2018 IEEE Conference on Control Technology and Applications (CCTA)
Copenhagen, Denmark, August 21-24, 2018

Policy Gradient based Reinforcement Learning Approach for Autonomous Highway Driving

Szilard Aradi[1], Tamas Becsi[1] and Peter Gaspar[2]

***Abstract*— The paper presents the application of the Policy Gradient reinforcement learning method in the area of vehicle control. The purpose of the research presented is to design an end-to-end behavior control of a kinematic vehicle model placed in a simulated highway environment, by using reinforcement learning approach. The environment model for the surrounding traffic uses microscopic simulation to provide different situations for the agent. The environment sensing model is based on high-level sensor information, e.g. the odometry, lane position and surrounding vehicle states that can be reached from the current automotive sensors, such as camera and radar based systems. The objectives were reached through the definition of a rewarding system, with subrewards and penalties enforcing desired speed, lane keeping, keeping right and avoiding collision. After the description of the theoretical basis the environment model with the reward function is detailed. Finally the experiments with the learning process are presented and the results of the evaluation are given from some aspects of control quality and safety.**

I. INTRODUCTION AND MOTIVATION

One of the ultimate goals of the automotive industry related researches is the development of the Level 5 Autonomous Vehicle [1]. Although many manufacturers often refer to their products as nearly self-driving car, currently these products only reach Level 2. Series production of the first Level 3 vehicle is likely to begin this year by Audi.

From the aspect of control design and complexity, one of the best-defined and best-maintained environments are the highways and motorways. Therefore the first Level 3 control systems will be able to operate on highways as well. The fully automated highway driving (AHD) has been researched by the control engineers for a long time, within this the platooning was one of the most promising highway automation solution, see e.g. [2]. There is a great deal of consensus among the professionals that the control problem of AHD can not be solved by such traditional rule-based algorithms, which are used in Level 2 driver assistance systems. The large number of varied and complex traffic situations need new approaches such as artificial intelligence. However, it is still questionable today which methods and architectures will be able to provide the appropriate level of safety. Meanwhile the IT giants devote huge resources to the AI developments and some of them has started to develop autonomous vehicles and/or control systems.

One common approach is supervised learning using artificial neural networks (ANN). In automotive control systems it can be used for image processing and the so-called "end-to-end learning" where the control signals can be calculated directly from the image sequence recorded by the camera. See e.g. [3]. These solutions assume the presence of huge-amount of training data containing corresponding control signal for each frame. Another considerable approach could be the reinforcement learning (RL) which is an intensive research area today. In the recent years several remarkable results were published by Google's Deepmind, see [4] and [5]. In the field of reinforcement learning the methods are demonstrated and compared mainly by video and board games, furthermore basic robotic and control tasks. These methods can be applied for automotive research areas as well, where the majority of the published results dealt with situations in highway environment. These topics include cooperative adaptive cruise control [6], overtaking [7] and [8], on-ramp merging [9] and lane keeping [10].

Our long-term research focuses on the highway environment as well. The control system of a highly automated vehicle control function typically consist of the following layers: perception, decision-making, planning and execution (low-level control). As mentioned formerly several end-to-end solutions exist which integrates all of the layers into one ANN-based control system trained with labeled ground-truth data. Other researches try to solve subproblems or one layer with reinforcement learning. In this paper a "quasi end-to-end" reinforcement learning approach is presented. The input for the system is not generated as the raw signals (e.g. video frames) of the sensors, rather assuming that the perception layer (intelligent sensors) ensures the environment variables (lane geometry, static and dynamic object lists) with added uncertainty and noise. Given these inputs the control system calculates the control signals (steering angle, acceleration) directly. The main motivation behind is to find the proper reinforcement learning methods suitable to solve this complex control task. Naturally because of the safety related needs of the application another important goal is to find the proper control architecture (probably a hybrid architecture) which will be able to ensure the required safety level.

The paper presents the results of the first phase of the research project. We have started to apply the state-of-art

*EFOP-3.6.3-VEKOP-16-2017-00001: Talent management in autonomous vehicle control technologies- The Project is supported by the Hungarian Government and co-financed by the European Social Fund

[1] Sz. Aradi and T. Bécsi are with the Department of Control for Transportation and Vehicle Systems, Budapest University of Technology and Economics, Stoczek u. 2, H-1111 Budapest, Hungary, [szabo.adam; aradi.szilard; becsi.tamas]@mail.bme.hu

[2] P. Gáspár is with the Systems and Control Laboratory, Computer and Automation Research Institute, Hungarian Academy of Sciences, Kende u. 13-17, H-1111 Budapest, Hungary gaspar.peter@sztaki.mta.hu

reinforcement learning based control methods on a highway environment model and developed a reward function which could be suitable for this complex environment.

Section 2 describes the used policy-based reinforcement learning method. Section 3 introduces the environment model which implemented as an extension of Open AI Gym [11], then in Section 4 the simulation results are exposed. Finally in Section 5 the conclusion and the possible improvements are detailed.

II. CONTROLLER DESIGN WITH POLICY-BASED REINFORCEMENT LEARNING

A. Reinforcement Learning

In reinforcement learning (as in other areas of artificial intelligence) the learner and decision maker is called the *agent*. It interacts with the so-called *environment*, comprising everything outside the agent. These interact continually with each other, the agent selecting actions and the environment responding to these actions and presenting new situations to the agent (see Fig. 1). The terms agent, environment, and action meet the control engineers' terms controller, plant and control signals. In reinforcement learning the

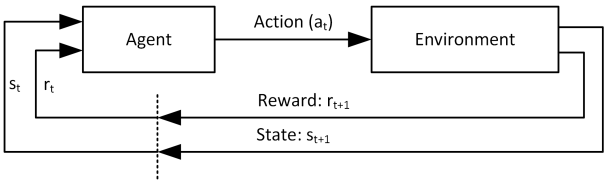

Fig. 1. Agent-environment interaction in reinforcement learning

environment can be typically formulated as a finite-state Markov Decision Process (MDP). It is described with state s_t (where $s_t \in S$ and S represents the system's state space), action a_t changes the environment state from s_t to s_{t+1} with state transition probability $P(s_t, a_t, s_{t+1})$. The agent and environment interact at each of a sequence of discrete time steps, $t = 0, 1, 2, 3, ...$ Finally the most interesting part of reinforcement learning is the numerical reward $r_{t+1} \in R$ indicates how well agent is doing. The agent's job is to learn how can be the reward maximized, i.e. to find policy $\pi : S \rightarrow A$ that maximizes the cumulative future reward $R = r_0 + r_1 + ... + r_n$, exactly the discounted future reward $R = \sum_0^n \gamma^t r_t$. The discount factor ($0 \leq \gamma \leq 1$) represents the uncertainty of the future, i.e. how much the future rewards depend on the actions in the past. A prediction of cumulative future reward $V_\pi(s) = \mathbf{E}_\pi[R|S_t = s]$ is defined as the "value" of the state which can be used to evaluate the badness or goodness of the state and therefore to select between actions. There exists an optimal policy π^* for which $V^{\pi *}$ is optimal for every state. In a finite MDP, the sets of states, actions, and rewards (S, A, R) all have a finite number of elements. The Markov property of MDP requires that the probability distribution of random variables (transitions and rewards) depends on the current state and action only, and on the past actions and states.

Depending on the selected method, the RL agent may include policy function, value function and model. The latter is the agent's representation of the environment, according to its presence in the solution we can categorize RL methods as model-free or model-based solutions.

B. Value-based Methods

The value function-based reinforcement learning is a heavily researched area thanks to Deepmind's improvements in deep Q-learning. The main idea is to learn a Q-function instead of the value function. Q-function is a predictor function which outputs a Q-value for each actions in a given state. The prediction can be updated according to the Bellmann equation:

$$Q(s_t, a_t) = r_t + \gamma \max Q(s_{t+1}, a) \tag{1}$$

The predictor function is typically a deep neural network where the training goal is the more accurate estimate of the Q-function according to the Bellmann equation. It should be noted that the usage of function approximation in RL algorithms do not guarantee the convergence, but these methods usually show good results in practice, in particular with regard to the improved versions, see e.g. [12] and [13].

C. Policy-based Methods

The other approach is based on the approximation of the policy directly. From this group of RL solutions the policy-gradient methods have gained interest in recent years to solve control problems. These algorithms modify the parameters of the function approximator (neural network) in the direction that maximizes the expected reward. Its advantage over the value-based methods is the guaranteed convergence, but typically to a local rather than global optimum [14].

Policy gradient assumes that the actions are given by a stochastic policy $\pi_\theta(a_t, s_t)$ with parameters θ. Policy-base RL is an optimization problem, where the goal is to find θ that maximizes $J(\theta) = J(\pi_\theta)$ where

$$J(\pi_\theta) = \mathbf{E}\left[\sum_{t=0}^{T} r_t\right] \tag{2}$$

in episodic environments. There are several optimization techniques available, some of them do not use gradient, but usually greater efficiency can be achieved using gradient. The policy gradient algorithms search for a local maximum in $J(\theta)$ by continuously refining a given parametrization vector. It follows the steepest ascent of the expected reward, which can be formulated by the gradient update rule:

$$\Delta\theta = \alpha \nabla_\theta J(\theta) \tag{3}$$

where $\nabla_\theta J(\theta)$ is the policy gradient vector and α is the learning rate.

Next step is computing the gradients, which can be realized in several different ways. The most straightforward method, which can be easily implemented is computing gradients by finite differences. It is sensitive to how the policy is parameterized, noisy and inefficient, but it works even if the policy is not differentiable.

D. Policy Gradient Method

We have applied a policy gradient method based on the Williams's theory of REINFORCE algorithms (see [15] and policy gradient theorem of Sutton et. al. (see [14]). This theorem generalizes the likelihood approach to multi-step MDPs. In one-step MDP the likelihood ratios are used to compute the policy gradient

$$\nabla_\theta J(\theta) = \mathbf{E}[\nabla_\theta \log \pi_\theta(s,a) r] \tag{4}$$

where r is the one time-step reward. In multi-step MDP the immediate reward is replaced with the long-term value

$$\nabla_\theta J(\theta) = \mathbf{E}\big[\nabla_\theta \log \pi_\theta(s,a) Q^\pi(s,a)\big]. \tag{5}$$

Accordingly the policy gradients can be calculated and the parameters can be updated, thus the core algorithm works as follows

1) Initialize θ and the starting state $s1$
2) Run an episode until terminated and store history
3) Discount rewards $r1, r2, ..., r_T$
4) For each episode $s_1, a_1, r_2, ..., s_{T-1}, r_T$ update the gradients:
   $\theta \leftarrow \theta + \alpha \nabla \log \pi_\theta(s_t, a_t) v_t$

In our implementation the policy is approximated with an artificial neural network with fully connected hidden layers using ReLu activations. The input layer receives the state observation and the outputs are the probabilities of the discrete actions. Vector v_t contains the discounted reward of the episode and the gradients are estimated as in [16], thus the loss function for the neural network optimization can be formalized as follows

$$Loss = -1/T \sum_{t=0}^{T} \theta \log \pi_\theta(a_t, s_t) v_t \tag{6}$$

Since the noisy gradient updates can cause instable convergence, the parameters θ are not updated after each episodes, but accumulated over a number of episodes and then applied to the network. It can reduce the gradient variance which might stabilize the learning.

III. TRAINING ENVIRONMENT

As previously mentioned the agent needs an environment where it can act and learn to drive. Such environment must consist at least the following subsystems:

- Vehicle model, including vehicle dynamics, sensor model and actuation:
- Highway model, describing topology, lane geometry, traffic generation, behavior for environmental vehicles etc.
- Reward function, where a positive value is given for the expected behavior and a negative value is given for undesired decisions.

A. Vehicle model

The model of the agent's vehicle is a rigid kinematic single-track vehicle model, where tire slip is neglected. Thus, lateral motion is only affected by the geometric parameters. More details about the model can be found in [17], [18]. This model seems quite simplified, and as stated in [19], such model can behave significantly different from an actual vehicle, though for the many control situations, the accuracy is suitable [18]. The main parameters of the model are: L-axle length. State parameters are x, y vehicle position, v longitudinal speed and ψ yaw angle. Control parameters are the acceleration demand, and the δ steering angle. By neglecting tire slip, the path radius R can be calculated as:

$$\tan \delta = \frac{L}{R} \tag{7}$$

Thus, the yaw rate is:

$$\dot{\psi} = \frac{v}{R} = \tan \delta \frac{v}{L} \tag{8}$$

B. Environment sensing

The environment sensing model of the agent assumes high level output from common automotive sensors, such as camera and radar based systems. This way the environment information that is available for the agent is its position and heading relative to the highway lane, and also the relative sates of the vehicles in the ego and neighboring lanes. These information together with the vehicle state provides the state input of the agent. The perception system is similar to that proposed in [20] with minor differences. Table I summarizes the state vector, while Fig. 2 gives an overview for better understanding. IDs $[0-11]$ provides relative distance and relative speed of the surrounding vehicles. In the case where the lane is unoccupied or not exists (e.g. when the agent is in the rightmost lane) the $[dx, dv]$ distance and relative speed values are filled with $[500, 0]$. The IDs $[12, 13]$ show the occupancy of the lane on the two sides of the agent while $[14-16]$ give information about the lateral position relative to the mid-line of the rightmost lane, the vehicle's heading relative to the orientation of the highway and the vehicle speed.

TABLE I
ENVIRONMENT STATE VECTOR DESCRIPTION

ID	Meaning	Elements
0,1	Front Left Lane	dx,dv
2,3	Front Ego Lane	dx,dv
4,5	Front Right Lane	dx,dv
6,7	Rear Left Lane	dx,dv
8,9	Rear Ego Lane	dx,dv
10,11	Rear Right Lane	dx,dv
12	Left Safe Zone	Occup [0,1]
13	Right Safe Zone	Occup [0,1]
14	Vehicle lateral position	y [m]
15	Vehicle heading	Ψ[rad]
16	Vehicle speed	v[m/s]

C. Traffic generation

In this research, a highway with constant lane count is considered through the simulation, and for each lane, the generation of the traffic is based on previously determined density. The parameters are the mean and deviation of the generated traffic density on the lane, and also the mean and deviation of the desired speed of the generated vehicles on

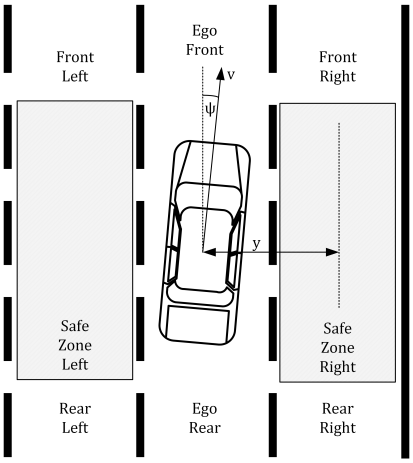

Fig. 2. Environment state on the highway

these lanes $[m(k_i), \sigma(k_i), m(v_i), \sigma(v_i)]$. These parameters also vary randomly throughout the learning process, to keep the diversity of the situations. At this stage of the research, no lane changing behavior is implemented. Therefore all environmental vehicles stay at their lane through the episode. To achieve various states, every simulation episode starts with a warm-up stage, where for a simulated 2 minutes, only environmental vehicles take place, to form a close to real traffic situation. After the warm up, a vehicle in the middle of highway is chosen to be the agent for learning. Throughout the simulation the environmental vehicles try to reach their desired speed while preventing collisions they also apply car-following rules. The car-following model is similar to the one described in [21]: first, the desired inter-distance of the j_{th} vehicle on the i_{th} lane $d^j_{des,i}$ is determined based on the speed of the leading vehicle:

$$d^j_{des,i} = \alpha^j_i v^{j+1}_i, \tag{9}$$

where v^{j+1}_i is the speed of the leading vehicle in m/s and α^j_i is the sensitivity parameter with random values from $\mathcal{N}(1.3, 0.02)$. Afterwards a PD controller, calculates the acceleration command of the car-following based on the following distance, and relative speed:

$$a^j_{cf,i} = K_p(x^{j+1}_i - x^j_i - d^j_{des,i}) + K_d(v^{j+1}_i - v^j_i), \tag{10}$$

where K_p and K_d are the gains of the car-following controller. The final acceleration command of the environmental vehicles are the minimum of the car-following and the desired speed based accelerations, saturated by the physical constraints.

D. Simulation and Reward Calculation

In each training step, the agent receives the state vector defined by Table I, and determines its action, the acceleration, and steering demand. Then, the simulation actuates these inputs to the agent's vehicle and also simulates one step for each environmental vehicle on the examined highway section. At each step, the traffic density of the each lanes is evaluated. If it is lower than the desired one, a new vehicle is added to the lane in the following way: if the average speed of the lane is below the desired one, the vehicle is added at the end, and conversely. Each episode of the training process lasts for an arbitrarily chosen 500 seconds unless a terminating condition stops it.

To define the reward function for the agent, the following requirements were considered, from which terminating conditions are:

- The vehicle leaves the highway.
- The vehicles collide.
- The vehicle stops or its speed is lower than a previously defined threshold (only applied when the low speed is not caused by traffic conditions).

Besides terminating conditions the following requirements describes the quality features of the performance of the agent:

- The vehicle should not travel between the lanes for a long time.
- The agent should travel at its desired speed, when possible.
- The vehicle should stay on the right most lane if not overtaking.

When a terminating condition rises, the episode is stopped, and the agent is given a high negative reward ($R \approx -100$). Otherwise, the reward is calculated from three factors: To evaluate the "Keep right" rule, the right neighboring lane is examined, through the values of "Right Safe Zone occupied" $ID[13]$ and Right Lane distance dx, $ID[4]$ from the state vector. In case, the lane is occupied in the defined close range d_l, the rule's reward is ($R_l = 1$), in case there is no vehicle in far range d_h, $R_l = 0$, otherwise the reward gets a value based on the distance of the vehicle:

$$R_l = \begin{cases} 1, & \text{if } dx < d_l \\ 0, & \text{if } dx > d_h \\ 1 - \frac{dx - d_l}{d_h - d_l}, & \text{otherwise} \end{cases} \tag{11}$$

Traveling between lanes is not rewarded, since previous training sessions with these kind of rewards proved, that collisions prevent the vehicle from this behavior. On the other hand staying on the highway is rewarded considering the lateral distance from either side of the highway y_d, with thresholds $[y_l, y_h]$:

$$R_y = \begin{cases} 0, & \text{if } y_d < y_l \\ 1, & \text{if } y_d > y_h \\ 1 - \frac{y_d - y_l}{y_h - y_l}, & \text{otherwise} \end{cases} \tag{12}$$

Finally, keeping the desired speed is rewarded by the trapezoid shaped function, considering the speed of the agent v, the desired speed v_d with thresholds $[v_l, v_h]$:

$$R_v = \begin{cases} 0, & \text{if } |v - v_d| > v_h \\ 1, & \text{if } |v - v_d| < v_l \\ 1 - \frac{|v - v_d| - v_l}{v_h - v_l}, & \text{otherwise} \end{cases} \tag{13}$$

Combining (11), (12) and (13) the final reward is calculated as the weighted average of the $[R_y, R_l, R_v]$ with

parameters $[\alpha_y, \alpha_l, \alpha_v]$:

$$R = \alpha_y R_y + \alpha_l R_l + \alpha_v R_v, \text{ where} \quad (14)$$

$$\alpha_y + \alpha_l + \alpha_v = 1 \quad (15)$$

In case of an incident the episode is terminated and a negative value (μ) is added to the cumulated reward. The termination causes are leaving the highway, collisions and driving at low speed.

IV. RESULTS

In machine learning the training process is usually carried out in an iterative way, the success of the agent's trial-and-error exploration of the possible actions is highly affected by the hyperparameters of the algorithm. Hence, compared to supervised learning the reinforcement learning process may be more complicated.

In the present case the most significant hyperparameters are the learning rate (α), the discount factor (γ) and the number of episodes (ξ) after the which parameters θ are updated. Further parameters are the subreward weights ($\alpha_y, \alpha_l, \alpha_v$) and the penalty constant (μ). The hyperparameters of the policy approximator neural network remained constant during the iterations. During the development process it became clear how strongly the reward function shape and parameters affect the learning and the results. After several iterations the chosen hyperparameters and reward function parameters are summarized in Table II.

TABLE II

HYPERPARAMETERS AND REWARD FUNCTION PARAMETERS

Parameter	Value
Learning rate (α)	0.0001
Discount factor (γ)	0.95
Num. of ep. after the params are upd. (ξ)	5
Position reward weight (α_y)	0.05
Lane reward weight (α_l)	0.4
Velocity reward weight (α_v)	0.55
Penalty constant (μ)	-80
Num. of hidden layers	2
Num. of neurons	256,256

The following figures show the result after 300,000 episodes training. During the learning 100 evaluation episodes were executed after every 1000 training episodes. In training phase the action is selected by a non-uniform random choice according to the probabilities given by the policy approximator, which ensures the exploration explicitly. While in the testing phase the action is determined by greedy algorithm.

Fig. 3 shows the trend of the training reward, smoothed by moving average using window length of 100 episodes.

The subrewards during the evaluation periods can be seen in Fig. 4. As it can be seen the position reward set to a nearly constant value with minimal noise after approx. 10,000 episodes. The lane reward is noisier and its graph is steeper, but after approx. 100,000 episodes the agent has learnt lane keeping. During the learning process, the agent first formed the appropriate acceleration model, then

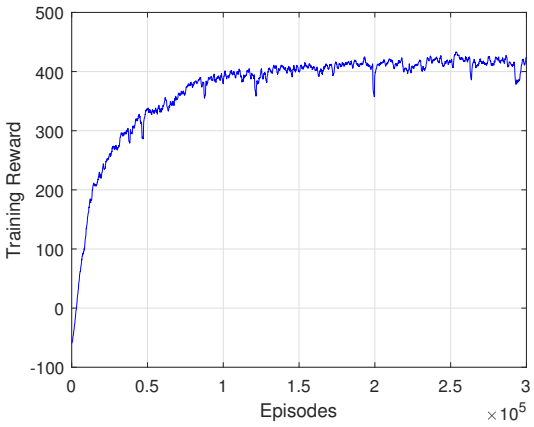

Fig. 3. Training rewards

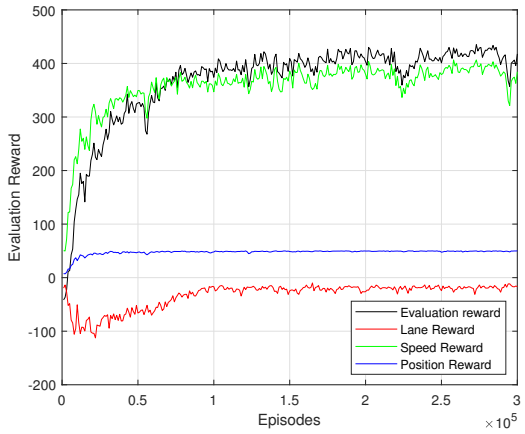

Fig. 4. Evaluation rewards during training

to further maximize the reward, the proper lane choice were carried out.

Finally one of the safety aspects is examined, namely the the termination cause ratio, which is shown in Fig. 5. Although the main conclusion can be deduced from the diagram, a short numerical summary is stated in Table III.

TABLE III

TERMINATION CAUSES OF THE LAST 3000 EVALUATION PERIODS

Cause	Value	Percentage
Good	2968	98.9333
Low speed	2	0.0667
Left highway	2	0.0667
Front collision	21	0.7
Rear collision	7	0.2333

The values show that the front collision significantly exceeds the other values. This means that in the current state of the agent's learning, incident occurs approximately once in every 360km. Though this value is quite high, it does not reach the requirements of safe driving. To evaluate the causes at such complex behavior, visual inspection is needed through many selected animated evaluation episodes

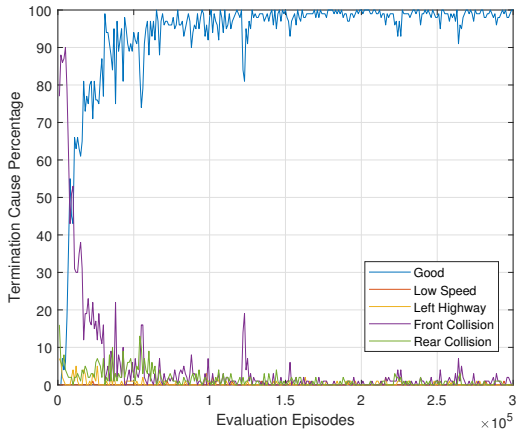

Fig. 5. Termination causes during evaluation

(see screenshot in Fig. 6) with critical traffic conditions. The experiments showed that the agent sometimes keeps too short distance to the leading vehicle, typically in lane changing situations, probably because it is trying to maximize the velocity reward and sometimes it can not decelerate in time.

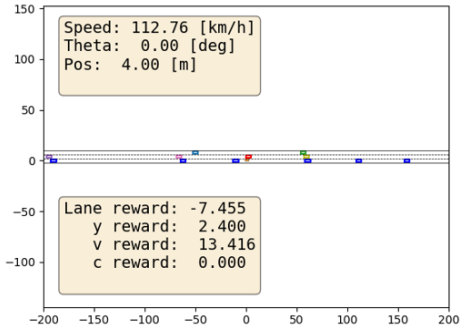

Fig. 6. Animation of the highway traffic, ego car marked with red

V. CONCLUSIONS

The paper presented a possible end-to-end vehicle control based on high level sensor information, with the application of policy gradient based reinforcement learning. The research showed that reinforcement learning is a strong tool for designing complex behavior on traffic situations, such as highway driving, where multiple objectives are present, such as lane keeping, keeping right, avoiding incidents while maintaining a target speed.

As the initial step of the research, the algorithm shows convergence during the learning hence the trained agent is basically capable to control a vehicle in a simplified highway environment. The visual inspection of the situations shows that the overall behavior of the agent fulfills the requirements, though in some cases safe driving is not achieved. With the utilization of the simple rewarding system presented in the paper, the agent established complex behavior schemes such as target gap acceleration or overtaking.

Further research will focus on the design of learning model and rewarding system that ensures safe driving. In the future we plan to extend the method to continuous action space and to improve and publish the environment as an open-source project.

REFERENCES

[1] SAE International, *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, 2016.

[2] G. Rödönyi, P. Gáspár, J. Bokor, and L. Palkovics, "Experimental verification of robustness in a semi-autonomous heavy vehicle platoon," *Control Engineering Practice*, vol. 28, no. Supplement C, pp. 13 – 25, 2014.

[3] Z. Chen and X. Huang, "End-to-end learning for lane keeping of self-driving cars," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 1856–1860.

[4] V. e. a. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529 EP –, Feb 2015.

[5] D. e. a. Silver, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354 EP –, Oct 2017.

[6] C. Desjardins and B. Chaib-draa, "Cooperative adaptive cruise control: A reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1248–1260, Dec 2011.

[7] X. Li, X. Xu, and L. Zuo, "Reinforcement learning based overtaking decision-making for highway autonomous driving," in *2015 Sixth International Conference on Intelligent Control and Information Processing (ICICIP)*, Nov 2015, pp. 336–342.

[8] B. Mirchevska, M. Blum, L. Louis, J. Boedecker, and M. Werling, "Reinforcement learning for autonomous maneuvering in highway scenarios," in *11. Uni-DAS e.V. Workshop Fahrerassistenz und automatisiertes Fahren*, March 2017.

[9] P. Wang and C. Chan, "Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge," *CoRR*, vol. abs/1709.02066, 2017.

[10] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," *CoRR*, vol. abs/1612.04340, 2016.

[11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[12] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015.

[13] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," *CoRR*, vol. abs/1511.06581, 2015.

[14] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, pp. 1057–1063, 2000.

[15] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[16] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 2219–2225.

[17] O. Töro, T. Bécsi, and S. Aradi, "Design of lane keeping algorithm of autonomous vehicle," *Periodica Polytechnica. Transportation Engineering*, vol. 44, no. 1, p. 60, 2016.

[18] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 1094–1099.

[19] P. Polack, F. Altché, B. D'Andréa-Novel, and A. De La Fortelle, "The kinematic bicycle model: a consistent model for planning feasible trajectories for autonomous vehicles?" in *IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[20] M. Zhang, N. Li, A. Girard, and I. Kolmanovsky, "A finite state machine based automated driving controller and its stochastic optimization," in *ASME 2017*, 2017.

[21] X. Fang, H. A. Pham, and M. Kobayashi, "Pd controller for car-following models based on real data," in *Proceedings of 1st Human-Centered Transportation Simulation Conference*, vol. 2001, 2001.