lanes to the right, keep current speed

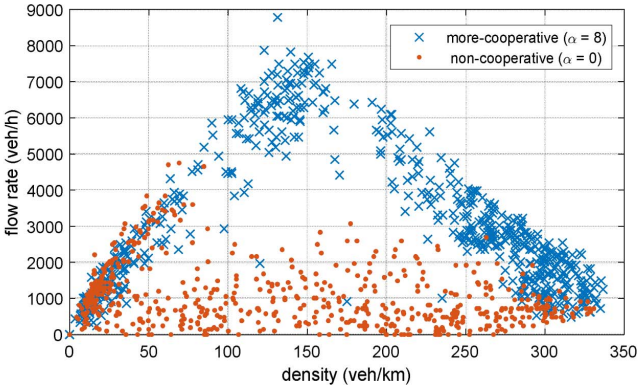


Fig. 12. Flow-density relations detected by the virtual loops under different strategies (source: [80]).

the acceleration commands to increasing and decreasing the ACC set-points and let the underlying controller perform these actions.

In [81], a two-lane scenario is considered to distribute the hierarchical decisions further. First, a DQN makes a binary decision about "to or not to change lane", and afterward, the other Q network is responsible for the longitudinal acceleration, based on the previous decision. Hence the second layer, integrated with classic control modules (e.g., Pure Pursuit Control), outputs appropriate control actions for adjusting its position. In [60], the above mentioned two-lane scenario is considered, though the authors used an actor-critic like learning agent.

An interesting question in automated driving is the cooperative behavior of the trained agent. In [80] the authors considered a three-lane highway with a lane-based grid representation as observation space and a simple tuple of four for action space left, right, speedup, none, and used the reward function to achieve cooperative and non-cooperative behaviors. Not only the classic performance indicators of the ego vehicle is considered in the reward function, but also the speed of the surrounding traffic, which is naturally affected by the behavior of the agent. The underlying network uses two convolutional layers with 16 filters of patch size (2,2) and RELU activation, and two dense layers with 500 neurons each. To evaluate the effects of the cooperative behavior, the authors collected traffic data by virtual loops in the simulation and visualized the performance of the resulting traffic in the classic flow-density diagram (see Fig. 12.) It is shown that the cooperative behavior results in higher traffic flow, hence better highway capacity and lower overall travel time.

The realism of the models could still differentiate end-to-end solutions. For example, in [57], instead of using the nonholonomic Ackermann steering geometry, the authors use a holonomic robot model for the action space, which highly reduces the complexity of the control problem. Their actions are Acceleration, Deceleration, Change lane to the left, Change lane to the right, and Take no action, where the first two apply maximal acceleration and deceleration, while the two lane-changing actions simply use constant speed lateral movements. They use Dueling DQN and prioritized experience replay with a grid-based observation model. A similar control method and nonholonomic kinematics is used in [54]. The importance of this research is that it considers safety aspects during the learning process. By using an MPC like safety check, the agent avoids actions that lead to a collision, which makes the training faster and more robust.

Using nonholonomic kinematics needs acceleration and steering commands. In [59], [83], the authors used a continuous observation space of the structured information of the surrounding vehicles and Policy-gradient RL structure to achieve end-to-end driving. Since the utilized method has discrete action-space, the steering and acceleration command needed to be quantized. The complexity of driving in traffic with an end-to-end solution can be well examined by the number of training episodes needed by the agent. While in simple lane-keeping scenarios, the agents finished the task in few hundreds of episodes, the agent used for these problems needed 300'000.

Some papers propose a multiagent approach to the "navigating in traffic" scenario also. In [103], the authors used a simple discrete, three-lane highway model, with simple choices, showing how the vehicle trained in a single agent approach fails, when placed in a multiagent environment and must deal with agents with the same policy as itself. Though it is also shown, the single agent is a good starting network to begin training in MARL setup.

As discussed earlier, centralized control could be a solution, though it has an exponential growth in terms of complexity, as the agent number grows. In [104], [105] the authors propose the utilization of the so-called Coordination Graph (CG) technique, which decomposes the global payoff function into a linear combination of local payoff functions. As an example, the I-DCG and P-DCG, an identity-based and a position-based coordination graph separation is shown, where the edges of the graph only deal with the cartesian product of the corresponding agents' actions. In [106], the authors seek answers to the same question, using MIT-Deeptraffic [110], a microscopic, strategic level simulator, with a total of 20 cars inside the environment, for which the intelligent control of up to 11 vehicles is allowed. The remaining cars choose their actions randomly. Two scenarios are compared: A single traffic agent's model is applied to multiple agents (transfer learning strategy) and the pure MARL approach.

In [107], the authors propose a periodic parameter sharing structure, where the agents share parameters periodically, but individual policies, which probably comes from the same intuition as the dueling DQN. In their example, two agents perform a cooperative static obstacle avoidance. This work uses mixed grid and ego state observation, hence utilizing

TABLE III
SINGLE AGENT APPROACHES FOR MOTION PLANNING IN DIFFERENT SCENARIOS

Research	Scenario, Model	Observation[a]	Action[b]	Rewarding[c]	RL agent Network
Zhu et al. [64][63]	Car following kinematic	Continuous (v, d^{head}, dv)	Continuous (a)	dis, spe	DDPG DNN
Folkers et al. [78]	Parking navigation kinematic	Grid + Contionuous (x, y, Θ)	Continuous (a, st')	succ, head	PPO CNN, DNN
Lee et al. [73]	Lane keeping lateral	Continuous Lidar-like track sensor	Discrete (st)	coll, lat	DQN
Wolf et al. [67]	Lane keeping lateral, Gazebo	Front Cam (48*27)	Discrete (st)	lat	DQN CNN
Li et al. [70]	Lane keeping lateral, TORCS	Front Cam (MTL) (y, Θ, v)	Continuous (st)	lat, head	DDPG CNN/DNN
Xia et al. [101]	Lane keeping lateral, TORCS	Continuous (y, v, Θ), track info	Discrete (st)	trav	DQN (DQFE)
Feher et al. [56]	Lane keeping kinematic	Continuous (y, v, Θ)	Discrete (a, st)	lat, head	DQN DNN
Xu et al. [72]	Lane keeping TORCS/real data	Front Cam segmented	Discrete (a, st)	coll, spe, head, lat	A3C CNN
Kotyan et al. [71]	Lane keeping OpenAI CarRacing-v0	Bird's eye image difference	Both Disc./Cont. (a, br, st)	time, trav	DQN, DDPG, A3C / CNN
Jaritz et al. [68]	Lane keeping WRC6	Front Cam (84x84x3)	Contiuous (th, st, br, hb)	spe, lat, head	A3C CNN LSTM
Qiao et al. [18]	Intersection longitudinal	Continuous (TTC, v, v^{des}, geo)	1:Discrete $(WAIT, GO)$ 2:Continuous (a)	succ, time, coll, dis	1: DQN 2:DDPG DNN
Isele et al. [102]	Intersection Longitudinal, SUMO	Grid (v, Θ)	Discrete 1:$(WAIT, GO)$ 2:(a)	coll, succ, time	DQN
Bouton et al. [84]	Intersection Longitudinal	Continuous (x, y, v) ego + 2 surr	Discrete (a)	coll, succ	DQN DNN
Ma et al. [22]	Lanechange maneouver kinematic	Continuous (x, y, v, Θ)	Continuous (st, a)	spe, acc, ste, head	TRPO, RARL, NFSP/ DNN
Wang et al. [61]	Lanechange maneouver kinematic	Continuous $(x, y, v, a, \Theta, id, w, c)$	Discrete a_{yaw}	a_{yaw}, time	DQN DNN
An et al. [48]	Lanechange maneouver AirSim	Continuous (x, y, v, Θ) host, target	Continuous (th, st)	coll, succ, lane, spe	DDPG DNN
Shi et al. [81]	Lane change, Car Following	Continuous (x, y, v, a) ego + 4 surr.	1: Discrete (LC) 2: (a)	dis, spe	DQN DNN
Wang et al. [91]	Highway merging Longitudinal kinematic	Continuous (x, v) ego + 2 gap.	Discrete (a)	acc, dis, spe	DQN DNN
Bouton et al. [19]	Highway merging Longitudinal kinematic	Continuous (x, v) ego + 4 surr.	Discrete $(\Delta a\{5\} a\{2\})$	coll, succ, time	DQN DNN
Kaushik et al. [20]	Overtaking TORCS	Ego state, Lidar	Continuous (st, a, br)	lat, spe, head, coll, over	DDPG DNN
Wang et al. [82]	Highway merging kinematic	Continuous (x, y, v, Θ) (v, x) 2 closest agents	Discrete (a, st)	acc,ste,dis, spe	DQN DNN, LSTM
Ronecker et al. [62]	Highway Strategic	Grid $(speed)$	LC target (x, y)	spe, succ, jerk	DQN CNN
Alizadeh et al. [55]	Highway Strategic	Continuous (x, y, v) ego + 8 surr.	Discrete strat. (LLC, RLC, FOL)	spe, lc, coll, dis, succ	DQN DNN
Ye et al. [44]	Highway Strategic Vissim, kinematic	Continuous (x, y, v) ego + 6 surr	Discrete strat. (LLC, RLC, FOL)	coll, spe, jerk	DDPG DNN
Wang et al. [80]	Highway Strategic microscopic	Lane based grid	Discrete strat. (LLC, RLC, ACC, NA)	spe, lc, q	DQN CNN
Wang et al. [49]	Highway Strategic Udacity	Lane based grid	Discrete strat. (LLC, RLC, FOL)	spe, lc, coll	DQN CNN
Hoel et al. [65]	Highway Strategic	Continuous (x, y, v) ego + 8 surr.	Discrete strat. (LLC, RLC, FOL) (LLC, RLC, ACC $\{4\}$)	spe, lc, coll, clo	DDQN CNN, DNN
Xu et al. [60]	Highway Strategic	Continuous (id, v, v^{des}) (v, x) 4 closest agents	Discrete strat. (LLC, RLC, FOL)	spe, coll, smo	MO-RL -
Wolf et al. [58]	Highway Strategic SUMO	Continuous (x, y, v) ego + 8 surr.	Discrete strat. (LLC, RLC, NA, ACC, DEC)	coll, kright, dis, sty	DQN DNN
Bai et al. [57]	Highway Strategic	Lane based grid (speed)	Discrete strat. (LLC, RLC, NA, ACC, DEC)	spe, dis, lc, coll, over	DQN, DDQN CNN
Nageshrao et al. [54]	Highway Strategic	Continuous (x, y, v) ego + 6 surr.	Discrete strat. (LLC, RLC, NA, ACC(3), DEC)	dis, spe	DDQN DNN
Hoel et al. [100]	Highway Strategic	Continuous (x, y, v) ego + N surr.	Discrete strat. (LLC, RLC, ACCSP)	time, lc, spe, succ	AlphaGo Zero CNN
Aradi et al. [59]	Highway kinematic	Continuous (x, y, v) ego + 8 surr.	Discrete (a, st)	krightt, lat, spe	PG DNN
Saxena et al. [10]	Highway kinematic	Lane based grid (speed, heading)	Continuous (a, st)	lat, spe, jerk, time	PPO CNN

TABLE IV
MULTIAGENT APPROACHES FOR MOTION PLANNING IN DIFFERENT SCENARIOS

Research	Scenario Model	Observation Space[a]	Action Space[b]	Rewarding[d]	MARL Approach Network Topology
Chen et al. [93]	Intersection Longitudinal control Kinematic	Continuous (x, v) all agents	Continuous (a)	Joined	MA-DDPG [94]
Bacchiani et al. [92]	Intersection (Roundabout) Longitudinal control Kinematic	Grid sequence, Continuous $(v, v^{des}, t, d^{goal})$	Discrete $(a\{3\})$	Individual	Parameter sharing A3C CNN+Dense
Kalantari et al. [95]	Intersection Strategic decisions SUMO Microscopic	Discrete (x, y, v, TTC, d^{head})	Discrete $(a\{3\}, LC_y\{3\}, Tn\{2\})$	Joined	Parameter Sharing Tabled Q
Schester et al. [97]	Highway, on-ramp Longitudinal Merging Celled	Continuous (x, d^{gap}, dv, TTP)	Discrete $(steps\{3\})$	Centralized	Centralized Control Tabled Q
Rădulescu et al. [103]	Highway Navigation, strategic Celled	Lane based grid (occopancy and speed)	Discrete $(a\{5\}, Dir\{3\})$	Individual	Parameter sharing DQN Dense
Shalev-Shwartz et al. [99]	Highway Double merge Microscopic	Continuous (x, y, v, Θ) all agents and target lane for ego	Discrete Desired (v, LC_y)	Individual	Supervisory control PG
Yang et al. [98]	Highway Double merge SUMO Microscopic	Lane based grid (occupancy and speed)	Strategic (ACC, DEC, LLC, RLC, NA)	Individual	CM3
Yu et al. [104] [105]	Highway Navigation, strategic Microscopic	Continuous (x, y, v) ego and 4 closest agents	Strategic Lane choice	Individual	I-DCG, P-DCG DQN
Schutera et al. [106]	Highway Navigation, strategic MIT deeptraffic Microscopic	Lane based grid (occupancy and speed)	Strategic (ACC, DEC, LLC, RLC, NA)	Joined	Parameter sharing DQN
Cicek et al. [107]	Highway Navigation Kinematic	Occupancy Grid Continuous Ego state	Discrete (a, st)	Joined	Parameter sharing DQN (CNN+Dense), LSTM
Kaushik et al. [108]	Highway Navigation TORCS Dynamic	Continuous ego state, LIDAR	Continuous (a, br, st)	Individual	Parameter sharing DDPG
Bhattacharyya et al. [109]	Highway Navigation NGSIM data Kinematic	Continuous states, LIDAR	Continuous (a, st)	Not applicable	PS-GAIL TRPO RNN, Dense

a CNN/DNN network. The results are compared to pure parameter sharing and full individual training, showing that in this particular case, this golden mean performs better than the original agents.

There are also research groups, whose attention turns from single RL to MARL. In [20], the authors searched the solution for competitive overtaking in the TORCS environment, and afterward, extended the research to multiagent in [108]. They use one simple parameter sharing DDPG, though train the agent for two distinct tasks. The first rewards only lane following, though the second also rewards race position. The "task" is injected into the observation space as a binary information to learn the same strategy with one agent. Hence, based on the command received in the observation vector, the same agent acts competitively or cooperatively.

Finally, a not clearly RL, but imitation learning is proposed by [109], where Generative Adversarial Imitation Learning (GAIL) was extended with Parameter Sharing Trust Region Policy Optimization (PS-TRPO) [35] to enable imitation learning in the multi-agent context, calling the new PS-GAIL. For this framework, the agent needs demonstration, which was extracted from the Next-Generation Simulation (NGSIM) dataset [111].

E. Literature Summary

The summarization of the single-agent approaches are given in table III, providing information on the scenario, the utilized model or simulation environment, the observation and action spaces, the elements considered in the reward function, and also the type of RL agent, and the neural network.

Table IV presents the researches with multiagent (MARL) approaches, with similar columns, except for rewarding, where the distribution of the reward among agents is outlined. There could be further criteria for classifying MARL, e.g., heterogeneity of the agent, communication between the agents, and cooperativeness. Though this area, the agents involved are all homogeneous, use some kind parameter sharing or centralized control, with one exception in [93] using MA-DDPG from [94] which is a centralized critic, decentralized actor scheme. Regarding competitiveness, the nature of reward (joined, or individual) does not determine the competitiveness, as the rewards given to agents do not add up to zero. Hence all research deals with cooperative problem setups, again with one exception in [108], where the scenario is racing, and race position is part of the reward, which establishes competition among the agents. And finally, there is modeled communications between the agents in some papers through dedicated

short-range communications (DSRC). However, these only serve the purpose of establishing the observation space for the individual agents. In terms of learning and MARL, the agents don't establish or develop communication to form joined strategies. Though the elements and values used in the MDP formulation are detailed in the paper, this section summarizes

[a] Observation space elements: (x, y) are longitudinal and lateral position. $(d^{goal}, d^{head}, d^{gap})$ are distances to goal, leading vehicle, and the target gap, respectively. (T, TTC, TTP) are Elapsed time, Time-to-collision and Time-to-position. (v_x, v^{des}, dv) are for longitudinal speed, desired speed, and speed difference. Θ denote the heading of the vehicle. (id, w, c, geo) are for lane id, width, curvature, and geometry.

[b] Action Space elements: a-acceleration, th-throttle, st-steering, st' -steering speed, a_{yaw} - yaw rate, br - brake, hb - handbrake stands for longitudinal acceleration, LC_y and Dir are lateral, $steps$ are longitudinal discrete steps, Tn is action for turning in intersection. $\{N\}$ shows the size of the discrete action set, if known. For strategic decisions: (WAIT, GO) - halts/starts the agent in intersection, ACC - acceleration, ACCSP - ACC setpoint, DEC - deceleration, (LLC, RLC) - Lane change left or right, NA - No action (keep state), FOL - stay in lane and use car following rules.

[c] For single-agent rewards, the followings were considered: spe - speed, lc - lane chage (penalty), q - traffic flow, coll- collision, clo - unsafe distance (closing), time - time elapsed (step cost), succ - success/failure, lane - lane choice, acc - acceleration, trav - distance traveled, head - heading, lat - lateral distance, kright - keep right, sty - style, a_{yaw} - yaw rate, jerk - jerk, over - overtaking.

[d] Multiagent rewards are categorized as follows: Centralized rewards are given, when there is one centralized controller for the multiagent task. "Joined" means that the agent receives the reward based on the performance of every agent, while "Individual" means the agent receives reward only for its performance. This scheme does not apply to [109], since it uses imitation learning.

IV. Future Challenges

The recent achievements on the field showed that different deep reinforcement learning techniques could be effectively used for different levels of autonomous vehicles' motion planning problems, though many questions remain unanswered. The main advantage of these methods is that they can handle unstructured data such as raw or slightly pre-processed radar or camera-based image information.

One of the main benefits of using deep neural networks trained by a reinforcement learning agent in motion planning is the relatively low computational requirements of the trained network. Though this property needs a vast amount of trials in the learning phase to gain enough experience, as mentioned before, for simple convex optimization problems, the convergence of the process is fast. However, for complex scenarios, the training can quickly reach millions of steps, meaning that one setup of hyper-parameters or reward hypothesis can last hours or even days. Since complicated reinforcement learning tasks need continuous iteration on the environment design, network structure, reward scheme, or even the used algorithm itself, designing such a system is a time-consuming project. Besides the appropriate result analysis and inference, the evaluation time highly depends on the computational capacities assigned. On this basis, it is not a surprise that most papers nowadays deal with minor subtasks of the motion planning problem, and the most complex scenarios, such as navigating in urban traffic, can not be found in the literature. As many heuristics, RL itself has its trade-off between performance and resource neediness. The performance in vehicle control is not only journey time, average speed, or passenger comfort, but primarily safety and robustness. Reinforcement Learning has many challenges in these two fields. In the following, these two major problems will be outlined.

A. Safety

Using neural networks and deep learning techniques as universal function-approximators in automotive systems poses several questions. For example, what is the amount of training data needed for safe driving? [112]. As stated in [113], function development for automotive applications realized in electronic control units (ECUs) is subject to proprietary OEM norms and several international standards, such as Automotive SPICE (Software Process Improvement and Capability Determination) [114] and ISO 26262 [115]. However, these standards are still far from addressing deep learning with dedicated statements, since verification and validation is not a solved issue in this domain. Some papers deal with these issues by using an underlying safety layer, which verifies the safety of a planned trajectory before the vehicle control system executes it. However, full functional safety coverage can not be guaranteed in complex scenarios this way.

The main goal of reinforcement learning is to maximize the long-term reward statistically. Still, the primary goal is the prevention of accidents for vehicle control tasks. Since RL does not necessarily prevent the use of actions that cause large negative rewards, other methods are needed to handle the risks. The literature deals with security and risks in many forms, for which [116] provides an excellent summary. Two main directions can be distinguished in this area. One group of solutions includes methods using the optimization criterion, while the other group contains algorithms that modify the exploration process. One also has several options for modifying optimization criteria. The first is the worst-case criterion. It addresses the problems caused by the uncertainties arising from the stochastic volatility of the system and the parameter uncertainties by considering the worst-case scenarios. The second option is the risk-sensitive criterion. In this case, a scalar parameter is added to the loss function, the so-called risk sensitivity parameter to control the level of risk. Finally, it is possible to use constrained MDP, where the standard MDP tuple is extended with a constraint set, which the policy function must fulfill.

Modifying the exploration process is an option, contrary to the classic exploration strategy, which assumes that the agent learns everything from scratch. With vehicle control applications, this often leads to catastrophic situations. Furthermore,

completely unintentional exploration strategies waste a lot of time exploring the irrelevant areas of the underlying state space, which is especially important in large and continuous state spaces. There are two main directions. One directs the exploration process by applying external intelligence, while the other is using risk estimation. The first case uses a finite set of demonstrations by a human demonstrator, which can then be further optimized, creating a preliminary value function. This approach resembles imitation learning. The demonstrator can also guide the exploration online by showing the interesting, or hazardous parts of the state space. And finally hard constraints can be satisfied by a supervisory control scheme, as seen in [99]. There are already some researches dealing with RL ensuring safer driving. In [117], the authors use DDPG algorithm combined with the artificial potential field to develop a safe lane following and collision avoidance algorithm. An inspiring approach is presented in [118] article, where the authors also train a mobile robot for collision avoidance, combining exploration modification and curriculum learning methods, starting with low-speed maneuvers, and continuously raising the hardness of the task. For this purpose, they proposed an uncertainty-dependent cost function to estimate the risk of collision. The training process was demonstrated both in a simulator and on a real robot.

The authors of [119] provide an example of safe highway driving, increasing security in two ways. On the one hand, a module for learning safety patterns is created, which works from preliminary driving data, and uses a prediction in the distant future. Also, a heuristic hand-crafted safety module has been developed based on common driving practice and ensuring a minimum following distance. They demonstrate the results in a simulation with varying traffic density.

In [120], a so-called "Parallel Constrained Policy Optimization" method is presented, which is demonstrated in two scenarios. The general actor-critic structure is extended by an approximation of a risk function with a third neural network. The results are presented in a lane following and intersection crossing simulation.

Overall, the theory of safe RL is a dynamically evolving field. In addition to the survey article quoted above, the interested reader can find theoretical details of each solution in [121]. From the point of view of vehicle control, the importance of the topic is unquestionable, not only for safety but also for the reduction of the state and action space. One of the big problems with training and validation is choosing the problematic, so-called corner cases from a vast number of irrelevant situations.

B. Sim2Real

By examining the observation element of the recent articles, it can be stated that most researches ignore complex sensor models. Some papers use "ground truth" environment representations or "ideal" sensor models, and only a few articles utilize sensor noise. On the one hand, transferring the knowledge acquired from ideal observations to real-world application poses several feasibility questions [122], on the other hand, using noisy or erroneous models could lead to actually more robust agents, as stated in [55].

The same applies to the environment, which can be examined best amongst the group of highway learners, where the road topology is almost always fixed, and the surrounding vehicles' behavior is limited. Validation of these agents is usually made in the same environment setup, which contradicts the basic techniques of machine learning, where the training and validation scenarios should differ in some aspects. As a reinforcement learning agent can generally act well in the situations that are close to those it has experience with, it is crucial to focus on developing more realistic and diverse environments, including the modeling level of any interacting traffic participant to achieve such agents that are easily transferable to real-world applications. This applies to vehicle dynamics, where more diverse and more realistic modeling would be needed. Naturally, these improvements increase the numerical complexity of the environment model, which is one of the main issues in these applications.

Among the research evaluated in this review, all problems were trained in a simulated environment. There exists only one exception in [123], where the authors taught the agent for lane following on a real vehicle using a continuous, model-free deep reinforcement learning algorithm DDPG, with extensions, that enabled a significant decrease of episodes used for training.

There are many reasons for using simulation as a training tool for RL in this field. The first is that one can afford many more samples since the simulations can be significantly faster and cheaper (fuel, personnel, equipment costs) than the real experiments. The second is safety since it can not be guaranteed in real traffic for trial-and-error-like learning of RL. Naturally, using simulations has its cons in RL also. The first comes from modeling and identification. Lots of simulators are under-modeled, usually to keep the trade-off for computational resources. The discrepancy regarding the real world can come either from the side of observations or vehicle dynamics. Sensors can be too precise, too reliable, or provide ground truth on the full state, which can not be achieved in real-world scenarios. Or, even the contrary, can lack detail, which is usually the case of rendered visual environments providing camera information. As a result, policies that are learned in simulation do not transfer to the real world, often called the "reality gap" or the "sim2real gap". The handling of such problems is hard, even when the underlying MDP assumption stands. Though when the environment becomes partially observable, or multiple active agents appear whose actions can not be predicted, this gap widens. In a real traffic simulation, it is almost (if not entirely) impossible to cover all possible circumstances. Table V summarizes the key pros and cons of using simulation. As the reality gap is wide, real vehicle testing of the developed algorithms can not guarantee safety. Moreover, many feasibility questions arise, such as costs, automation, equipment, and test site. These together results in that most of the researches stay on simulation level, and only a few can provide real-world application, all of them with some limitations. In [60], the decisions of the lane choice algorithm are shown on a two-lane highway without providing

TABLE V
PROS AND CONS OF USING SIMULATION FOR TRAINING RL

Pros	Cons
Costs	Under-modeling
Training Speed	Observations/sensor models
Ensured Safety	Dynamics
Simple dynamics can be handled	Multiagent behaviors

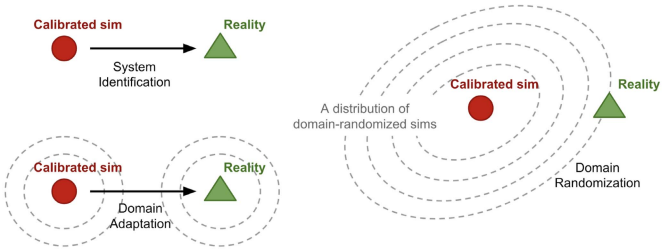


Fig. 13. Conceptual illustrations of three approaches for sim2real transfer (source: [126]).

full-control to the algorithm. The parking navigation algorithm of [78] provides an example in a closed parking place, and the lane-change maneuver developed in [22] is evaluated in a closed test track.

Generally, there are three approaches to close the reality gap:

- System identification, trying to match the simulation to reality.
- Domain adaptation, aiming at learning from a source data distribution (simulation) a well performing model on a different (but related) target data distribution (reality). [124]
- Domain randomization, aiming at learning on a very randomized environment (simulation), which (probably) covers the target (reality), making the agent robust. [125]

These three concepts are illustrated in Fig. 13 The trade-off between the fully modeled system and feasibility was discussed before, hence system identification is not outlined here. During Domain adaptation, one tries to find the transfer technique between the simulated and the real representations. As an example, for image sequences taken from a front-facing camera, this transfer can be solved through the semantically segmented image. In [72] the two domains meet in the middle at the segmented level, while in [127], the authors try to create "realistic" images for training by using generative adversarial nets (GAN) [128]. Naturally, this approach relies on the GAN training data and does not guarantee full coverage.

According to many research, such as [129], RL agents usually overfit to the environments they are trained on, even developing policies, that are totally unusable in the real application. Domain randomization, among increasing robustness, is a kind of generalization or regularization technique. Though as the possible dimensions of the randomization increases, its scalability issues are becoming serious [130]. And on the other hand, regarding [131], too many randomizations imply a conservative policy from the agent. Although most research presented use some randomization (multiple tracks, random initialization or goal, etc.), these are far from covering all possible cases of real driving. Based on the reasons above, sim2real is one of the critical research problems of the field in the future.

Overall it can be said that many problems need to be solved in this field, such as the detail of the environment and sensor modeling, the computational requirements, the transferability to real applications, robustness, and validation of the agents. Because of these issues, it can be stated that reinforcement learning is not a sufficient tool for automotive motion planning in itself. Still, it can be very efficient in solving complex optimization tasks by combining with other methods.

ACKNOWLEDGMENT

The research reported in this paper and carried out at the Budapest University of Technology and Economics was supported by the "TKP2020, Institutional Excellence Program" of the National Research Development and Innovation Office in the field of Artificial Intelligence (BME IE-MI-FM TKP2020).

REFERENCES

[1] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: http://arxiv.org/abs/1312.5602
[2] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://www.nature.com/articles/nature14236
[3] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016, doi: 10.1109/TIV.2016.2578706.
[4] H. Bast *et al.*, "Route planning in transportation networks," in *Algorithm Engineering* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2016, pp. 19–80.
[5] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 392–399. [Online]. Available: http://ieeexplore.ieee.org/document/6957722/
[6] J. Wiest, M. Hoffken, U. Kresel, and K. Dietmayer, "Probabilistic trajectory prediction with Gaussian mixture models," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2012, pp. 141–146. [Online]. Available: http://ieeexplore.ieee.org/document/6232277/
[7] Y. Dou, F. Yan, and D. Feng, "Lane changing prediction at highway lane drops using support vector machine and artificial neural network classifiers," in *Proc. IEEE Int. Conf. Adv. Intell. Mechatronics (AIM)*, Jul. 2016, pp. 901–906. [Online]. Available: http://ieeexplore.ieee.org/document/7576883/
[8] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. 20th Annu. Symp. Found. Comput. Sci. (SFCS)*, Oct. 1979, pp. 421–427. [Online]. Available: http://ieeexplore.ieee.org/document/4568037/
[9] F. Hegedus, T. Becsi, S. Aradi, and G. Galdi, "Hybrid trajectory planning for autonomous vehicles using neural networks," in *Proc. IEEE 18th Int. Symp. Comput. Intell. Informat. (CINTI)*, Nov. 2018, pp. 25–30. [Online]. Available: https://ieeexplore.ieee.org/document/8928220/
[10] D. Mauria Saxena, S. Bae, A. Nakhaei, K. Fujimura, and M. Likhachev, "Driving in dense traffic with model-free reinforcement learning," 2019, *arXiv:1909.06710*. [Online]. Available: http://arxiv.org/abs/1909.06710
[11] Á. Fehér, S. Aradi, F. Hegedüs, T. Bécsi, and P. Gáspár, "Hybrid DDPG approach for vehicle motion planning," in *Proc. 16th Int. Conf. Informat. Control, Autom. Robot.*, 2019, pp. 422–429.
[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2017.
[13] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1957.
[14] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 1–13.

[15] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," 2015, *arXiv:1511.06581*. [Online]. Available: http://arxiv.org/abs/1511.06581

[16] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, 2014, pp. 387–395.

[17] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971

[18] Z. Qiao, K. Muelling, J. M. Dolan, P. Palanisamy, and P. Mudalige, "Automatically generated curriculum based reinforcement learning for autonomous vehicles in urban environment," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1233–1238. [Online]. Available: https://ieeexplore.ieee.org/document/8500603/

[19] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Cooperation-aware reinforcement learning for merging in dense traffic," 2019, *arXiv:1906.11021*. [Online]. Available: http://arxiv.org/abs/1906.11021

[20] M. Kaushik, V. Prasad, K. M. Krishna, and B. Ravindran, "Overtaking maneuvers in simulated highway driving using deep reinforcement learning," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1885–1890. [Online]. Available: https://ieeexplore.ieee.org/document/8500718/

[21] A. Ferdowsi, U. Challita, W. Saad, and N. B. Mandayam, "Robust deep reinforcement learning for security and safety in autonomous vehicle systems," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 307–312.

[22] X. Ma, K. Driggs-Campbell, and M. J. Kochenderfer, "Improved robustness and safety for autonomous vehicle control with adversarial reinforcement learning," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1665–1671.

[23] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, nos. 1–2, pp. 99–134, May 1998. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S000437029800023X

[24] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Auto. Robots*, vol. 8, pp. 345–383, Jun. 2000.

[25] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Auto. Agents Multi-Agent Syst.*, vol. 33, no. 6, pp. 750–797, Nov. 2019.

[26] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Math. Oper. Res.*, vol. 27, no. 4, pp. 819–840, Nov. 2002.

[27] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*, vol. 1. Cham, Switzerland: Springer, 2016.

[28] M. Lanctot *et al.*, "A unified game-theoretic approach to multiagent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4190–4203.

[29] Y. Shoham, R. Powers, and T. Grenager, "If multi-agent learning is the answer, what is the question?" *Artif. Intell.*, vol. 171, no. 7, pp. 365–377, May 2007.

[30] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Auto. Agents Multi-Agent Syst.*, vol. 11, no. 3, pp. 387–434, Nov. 2005.

[31] J. N. Foerster, Y. M. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2137–2145.

[32] E. Pesce and G. Montana, "Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication," in *Machine Learning*. Cham, Switzerland: Springer, 2020, doi: 10.1007/s10994-019-05864-5.

[33] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[34] R. Lowe, J. Foerster, Y. L. Boureau, J. Pineau, and Y. Dauphin, "On the pitfalls of measuring emergent communication," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2019, pp. 693–701.

[35] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems. AAMAS* (Lecture Notes in Computer Science), vol. 10642, G. Sukthankar and J. Rodriguez-Aguilar, Eds. Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-71682-4_5.

[36] I. Lubashevsky and S. Kanemoto, "Scale-free memory model for multi-agent reinforcement learning. Mean field approximation and rock-paper-scissors dynamics," *Eur. Phys. J. B*, vol. 76, no. 1, pp. 69–85, Jul. 2010.

[37] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2015, pp. 1094–1099. [Online]. Available: http://ieeexplore.ieee.org/document/7225830/

[38] P. Polack, F. Altche, B. d'Andrea-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 812–818. [Online]. Available: http://ieeexplore.ieee.org/document/7995816/

[39] C. You, J. Lu, D. Filev, and P. Tsiotras, "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning," *Robot. Auto. Syst.*, vol. 114, pp. 1–18, Apr. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889018302021

[40] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model MOBIL for car-following models," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1999, no. 1, pp. 86–94, Jan. 2007. [Online]. Available: http://journals.sagepub.com/doi/10.3141/1999-10

[41] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.62.1805

[42] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO-simulation of urban mobility," *Int. J. Adv. Syst. Meas.*, vol. 5, no. 3, pp. 128–138, 2012.

[43] M. Fellendorf and P. Vortisch, "Microscopic traffic flow simulator VISSIM," in *Fundamentals of Traffic Simulation* (International Series in Operations Research & Management Science), J. Barceló, Ed. Cham, Switzerland: Springer, 2010, pp. 63–93.

[44] Y. Ye, X. Zhang, and J. Sun, "Automated vehicle's behavior decision making using deep reinforcement learning and high-fidelity simulation environment," *Transp. Res. C, Emerg. Technol.*, vol. 107, pp. 155–170, Oct. 2019, doi: 10.1016/j.trc.2019.08.011.

[45] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner. (2014). *TORCS: The Open Racing Car Simulator*. [Online]. Available: http://www.torcs.org

[46] *CarSIM, Mechanical Simulation Corporation*. Accessed: Jan. 17, 2020. [Online]. Available: https://www.carsim.com/

[47] CarMaker. *IPG Automotive*. Accessed: Jan. 17, 2020. [Online]. Available: https://ipg-automotive.com/products-services/simulation-software/carmaker/

[48] H. An and J.-I. Jung, "Decision-making system for lane change using deep reinforcement learning in connected and automated driving," *Electronics*, vol. 8, no. 5, p. 543, May 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/5/543

[49] J. Wang, Q. Zhang, D. Zhao, and Y. Chen, "Lane change decision-making through deep reinforcement learning with rule-based constraints," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–6. [Online]. Available: http://arxiv.org/abs/1904.00231 and https://ieeexplore.ieee.org/document/8852110/

[50] *Welcome to Udacity's Self-Driving Car Simulator*. Accessed: Jan. 17, 2020. [Online]. Available: https://github.com/udacity/self-driving-car-sim

[51] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–17.

[52] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors*, vol. 19, no. 3, p. 648, Feb. 2019. [Online]. Available: http://www.mdpi.com/1424-8220/19/3/648

[53] K. Kashihara, "Deep q learning for traffic simulation in autonomous driving at a highway junction," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 984–988. [Online]. Available: http://ieeexplore.ieee.org/document/8122738/

[54] S. Nageshrao, H. E. Tseng, and D. Filev, "Autonomous highway driving using deep reinforcement learning," in *Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC)*, Oct. 2019, pp. 2326–2331. [Online]. Available: http://arxiv.org/abs/1904.00035 and https://ieeexplore.ieee.org/document/8914621/

[55] A. Alizadeh, M. Moghadam, Y. Bicer, N. K. Ure, U. Yavas, and C. Kurtulus, "Automated lane change decision making using deep reinforcement learning in dynamic and uncertain highway environment," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 1399–1404. [Online]. Available: https://ieeexplore.ieee.org/document/8917192/

[56] A. Feher, S. Aradi, and T. Becsi, "Q-learning based reinforcement learning approach for lane keeping," in *Proc. IEEE 18th Int. Symp. Comput. Intell. Informat. (CINTI)*, Budapest, Hungary, Nov. 2018, pp. 31–36. [Online]. Available: https://ieeexplore.ieee.org/document/8928230/

[57] Z. Bai, W. Shangguan, B. Cai, and L. Chai, "Deep reinforcement learning based high-level driving behavior decision-making model in heterogeneous traffic," in *Proc. Chin. Control Conf. (CCC)*, Feb. 2019, pp. 8600–8605. [Online]. Available: http://arxiv.org/abs/1902.05772 and https://ieeexplore.ieee.org/document/8866005/

[58] P. Wolf, K. Kurzer, T. Wingert, F. Kuhnt, and J. M. Zollner, "Adaptive behavior generation for autonomous driving using deep reinforcement learning with compact semantic states," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 993–1000. [Online]. Available: https://ieeexplore.ieee.org/document/8500427/

[59] S. Aradi, T. Becsi, and P. Gaspar, "Policy gradient based reinforcement learning approach for autonomous highway driving," in *Proc. IEEE Conf. Control Technol. Appl. (CCTA)*, Aug. 2018, pp. 670–675. [Online]. Available: https://ieeexplore.ieee.org/document/8511514/

[60] X. Xu, L. Zuo, X. Li, L. Qian, J. Ren, and Z. Sun, "A reinforcement learning approach to autonomous decision making of intelligent vehicles on highways," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 10, pp. 3884–3897, Oct. 2020.

[61] P. Wang, C.-Y. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1379–1384.

[62] M. P. Ronecker and Y. Zhu, "Deep Q-Network based decision making for autonomous driving," in *Proc. 3rd Int. Conf. Robot. Autom. Sci. (ICRAS)*, Jun. 2019, pp. 154–160. [Online]. Available: https://ieeexplore.ieee.org/document/8808950/

[63] M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, and R. Ke, "Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving," 2019, *arXiv:1902.00089*. [Online]. Available: http://arxiv.org/abs/1902.00089

[64] M. Zhu, X. Wang, and Y. Wang, "Human-like autonomous car-following model with deep reinforcement learning," *Transp. Res. C, Emerg. Technol.*, vol. 97, pp. 348–368, Dec. 2018.

[65] C.-J. Hoel, K. Wolff, and L. Laine, "Automated speed and lane change decision making using deep reinforcement learning," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2148–2155.

[66] E. Leurent, "A survey of state-action representations for autonomous driving," HAL Id: hal-01908175, 2018.

[67] P. Wolf *et al.*, "Learning how to drive in a real world simulation with deep Q-networks," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 244–250. [Online]. Available: http://ieeexplore.ieee.org/document/7995727/

[68] M. Jaritz, R. de Charette, M. Toromanoff, E. Perot, and F. Nashashibi, "End-to-end race driving with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 2070–2075.

[69] E. Perot, M. Jaritz, M. Toromanoff, and R. De Charette, "End-to-end driving in a realistic racing game with deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 474–475.

[70] D. Li, D. Zhao, Q. Zhang, and Y. Chen, "Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]," *IEEE Comput. Intell. Mag.*, vol. 14, no. 2, pp. 83–98, May 2019.

[71] S. Kotyan, D. V. Vargas, and U. Venkanna, "Self Training autonomous driving agent," in *Proc. 58th Annu. Conf. Soc. Instrum. Control Eng. Jpn. (SICE)*, Sep. 2019, pp. 1456–1461. [Online]. Available: http://arxiv.org/abs/1904.12738 and https://ieeexplore.ieee.org/document/8859883/

[72] N. Xu, B. Tan, and B. Kong, "Autonomous driving in reality with reinforcement learning and image translation," 2018, *arXiv:1801.05299*. [Online]. Available: http://arxiv.org/abs/1801.05299

[73] J. Lee, T. Kim, and H. J. Kim, "Autonomous lane keeping based on approximate Q-learning," in *Proc. 14th Int. Conf. Ubiquitous Robots Ambient Intell. (URAI)*, Jun. 2017, pp. 402–405. [Online]. Available: http://ieeexplore.ieee.org/document/7992762/

[74] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, Jun. 1989. [Online]. Available: http://ieeexplore.ieee.org/document/30720/

[75] S. Thrun *et al.*, "Stanley: The robot that won the DARPA grand challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, 2006, doi: 10.1002/rob.20147.

[76] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based LSTMs," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1179–1184. [Online]. Available: https://ieeexplore.ieee.org/document/8500493/

[77] T. Hegedűs, B. Németh, and P. Gáspár, "Graph-based multi-vehicle overtaking strategy for autonomous vehicles," *IFAC-PapersOnLine*, vol. 52, no. 5, pp. 372–377, 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2405896319306822

[78] A. Folkers, M. Rick, and C. Buskens, "Controlling an autonomous vehicle with deep reinforcement learning," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 2025–2031. [Online]. Available: https://ieeexplore.ieee.org/document/8814124/

[79] J. Esser and M. Schreckenberg, "Microscopic simulation of urban traffic based on cellular automata," *Int. J. Mod. Phys. C*, vol. 8, no. 5, pp. 1025–1036, Oct. 1997. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/S0129183197000904

[80] G. Wang, J. Hu, Z. Li, and L. Li, "Cooperative lane changing via deep reinforcement learning," 2019, *arXiv:1906.08662*. [Online]. Available: http://arxiv.org/abs/1906.08662

[81] T. Shi, P. Wang, X. Cheng, and C.-Y. Chan, "Driving decision and control for autonomous lane change based on deep reinforcement learning," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Auckland, New Zealand, Apr. 2019, pp. 1–6. [Online]. Available: http://arxiv.org/abs/1904.10171

[82] P. Wang and C.-Y. Chan, "Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8317735/

[83] T. Bécsi, S. Aradi, Á. Fehér, J. Szalay, and P. Gáspár, "Highway environment model for reinforcement learning," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 429–434, 2018.

[84] M. Bouton, J. Karlsson, A. Nakhaei, K. Fujimura, M. J. Kochenderfer, and J. Tumova, "Reinforcement learning with probabilistic guarantees for autonomous driving," 2019, *arXiv:1904.07189*. [Online]. Available: http://arxiv.org/abs/1904.07189

[85] D. Loiacono, A. Prete, P. L. Lanzi, and L. Cardamone, "Learning to overtake in TORCS using simple reinforcement learning," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2010, pp. 1–8.

[86] D. C. K. Ngai and N. H. C. Yung, "Automated vehicle overtaking based on a multiple-goal reinforcement learning framework," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Sep. 2007, pp. 818–823. [Online]. Available: http://ieeexplore.ieee.org/document/4357682/

[87] D. C. K. Ngai and N. H. C. Yung, "A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 509–522, Jun. 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5710424/

[88] C. Desjardins and B. Chaib-draa, "Cooperative adaptive cruise control: A reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1248–1260, Dec. 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5876320/

[89] M. Gómez, R. V. González, T. Martínez-Marín, D. Meziat, and S. Sánchez, "Optimal motion planning by reinforcement learning in autonomous mobile vehicles," *Robotica*, vol. 30, no. 2, pp. 159–170, Mar. 2012.

[90] A. El Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," 2016, *arXiv:1612.04340*. [Online]. Available: http://arxiv.org/abs/1612.04340

[91] P. Wang and C.-Y. Chan, "Autonomous ramp merge maneuver based on reinforcement learning with continuous action space," 2018, *arXiv:1803.09203*. [Online]. Available: http://arxiv.org/abs/1803.09203

[92] G. Bacchiani, D. Molinar, and M. Patander, "Microscopic traffic simulation by cooperative multi-agent deep reinforcement learning," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2019, pp. 1547–1555.

[93] B. Chen, M. Xu, Z. Liu, L. Li, and D. Zhao, "Delay-aware multi-agent reinforcement learning for cooperative and competitive environments," 2020, *arXiv:2005.05441*. [Online]. Available: http://arxiv.org/abs/2005.05441

[94] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[95] R. Kalantari, M. Motro, J. Ghosh, and C. Bhat, "A distributed, collective intelligence framework for collision-free navigation through busy intersections," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 1378–1383.

[96] D. Wolpert, "Theory of collective intelligence," in *Collectives and the Design of Complex Systems*, K. Tumer and D. Wolpert, Eds. New York, NY, USA: Springer, 2004, doi: 10.1007/978-1-4419-8909-3_2.

[97] L. Schester and L. E. Ortiz, "Longitudinal position control for highway on-ramp merging: A multi-agent approach to automated driving," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 3461–3468. [Online]. Available: https://ieeexplore.ieee.org/document/8916951/

[98] J. Yang, A. Nakhaei, D. Isele, K. Fujimura, and H. Zha, "CM3: Cooperative multi-goal multi-stage multi-agent reinforcement learning," 2018, *arXiv:1809.05188*. [Online]. Available: http://arxiv.org/abs/1809.05188
[99] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," 2016, *arXiv:1610.03295*. [Online]. Available: http://arxiv.org/abs/1610.03295
[100] C.-J. Hoel, K. Driggs-Campbell, K. Wolff, L. Laine, and M. J. Kochenderfer, "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving," 2019, *arXiv:1905.02680*. [Online]. Available: http://arxiv.org/abs/1905.02680
[101] W. Xia, H. Li, and B. Li, "A control strategy of autonomous vehicles based on deep reinforcement learning," in *Proc. 9th Int. Symp. Comput. Intell. Design (ISCID)*, Dec. 2016, pp. 198–201. [Online]. Available: http://ieeexplore.ieee.org/document/7830823/
[102] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 2034–2039. [Online]. Available: https://ieeexplore.ieee.org/document/8461233/
[103] R. Rădulescu, M. Legrand, K. Efthymiadis, D. M. Roijers, and A. Nowé, "Deep multi-agent reinforcement learning in a homogeneous open population," in *Proc. Benelux Conf. Artif. Intell.*, 2019, pp. 90–105.
[104] C. Yu, J. Hao, X. Wang, and Z. Feng, "Reinforcement learning for cooperative overtaking," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2019, pp. 341–349.
[105] C. Yu *et al.*, "Distributed multiagent coordinated learning for autonomous driving in highways based on dynamic coordination graphs," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 2, pp. 735–748, Feb. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8638814/
[106] M. Schutera, N. Goby, D. Neumann, and M. Reischl, "Transfer learning versus multi-agent learning regarding distributed decision-making in highway traffic," 2018, *arXiv:1810.08515*. [Online]. Available: http://arxiv.org/abs/1810.08515
[107] S. Cicek, S. Soatto, A. Nakhaei, and K. Fujimura, "MARL-PPS: Multi-agent reinforcement learning with periodic parameter sharing," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2019, pp. 1883–1885.
[108] M. Kaushik, N. Singhania, P. S., and K. M. Krishna, "Parameter sharing reinforcement learning architecture for multi agent driving," in *Proc. Adv. Robot.*, Jul. 2019, pp. 1–7.
[109] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-agent imitation learning for driving simulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1534–1539. [Online]. Available: https://ieeexplore.ieee.org/document/8593758/
[110] L. Fridman, B. Jenik, and J. Terwilliger, "DeepTraffic: Driving fast through dense traffic with deep reinforcement learning," in *Proc. Neural Inf. Process. Syst. (NIPS) Deep Reinforcement Learn. Workshop*, Montreal, QC, Canada, Jan. 2019, doi: 10.5281/zenodo.2530457.
[111] J. Colyar and J. Halkias, "US highway 101 dataset," United States Dept. Transp. Federal Highway Admin., Washington, DC, USA, Tech. Rep. HRT-07-030, 2007.
[112] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transp. Res. A, Policy Pract.*, vol. 94, pp. 182–193, Dec. 2016.
[113] F. Falcini, G. Lami, and A. M. Costanza, "Deep learning in automotive software," *IEEE Softw.*, vol. 34, no. 3, pp. 56–63, May 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7927925/
[114] *Automotive SPICE Process Assessment/Reference Model*, VDA QMC Working Group, Automot. SIG, Berlin, Germany, 2015.
[115] *Road Vehicles—Functional Safety—Part 1: Vocabulary*, Standard ISO 26262, 2011.
[116] J. Garcia and F. Fernandez. (2015). *A Comprehensive Survey on Safe Reinforcement Learning*. pp. 1437–1480. [Online]. Available: http://jmlr.org/papers/v16/garcia15a.html
[117] X. Xiong, J. Wang, F. Zhang, and K. Li, "Combining deep reinforcement learning and safety based control for autonomous driving," 2016, *arXiv:1612.00147*. [Online]. Available: http://arxiv.org/abs/1612.00147
[118] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," 2017, *arXiv:1702.01182*. [Online]. Available: http://arxiv.org/abs/1702.01182
[119] A. Baheri, S. Nageshrao, H. Eric Tseng, I. Kolmanovsky, A. Girard, and D. Filev, "Deep reinforcement learning with enhanced safety for autonomous highway driving," 2019, *arXiv:1910.12905*. [Online]. Available: http://arxiv.org/abs/1910.12905
[120] L. Wen, J. Duan, S. Eben Li, S. Xu, and H. Peng, "Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization," 2020, *arXiv:2003.01303*. [Online]. Available: http://arxiv.org/abs/2003.01303
[121] P. S. Thomas, "Safe reinforcement learning," Ph.D. dissertation, College Inf. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, 2015. [Online]. Available: https://people.cs.umass.edu/~pthomas/papers/Thomas2015c.pdf
[122] Z. Szalay, T. Tettamanti, D. Esztergár-Kiss, I. Varga, and C. Bartolini, "Development of a test track for driverless cars: Vehicle design, track configuration, and liability considerations," *Periodica Polytechnica Transp. Eng.*, vol. 46, no. 1, p. 29, Mar. 2017. [Online]. Available: https://pp.bme.hu/tr/article/view/10753
[123] A. Kendall *et al.*, "Learning to drive in a day," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8248–8254.
[124] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, nos. 1–2, pp. 151–175, May 2010.
[125] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 23–30.
[126] L. Weng. (2019). *Domain Randomization for Sim2Real Transfer*. [Online]. Available: http://lilianweng.github.io/lil-log/2019/05/04/domain-randomization.html
[127] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 1–13.
[128] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[129] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Dec. 2018, pp. 1–14. [Online]. Available: http://arxiv.org/abs/1812.02341
[130] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," 2019, *arXiv:1904.04762*. [Online]. Available: http://arxiv.org/abs/1904.04762
[131] F. Ramos, R. Carvalhaes Possas, and D. Fox, "BayesSim: Adaptive domain randomization via probabilistic inference for robotics simulators," 2019, *arXiv:1906.01728*. [Online]. Available: http://arxiv.org/abs/1906.01728

Szilárd Aradi (Member, IEEE) received the M.Sc. and Ph.D. degrees from the Budapest University of Technology and Economics, Budapest, Hungary, in 2005 and 2015, respectively. He is currently working with the Department of Control for Transportation and Vehicle Systems, Budapest University of Technology and Economics. Since 2016, he has been a Senior Lecturer with the Department of Control for Transportation and Vehicle Systems, Budapest University of Technology and Economics. His research interests include embedded systems, communication networks, vehicle mechatronics, and reinforcement learning. His research and industrial works have involved railway information systems, vehicle on-board networks, and vehicle control.