Adaptive Behavior Generation for Autonomous Driving using Deep Reinforcement Learning with Compact Semantic States

Peter Wolf[1], Karl Kurzer[2], Tobias Wingert[2], Florian Kuhnt[1] and J. Marius Zöllner[1,2]

Abstract— Making the right decision in traffic is a challenging task that is highly dependent on individual preferences as well as the surrounding environment. Therefore it is hard to model solely based on expert knowledge. In this work we use Deep Reinforcement Learning to learn maneuver decisions based on a compact semantic state representation. This ensures a consistent model of the environment across scenarios as well as a behavior adaptation function, enabling on-line changes of desired behaviors without re-training. The input for the neural network is a simulated object list similar to that of Radar or Lidar sensors, superimposed by a relational semantic scene description. The state as well as the reward are extended by a behavior adaptation function and a parameterization respectively. With little expert knowledge and a set of mid-level actions, it can be seen that the agent is capable to adhere to traffic rules and learns to drive safely in a variety of situations.

I. INTRODUCTION

While sensors are improving at a staggering pace and actuators as well as control theory are well up to par to the challenging task of autonomous driving, it is yet to be seen how a robot can devise decisions that navigate it safely in a heterogeneous environment that is partially made up by humans who not always take rational decisions or have known cost functions.

Early approaches for maneuver decisions focused on predefined rules embedded in large state machines, each requiring thoughtful engineering and expert knowledge [1], [2], [3].

Recent work focuses on more complex models with additional domain knowledge to predict and generate maneuver decisions [4]. Some approaches explicitly model the interdependency between the actions of traffic participants [5] as well as address their replanning capabilities [6].

With the large variety of challenges that vehicles with a higher degree of autonomy need to face, the limitations of rule- and model-based approaches devised by human expert knowledge that proved successful in the past become apparent.

At least since the introduction of AlphaZero, which discovered the same game-playing strategies as humans did in Chess and Go in the past, but also learned entirely unknown strategies, it is clear, that human expert knowledge is overvalued [7], [8]. Hence, it is only reasonable to apply the same techniques to the task of behavior planning in
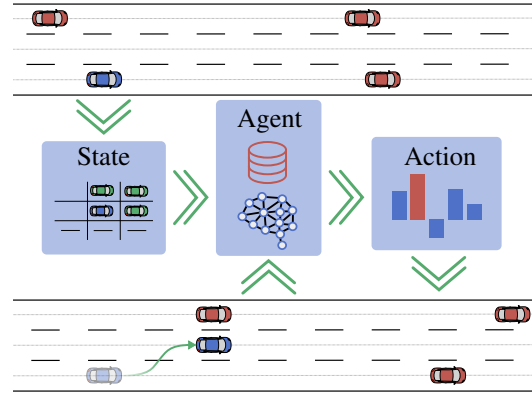

Fig. 1: The initial traffic scene is transformed into a compact semantic state representation s and used as input for the reinforcement learning (RL) agent. The agent estimates the action a with the highest return (Q-value) and executes it, e.g., changing lanes. Afterwards a reward r is collected and a new state s' is reached. The transition (s, a, r, s') is stored in the agent's replay memory.

autonomous driving, relying on data-driven instead of model-based approaches.

The contributions of this work are twofold. First, we employ a compact semantic state representation, which is based on the most significant relations between other entities and the ego vehicle. This representation is neither dependent on the road geometry nor the number of surrounding vehicles, suitable for a variety of traffic scenarios. Second, using parameterization and a behavior adaptation function we demonstrate the ability to train agents with a changeable desired behavior, adaptable on-line, not requiring new training.

The remainder of this work is structured as follows: In Section II we give a brief overview of the research on behavior generation in the automated driving domain and deep reinforcement learning. A detailed description of our approach, methods and framework follows in Section III and Section IV respectively. In Section V we present the evaluation of our trained agents. Finally, we discuss our results in Section VI.

II. RELATED WORK

Initially most behavior planners were handcrafted state machines, made up by a variety of modules to handle different tasks of driving. During the DARPA Urban Challenge Boss (CMU) for example used five different modules to conduct on road driving. The responsibilities of the modules ranged from lane selection, merge planning to distance keeping [1]. Other participants such as Odin (Virginia Tech)

[1]FZI Research Center for Information Technology, Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany {wolf, kuhnt, zoellner}@fzi.de [2]Karlsruhe Institute of Technology, Kaiserstr. 12, 76131 Karlsruhe, Germany kurzer@kit.edu, wingerttobias@gmail.com

or Talos (MIT) developed very similar behavior generators [2], [3].

Due to the limitations of state machines, current research has expanded on the initial efforts by creating more complex and formal models: A mixture of POMDP, stochastic non-linear MPC and domain knowledge can be used to generate lane change decisions in a variety of traffic scenarios [4]. Capturing the mutual dependency of maneuver decisions between different agents, planning can be conducted with foresight [5], [6]. While [5] plans only the next maneuver focusing on the reduction of collision probabilities between all traffic participants, [6] explicitly addresses longer planning horizons and the replanning capabilities of others.

Fairly different is the high-level maneuver planning presented by [9] that is based on [10], where a semantic method to model the state and action space of the ego vehicle is introduced. Here the ego vehicle's state is described by its relations to relevant objects of a structured traffic scene, such as other vehicles, the road layout and road infrastructure. Transitions between semantic states can be restricted by traffic rules, physical barriers or other constraints. Combined with an A* search and a path-time-velocity planner a sequence of semantic states and their implicit transitions can be generated.

In recent years, deep reinforcement learning (DRL) has been successfully used to learn policies for various challenges. Silver et al. used DRL in conjunction with supervised learning on human game data to train the policy networks of their program AlphaGo [11]; [12], [13] present an overview of RL and DRL respectively. In [7] and [8] their agents achieve superhuman performance in their respective domains solely using a self-play reinforcement learning algorithm which utilizes Monte Carlo Tree Search (MCTS) to accelerate the training. Mnih et al. proposed their deep Q-network (DQN) [14], [15] which was able to learn policies for a plethora of different Atari 2600 games and reach or surpass human level of performance. The DQN approach offers a high generalizability and versatility in tasks with high dimensional state spaces and has been extended in various work [16], [17], [18]. Especially actor-critic approaches have shown huge success in learning complex policies and are also able to learn behavior policies in domains with a continuous action space [19], [20].

In the domain of autonomous driving, DRL has been used to directly control the movements of simulated vehicles to solve tasks like lane-keeping [21], [22]. In these approaches, the agents receive sensor input from the respective simulation environments and are trained to determine a suitable steering angle to keep the vehicle within its driving lane. Thereby, the focus lies mostly on low-level control.

In contrast, DRL may also be applied to the problem of forming long term driving strategies for different driving scenarios. For example, DRL can be used in intersection scenarios to determine whether to cross an intersection by controlling the longitudinal movement of the vehicle along a predefined path [23]. The resulting policies achieve a lower wait time than using a Time-To-Collision policy. In the same
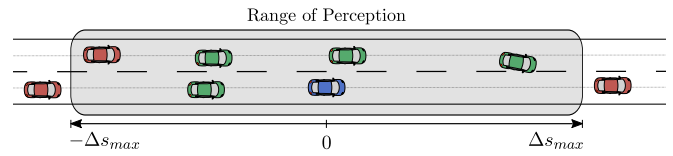
(a) The ego vehicle (blue) is driving on a two lane road. Five other vehicles are in its sensor range. Based on a vehicle scope with $\Lambda_{lateral} = \Lambda_{ahead} = \Lambda_{behind} = 1$ only four vehicles (green) are considered in the semantic state representation.
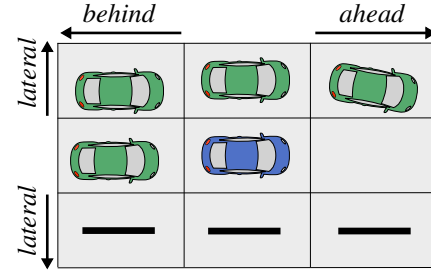
(b) The relational grid resulting from the scene of (a) with $\Lambda_{lateral} = \Lambda_{ahead} = \Lambda_{behind} = 1$. Moreover, there is no adjacent lane to the right, which is represented in the grid as well.

Fig. 2: An example scene (a) is transformed to a relational grid (b) using a vehicle scope Λ with $\Lambda_{lateral} = \Lambda_{ahead} = \Lambda_{behind} = 1$. The red vehicle, which is in sensor range, is not represented in the grid. Since there is no vehicle driving on the same lane in front of the ego vehicle, the respective relational position in the grid is empty.

way, DRL techniques can be used to learn lateral movement, e.g. lane change maneuvers in a highway simulation [24].

Since it can be problematic to model multi-agent scenarios as a Markov Decision Process (MDP) due to the unpredictable behavior of other agents, one possibility is to decompose the problem into learning a cost function for driving trajectories [25]. To make learning faster and more data efficient, expert knowledge can be incorporated by restricting certain actions during the training process [26]. Additionally, there is the option to handle task and motion planning by learning low-level controls for lateral as well as longitudinal maneuvers from a predefined set and a high-level maneuver policy [27].

Our approach addresses the problem of generating adaptive behavior that is applicable to a multitude of traffic scenarios using a semantic state representation. Not wanting to limit our concept to a single scenario, we chose a more general set of "mid-level" actions, e.g. *decelerate* instead of *Stop* or *Follow*. This does not restrict the possible behavior of the agent and enables the agent to identify suitable high-level actions implicitly. Further, our agent is trained without restrictive expert knowledge, but rather we let the agent learn from its own experience, allowing it to generate superior solutions without the limiting biases introduced by expert knowledge [7], [8].

III. APPROACH

We employ a deep reinforcement learning approach to generate adaptive behavior for autonomous driving.
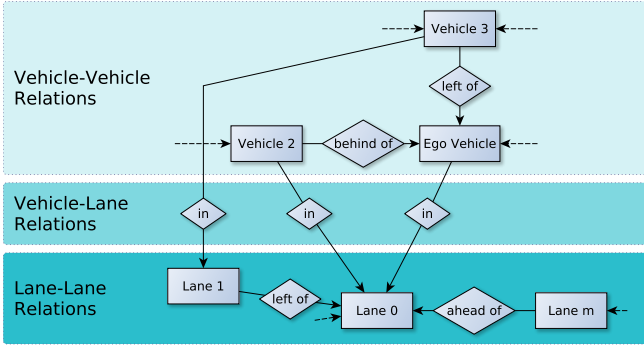
Fig. 3: Partially visualized entity-relationship model of the scene in Fig. 2a. The vehicle topology is modeled by vehicle-vehicle relations whereas the lane topology is modeled by lane-lane relations.

A reinforcement learning process is commonly modeled as an MDP [28] (S, A, R, δ, T) where S is the set of states, A the set of actions, $R : S \times A \times S \to \mathbb{R}$ the reward function, $\delta : S \times A \to S$ the state transition model and T the set of terminal states. At timestep i an agent in state $s \in S$ can choose an action $a \in A$ according to a policy π and will progress into the successor state s' receiving reward r. This is defined as a transition $t = (s, a, s', r)$.

The aim of reinforcement learning is to maximize the future discounted return $G_i = \sum_{n=i} \gamma^{n-i} r_i$. A DQN uses Q-Learning [29] to learn Q-values for each action given input state s based on past transitions. The predicted Q-values of the DQN are used to adapt the policy π and therefore change the agent's behavior. A schematic of this process is depicted in Fig. 1.

For the input state representation we adapt the ontology-based concept from [10] focusing on relations with other traffic participants as well as the road topology. We design the state representation to use high level preprocessed sensory inputs such as object lists provided by common Radar and Lidar sensors and lane boundaries from visual sensors or map data. To generate the desired behavior the reward is comprised of different factors with varying priorities. In the following the different aspects are described in more detail.

A. Semantic Entity-Relationship Model

A traffic scene τ is described by a semantic entity-relationship model, consisting of all scene objects and relations between them. We define it as the tuple $(\mathcal{E}, \mathcal{R})$, where

- $\mathcal{E} = \{e_0, e_1, ..., e_n\}$: set of scene objects (entities).
- $\mathcal{R} = \{r_0, r_1, ..., r_m\}$: set of relations.

The scene objects contain all static and dynamic objects, such as vehicles, pedestrians, lane segments, signs and traffic lights.

In this work we focus on vehicles $\mathcal{V} \subset \mathcal{E}$, lane segments $\mathcal{L} \subset \mathcal{E}$ and the three relation types vehicle-vehicle relations, vehicle-lane relations and lane-lane relations. Using these entities and relations an entity-relationship representation of a traffic scene can be created as depicted in Fig. 3. Every and relation holds several properties or attributes of the scene objects, such as e.g. absolute positions or relative velocities.

This scene description combines low level attributes with high level relational knowledge in a generic way. It is thus applicable to any traffic scene and vehicle sensor setup, making it a beneficial state representation.

But the representation is of varying size and includes more aspects than are relevant for a given driving task. In order to use this representation as the input to a neural network we transform it to a fixed-size relational grid that includes only the most relevant relations.

B. Relational Grid

We define a relational grid, centered at the ego vehicle $v_{ego} \in \mathcal{V}$, see Fig. 2. The rows correspond to the relational lane topology, whereas the columns correspond to the vehicle topology on these lanes.

To define the size of the relational grid, a vehicle scope Λ is introduced that captures the lateral and longitudinal dimensions, defined by the following parameters:

- $\Lambda_{lateral} \in \mathbb{N}$: maximum number of laterally adjacent lanes to the ego vehicle's lane that is considered
- $\Lambda_{ahead} \in \mathbb{N}$: maximum number of vehicles per lane driving in front of the ego vehicle that is considered
- $\Lambda_{behind} \in \mathbb{N}$: maximum number of vehicles per lane driving behind the ego vehicle that is considered

The set of vehicles inside the vehicle scope Λ is denoted by $\mathcal{V}_\Lambda \subseteq \mathcal{V}$. The different features of vehicles and lanes are represented by separate layers of the grid, resulting in a semantic state representation, see Fig. 4. The column vector of each grid cell holds attributes of the vehicle and the lane it belongs to. While the ego vehicle's features are absolute values, other vehicles' features are relative to the ego vehicle or the lane they are in (induced by the vehicle-vehicle and vehicle-lane relations of the entity-relationship model):

1) Non-ego vehicle features: The features of non-ego vehicles $v_i \in \mathcal{V}_\Lambda \backslash v_{ego}$ are $\boldsymbol{f}_i^{\mathcal{V}} = (\Delta s_i, \Delta \dot{s}_i, \Delta d_i, \Delta \phi_i)$:

- Δs_i: longitudinal position relative to v_{ego}
- $\Delta \dot{s}_i$: longitudinal velocity relative to v_{ego}
- Δd_i: lateral alignment relative to lane center
- $\Delta \phi_i$: heading relative to lane orientation

2) Ego vehicle features: To generate adaptive behavior, the ego vehicle's features include a function $\Omega(\tau, \theta)$ that describes the divergence from the desired behavior of the ego vehicle parameterized by θ given the traffic scene τ.

Thus the features for v_{ego} are $\boldsymbol{f}_{ego}^{v} = (\Omega(\tau, \theta), \dot{s}_{ego}, k)$:

- $\Omega(\tau, \theta)$: behavior adaptation function
- $\dot{s}_{ego}$: longitudinal velocity of ego vehicle
- k: lane index of ego vehicle
  ($k = 0$ being the right most lane)

3) Lane features: The lane topology features of $l_k \in \mathcal{L}$ are $\boldsymbol{f}_k^{\mathcal{L}} = (\Gamma_k, \Delta s_k)$:

- Γ_k: lane type (normal, acceleration)
- Δs_k: lane ending relative to v_{ego}

Non-perceivable features or missing entities are indicated in the semantic state representation by a dedicated value.
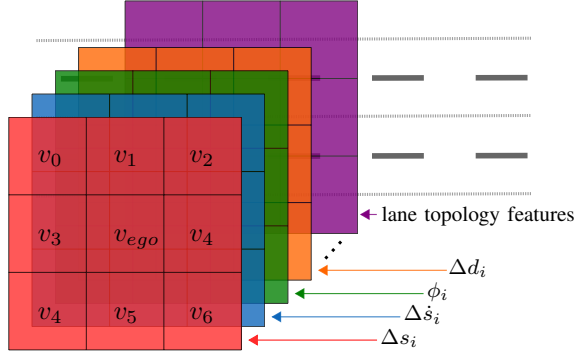
Fig. 4: The relational grid contains one layer per feature. The vehicle features $\boldsymbol{f}_i^{\mathcal{V}}$ and $\boldsymbol{f}_{ego}^{\mathcal{V}}$ share layers and are in the cells of the v_i and v_{ego} respectively. The lane features $\boldsymbol{f}_k^{\mathcal{L}}$ are on additional layers in the k-th row of the grid.

The relational grid ensures a consistent representation of the environment, independent of the road geometry or the number of surrounding vehicles.

The resulting input state $s \in S$ is depicted in Fig. 4 and fed into a DQN.

C. Action Space

The vehicle's actions space is defined by a set of semantic actions that is deemed sufficient for most on-road driving tasks, excluding special cases such as U-turns. The longitudinal movement of the vehicle is controlled by the actions *accelerate* and *decelerate*. While executing these actions, the ego vehicle keeps its lane. Lateral movement is generated by the actions *lane change left* as well as *lane change right* respectively. 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. The *default* action results in no change of either velocity, lateral alignment or heading.

D. Adaptive Desired Behavior through Reward Function

With the aim to generate adaptive behavior we extend the reward function $R(s,a)$ by a parameterization θ. This parameterization is used in the behavior adaptation function $\Omega(\tau,\theta)$, so that the agent is able to learn different desired behaviors without the need to train a new model for varying parameter values.

Furthermore, the desired driving behavior consists of several individual goals, modeled by separated rewards. We rank these reward functions by three different priorities. The highest priority has collision avoidance, important but to a lesser extent are rewards associated with traffic rules, least prioritized are rewards connected to the driving style.

The overall reward function $R(s,a,\theta)$ can be expressed as follows:

$$R(s,a,\theta) = \begin{cases} R_{collision}(s,\theta) & s \in S_{collision} \\ R_{rules}(s,\theta) & s \in S_{rules} \\ R_{driving\ style}(s,a,\theta) & \text{else,} \end{cases} \quad (1)$$

The subset $S_{collision} \subset S$ consists of all states s describing a collision state of the ego vehicle v_{ego} and another vehicle v_i. In these states the agent only receives the immediate reward without any possibility to earn any other future rewards. Additionally, attempting a lane change to a nonexistent adjacent lane is also treated as a collision.

The state dependent evaluation of the reward factors facilitates the learning process. As the reward for a state is independent of rewards with lower priority, the eligibility trace is more concise for the agent being trained. For example, driving at the desired velocity does not mitigate the reward for collisions.

IV. EXPERIMENTS

A. Framework

While our concept is able to handle data from many preprocessing methods used in autonomous vehicles, we tested the approach with the traffic simulation SUMO [30]. A schematic overview of the framework is depicted in Fig. 5. We use SUMO in our setup as it allows the initialization and execution of various traffic scenarios with adjustable road layout, traffic density and the vehicles's driving behavior. To achieve this, we extend TensorForce [31] with a highly configurable interface to the SUMO environment. Tensor-Force is a reinforcement library based on TensorFlow [32], which enables the deployment of various customizable DRL methods, including DQN.
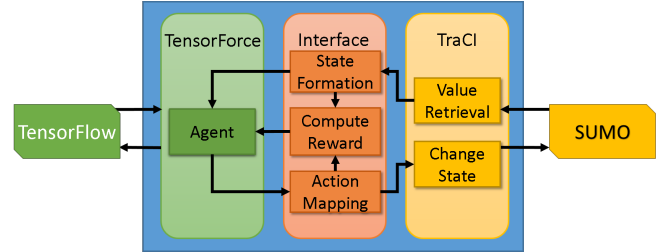

Fig. 5: The current state from SUMO is retrieved and transformed into the semantic state representation. This is the input to the RL agent, which is trained using the TensorForce library. Chosen actions are mapped to a respective state change of the simulation. The agent's reward is then computed based on the initial state, the chosen action and the successor state.

The interface fulfills three functions: the state extraction from SUMO, the calculation of the reward and the mapping of the chosen maneuver onto valid commands for the SUMO controlled ego vehicle. The necessary information about the ego vehicle, the surrounding vehicles as well as additional information about the lanes are extracted from the running simulation. These observed features are transformed into the state representation of the current scene. For this purpose the surrounding positions relative to the ego vehicle are checked and if another vehicle fulfills a relation, the selected feature set for this vehicle is written into the state representation for its relation. The different reward components can be added or removed, given a defined value or range of values and a priority.
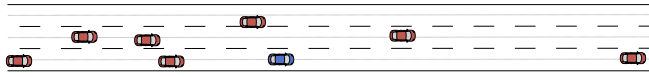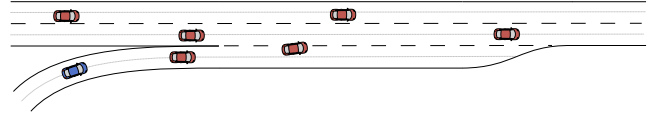
(a) Highway Driving
(b) Highway Merging

Fig. 6: To examine the agent's compliance to traffic rules, it is trained and evaluated on two different traffic scenarios. In (a) the agent has the obligation to drive on the right most lane and must not pass others from the right, amongst other constraints. In (b) the agent is allowed to accelerate while on the on-ramp and also might overtake vehicles on its left. But it has to leave the on-ramp before it ends.

This setup allows us to deploy agents using various DRL methods, state representations and rewards in multiple traffic scenarios. In our experiments we used a vehicle scope with $\Lambda_{behind} = 1$ and $\Lambda_{ahead} = \Lambda_{lateral} = 2$. This allows the agent to always perceive all lanes of a 3 lane highway and increases its potential anticipation.

B. Network

In this work we use the DQN approach introduced by Mnih et al. [15] as it has shown its capabilities to successfully learn behavior policies for a range of different tasks. While we use the general learning algorithm described in [15], including the usage of experience replay and a secondary target network, our actual network architecture differs from theirs. The network from Mnih et al. was designed for a visual state representation of the environment. In that case, a series of convolutional layers is commonly used to learn a suitable low-dimensional feature set from this kind of high-dimensional sensor input. This set of features is usually further processed in fully-connected network layers.

Since the state representation in this work already consists of selected features, the learning of a low-dimensional feature set using convolutional layers is not necessary. Therefore we use a network with solely fully-connected layers, see Tab. I.

The size of the input layer depends on the number of features in the state representation. On the output layer there is a neuron for each action. The given value for each action is its estimated Q-value.

Layer	Neurons
Input	—
Hidden Layer 1	512
Hidden Layer 2	512
Hidden Layer 3	256
Hidden Layer 4	64
Output	5

TABLE I: The network layout of our DQN agent. The size of the input neurons is determined by the number of features in the state space. The five output neurons predict the five Q-values for respective actions.

C. Training

During training the agents are driving on one or more traffic scenarios in SUMO. An agent is trained for a maximum of 2 million timesteps, each generating a transition consisting of the observed state, the selected action, the subsequent state and the received reward. The transitions are stored in the replay memory, which holds up to 500,000 transitions. After reaching a threshold of at least 50,000 transitions in memory,

a batch of 32 transitions is randomly selected to update the network's weights every fourth timestep. We discount future rewards by a factor of 0.9 during the weight update. The target network is updated every 50,000th step.

To allow for exploration early on in the training, an ϵ-greedy policy is followed. With a probability of ϵ the action to be executed is selected randomly, otherwise the action with the highest estimated Q-value is chosen. The variable ϵ is initialized as 1, but decreased linearly over the course of 500,000 timesteps until it reaches a minimum of 0.1, reducing the exploration in favor of exploitation. As optimization method for our DQN we use the RMSProp algorithm [33] with a learning rate of 10^{-5} and decay of 0.95.

The training process is segmented into episodes. If an agent is trained on multiple scenarios, the scenarios alternate after the end of an episode. To ensure the agent experiences a wide range of different scenes, it is started with a randomized departure time, lane, velocity and θ in the selected scenario at the beginning and after every reset of the simulation. In a similar vein, it is important that the agent is able to observe a broad spectrum of situations in the scenario early in the training. Therefore, should the agent reach a collision state $s \in S_{collision}$ by either colliding with another vehicle or attempting to drive off the road, the current episode is finished with a terminal state. Afterwards, a new episode is started immediately without resetting the simulation or changing the agent's position or velocity. Since we want to avoid learning an implicit goal state at the end of a scenario's course, the simulation is reset if a maximum amount of 200 timesteps per episode has passed or the course's end has been reached and the episode ends with a non-terminal state.

D. Scenarios

Experiments are conducted using two different scenarios, see Fig. 6. One is a straight multi-lane highway scenario. The other is a merging scenario on a highway with an on-ramp.

To generate the desired adaptive behavior, parameterized reward functions are defined (see Eq. 1). We base R_{rules} on German traffic rules such as the obligation to drive on the right most lane ($r_{keepRight}$), prohibiting overtaking on the right ($r_{passRight}$) as well as keeping a minimum distance to vehicles in front ($r_{safeDistance}$). A special case is the acceleration lane, where the agent is allowed to pass on the right and is not required to stay on the right most lane. Instead the agent is not allowed to enter the acceleration lane ($r_{notEnter}$).
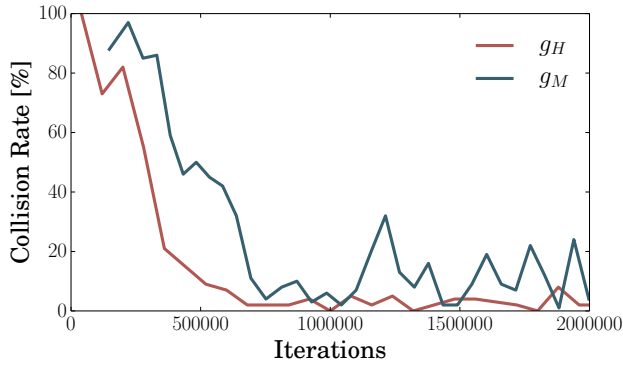
997

Fig. 7: Collision Rate of the agents g_H and g_M during training.

Fig. 8: Relative duration of rule violations of the agents g_H and g_M during training.

Similarly, $R_{driving\ style}$ entails driving a desired velocity ($r_{velocity}$) ranging from $80\,\text{km/h}$ to $115\,\text{km/h}$ on the highway and from $40\,\text{km/h}$ to $80\,\text{km/h}$ on the merging scenario. The desired velocity in each training episode is defined by θ_v which is sampled uniformly over the scenario's velocity range. Additionally, $R_{driving\ style}$ aims to avoid unnecssary lane and velocity changes (r_{action}).

With these constraints in mind, the parameterized reward functions are implemented as follows, to produce the desired behavior.

$$R_{collision}(s, \theta) = \theta_t r_{collision} \tag{2}$$

$$R_{rules}(s, \theta) = r_{passRight}(s, \theta_p) + r_{notEnter}(s, \theta_n) \\ + r_{safeDistance}(s, \theta_s) + r_{keepRight}(s, \theta_k) \tag{3}$$

$$R_{driving\ style}(s, a, \theta) = r_{action}(a, \theta_a) + r_{velocity}(s, \theta_v) \tag{4}$$

To enable different velocity preferences, the behavior adaptation function Ω returns the difference between the desired and the actual velocity of v_{ego}.

V. EVALUATION

During evaluation we trained an agent g_H only on the highway scenario and an agent g_M only on the merging scenario. In order to show the versatility of our approach, we additionally trained an agent g_C both on the highway as well as the merging scenario (see Tab.II). Due to the nature of our compact semantic state representation we are able to achieve this without further modifications. The agents are evaluated during and after training by running the respective scenarios 100 times. To assess the capabilities of the trained agents, using the concept mentioned in Section III, we introduce the following metrics.

Collision Rate [%]: The collision rate denotes the average amount of collisions over all test runs. In contrast to the training, a run is terminated if the agent collides. As this is such a critical measure it acts as the most expressive gauge assessing the agents performance.
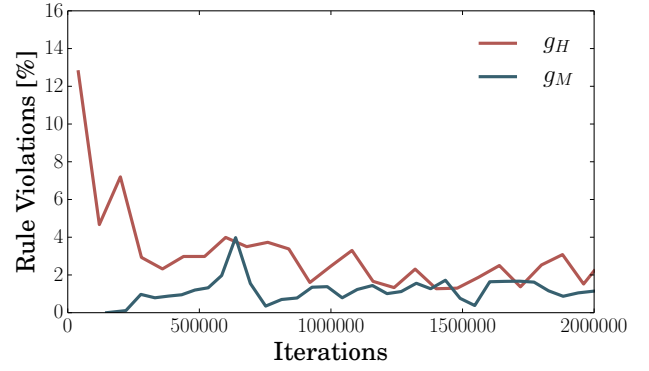
Avg. Distance between Collisions [km]: The average distance travelled between collisions is used to remove the bias of the episode length and the vehicle's speed.

Rule Violations [%]: Relative duration during which the agent is not keeping a safe distance or is overtaking on right.

Lane Distribution [%]: The lane distribution is an additional weak indicator for the agent's compliance with the traffic rules.

Avg. Speed [m/s]: The average speed of the agent does not only indicate how fast the agent drives, but also displays how accurate the agent matches its desired velocity.

The results of the agents trained on the different scenarios are shown in Tab. II. The agents generally achieve the desired behavior. An example of an overtaking maneuver is presented in Fig. 9. During training the collision rate of g_H decreases to a decent level (see Fig. 7). Agent g_M takes more training iterations to reduce its collision rate to a reasonable level, as it not only has to avoid other vehicles, but also needs to leave the on-ramp. Additionally, g_M successfully learns to accelerate to its desired velocity on the on-ramp. But for higher desired velocities this causes difficulties leaving the ramp or braking in time in case the middle lane is occupied. This effect increases the collision rate in the latter half of the training process.
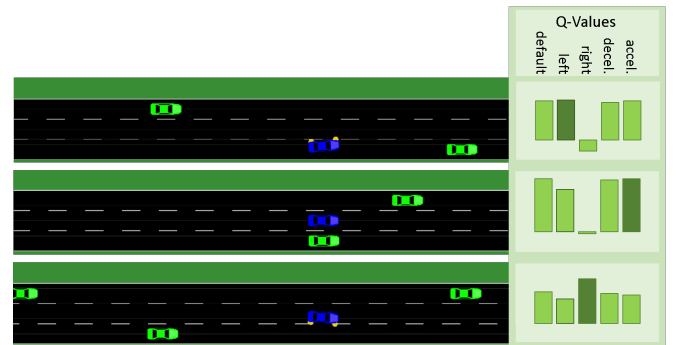

Fig. 9: The agent (blue) is driving behind a slower vehicle (green). In this situation the action *lane change left* has the highest estimated Q-value. After the lange change, the agent accelerates and overtakes the slower vehicle. Subsequently, the agent changes back to the right most lane.

The relative duration of rule violations by g_H reduces over the course of the training, but stagnates at approximately 2% (see Fig. 8). A potential cause is our strict definition of when an overtaking on the right occurs. The agent almost never performs a full overtaking maneuver from the right, but might drive faster than another vehicle on the left hand side, which will already be counted towards our metric. For g_M the duration of rule violations is generally shorter, starting low, peaking and then also stagnating similarly to g_H. This is explained due to overtaking on the right not being considered on the acceleration lane. The peak emerges as a result of the agent leaving the lane more often at this point.

The lane distribution of g_H (see Tab. II) demonstrates that the agent most often drives on the right lane of the highway, to a lesser extent on the middle lane and only seldom on the left lane. This reflects the desired behavior of adhering to the obligation of driving on the right most lane and only using the other lanes for overtaking slower vehicles. In the merging scenario this distribution is less informative since the task does not provide the same opportunities for lane changes.

TABLE II: Results of the trained agents. The agents g_H and g_M were only evaluated on their respective scenario, while g_C was evaluated on both. The results for g_C are listed separately for each scenario.

	g_H	g_M	g_C (highway)	g_C (merging)
Collision Rate	2%	1%	2%	4%
avg. Distance	98.37 km	28.98 km	75.92 km	7.19 km
Rule Violations	1.52%	0.87%	1.62%	0.5%
Lane 0	64.12%	87.99%	57.11%	87.8%
Lane 1	22.07%	11.81%	34.46%	11.85%
Lane 2	13.81%	0.2%	8.43%	0.35%

To measure the speed deviation of the agents, additional test runs with fixed values for the desired velocity were performed. The results are shown in Tab. III. As can be seen, the agents adapt their behavior, as an increase in the desired velocity raises the average speed of the agents. In tests with other traffic participants, the average speed is expectedly lower than the desired velocity, as the agents often have to slow down and wait for an opportunity to overtake. Especially in the merging scenario the agent is unable to reach higher velocities due to these circumstances. During runs on an empty highway scenario, the difference between the average and desired velocity diminishes.

Although g_H and g_M outperform it on their individual tasks, g_C achieves satisfactory behavior on both. Especially, it is able to learn task specific knowledge such as overtaking in the acceleration lane of the on-ramp while not overtaking from the right on the highway.

A video of our agents behavior is provided online.[1]

VI. Conclusions

In this work two main contributions have been presented. First, we introduced a compact semantic state representation that is applicable to a variety of traffic scenarios. Using

[1] http://url.fzi.de/behavior-iv2018

a relational grid our representation is independent of road topology, traffic constellation and sensor setup.

Second, we proposed a behavior adaptation function which enables changing desired driving behavior online without the need to retrain the agent. This eliminates the requirement to generate new models for different driving style preferences or other varying parameter values.

TABLE III: The average speed of each agent given varying desired velocities [m/s]. The agents have been evaluated on the training scenarios with normal traffic density as well as on an empty highway. While not every agent is able to achieve the desired velocity precisely, their behavior adapts to the different parameter values.

v_{des}	g_H		g_M		g_C		
	highway	empty	merging	empty	highway	merging	empty
12	12.8	13.2	10.8	13.3	12.1	10.3	12.6
17	17.1	17.1	13.5	18.6	16.9	13.2	17.1
22	21.1	21.9	14.7	23.7	20.9	14.5	21.8
25	23.4	24.8	15.1	26.5	23.0	15.0	23.9
30	25.7	29.2	15.3	30.4	25.2	14.8	28.0

Agents trained with this approach performed well on different traffic scenarios, i.e. highway driving and highway merging. Due to the design of our state representation and behavior adaptation, we were able to develop a single model applicable to both scenarios. The agent trained on the combined model was able to successfully learn scenario specific behavior.

One of the major goals for future work we kept in mind while designing the presented concept is the transfer from the simulation environment to real world driving tasks. A possible option is to use the trained networks as a heuristic in MCTS methods, similar to [27]. Alternatively, our approach can be used in combination with model-driven systems to plan or evaluate driving behavior.

To achieve this transfer to real-world application, we will apply our state representation to further traffic scenarios, e.g., intersections. Additionally, we will extend the capabilities of the agents by adopting more advanced reinforcement learning techniques.

References

[1] C. Urmson, J. Anhalt, D. Bagnell, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.
[2] A. Bacha, C. Bauman, R. Faruque, *et al.*, "Odin: Team victortango's entry in the darpa urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.
[3] J. Leonard, J. How, S. Teller, *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.
[4] S. Ulbrich and M. Maurer, "Towards tactical lane change behavior planning for automated vehicles," in *18th IEEE International Conference on Intelligent Transportation Systems, ITSC*, 2015, pp. 989–995.

[5] A. Lawitzky, D. Althoff, C. F. Passenberg, *et al.*, "Interactive scene prediction for automotive applications," in *Intelligent Vehicles Symposium (IV), IEEE*, 2013, pp. 1028–1033.

[6] M. Bahram, A. Lawitzky, J. Friedrichs, *et al.*, "A game-theoretic approach to replanning-aware interactive scene prediction and planning," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3981–3992, 2016.

[7] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[8] D. Silver, T. Hubert, J. Schrittwieser, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017.

[9] R. Kohlhaas, D. Hammann, T. Schamm, and J. M. Zöllner, "Planning of High-Level Maneuver Sequences on Semantic State Spaces," in *18th IEEE International Conference on Intelligent Transportation Systems, ITSC*, 2015, pp. 2090–2096.

[10] R. Kohlhaas, T. Bittner, T. Schamm, and J. M. Zöllner, "Semantic state space for high-level maneuver planning in structured traffic scenes," in *17th IEEE International Conference on Intelligent Transportation Systems, ITSC*, 2014, pp. 1060–1065.

[11] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[12] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[13] Y. Li, "Deep reinforcement learning: An overview," 2017.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," 2013.

[15] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[16] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," 2015.

[17] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," 2015.

[18] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling Network Architectures for Deep Reinforcement Learning," 2016.

[19] V. Mnih, A. P. Badia, M. Mirza, *et al.*, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1928–1937.

[20] Y. Wu, E. Mansimov, R. B. Grosse, *et al.*, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," in *Advances in Neural Information Processing Systems*, 2017, pp. 5285–5294.

[21] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, no. 19, pp. 70–76, 2017.

[22] P. Wolf, C. Hubschneider, M. Weber, *et al.*, "Learning how to drive in a real world simulation with deep q-networks," in *Intelligent Vehicles Symposium (IV), IEEE*, 2017, pp. 244–250.

[23] D. Isele, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating Intersections with Autonomous Vehicles using Deep Reinforcement Learning," 2017.

[24] B. Mirchevska, M. Blum, L. Louis, *et al.*, "Reinforcement learning for autonomous maneuvering in highway scenarios," 2017.

[25] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving," 2016.

[26] M. Mukadam, A. Cosgun, A. Nakhaei, and K. Fujimura, "Tactical decision making for lane changing with deep reinforcement learning," *Conference on Neural Information Processing Systems (NIPS)*, 2017.

[27] C. Paxton, V. Raman, G. D. Hager, and M. Kobilarov, "Combining Neural Networks and Tree Search for Task and Motion Planning in Challenging Environments," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6059–6066, 2017.

[28] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[29] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[30] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent Development and Applications of {SUMO - Simulation of Urban MObility}," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3, pp. 128–138, 2012.

[31] M. Schaarschmidt, A. Kuhnle, and K. Fricke, "Tensorforce: A tensorflow library for applied reinforcement learning," Web page, 2017. [Online]. Available: https://github.com/reinforceio/tensorforce

[32] M. Abadi, A. Agarwal, P. Barham, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[33] G. Hinton, N. Srivastava, and K. Swersky, "Rmsprop: Divide the gradient by a running average of its recent magnitude," *Neural networks for machine learning, Coursera lecture 6e*, 2012.