Hybrid DDPG Approach for Vehicle Motion Planning

Árpád Fehér[1][a], Szilárd Aradi[1][b], Ferenc Hegedűs[1][c], Tamás Bécsi[1][d] and Péter Gáspár[2][e]
[1] *Department of Control for Transportation and Vehicle Systems, Budapest University of Technology and Economics, Budapest, Hungary*
[2] *Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest, Hungary*

Keywords: Reinforcement Learning, Motion Planning, Autonomous Vehicles, Automotive Control.

Abstract: The paper presents a motion planning solution which combines classic control techniques with machine learning. For this task, a reinforcement learning environment has been created, where the quality of the fulfilment of the designed path by a classic control loop provides the reward function. System dynamics is described by a nonlinear planar single track vehicle model with dynamic wheel mode model. The goodness of the planned trajectory is evaluated by driving the vehicle along the track. The paper shows that this encapsulated problem and environment provides a one-step reinforcement learning task with continuous actions that can be handled with Deep Deterministic Policy Gradient learning agent. The solution of the problem provides a real-time neural network-based motion planner along with a tracking algorithm, and since the trained network provides a preliminary estimate on the expected reward of the current state-action pair, the system acts as a trajectory feasibility estimator as well.

1 INTRODUCTION AND MOTIVATION

Highly automated and autonomous driving is expected to enhance the quality of road transportation in multiple aspects, such as increasing the level of safety while reducing fuel consumption and emissions. The development potential makes the topic one of the most intense research fields both for vehicle industry and related academic institutions. This paper deals with the problem of feasible motion planning, i.e. the design and evaluation of the trajectory that the vehicle must follow.

Many different approaches have been evolved over the years to solve the motion planning problem for wheeled vehicles, all having advantages and drawbacks as well. Geometric approaches assemble the path of the vehicle from geometric curves as clothoids, circular arcs and splines. A popular choice is to define curvature as function of arc length (Li et al., 2015). They are often used in simple low-dynamic scenarios e.g. automatic parking (Vorobieva et al., 2013). While these algorithms are computationally cheap, the ability to consider the nonholonomic dynamics of the vehicle is limited to the usage of maximal steering angle and geometric acceleration constraints (Minh and Pumwa, 2014). Other popular methods used for trajectory planning are based on graph search techniques. The configuration space (space of possible states) of the vehicle is discretized or sampled in a random manner to build a graph of safely reachable and unoccupied states (Palmieri et al., 2016). The shortest connection defined by a suitably chosen metric is then searched along the graph via some heuristics (Gammell et al., 2015). The formulation of graph search based methods makes it easy to deal with collision avoidance, but vehicle dynamics considerations are again hard to incorporate. Variational methods are formulating the motion planning as a nonlinear optimization problem which enables the usage of almost arbitrary vehicle models (Singh et al., 2017). These methods are proven to be able to generate dynamically feasible trajectories even in case of high-dynamic scenarios, but this comes at a price of high computational requirements which often make real-time applications impossible (Hegedüs et al., 2017a).

Besides the classical methods, approaches based

[a] https://orcid.org/0000-0002-9491-4211
[b] https://orcid.org/0000-0001-6811-2584
[c] https://orcid.org/0000-0002-8063-6054
[d] https://orcid.org/0000-0002-1487-9672
[e] https://orcid.org/0000-0003-3388-1724

Fehér, Á., Aradi, S., Hegedűs, F., Bécsi, T. and Gáspár, P.
Hybrid DDPG Approach for Vehicle Motion Planning.
DOI: 10.5220/0007955504220429
In *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2019)*, pages 422-429
ISBN: 978-989-758-380-3

on artificial neural networks gain more and more interest thanks to their high performance in learning, adaptation and generalization. Supervised learning techniques were used for motion prediction of road vehicles (Yim and Oh, 2004) as well as motor control for industrial robots in dynamic environments (Liu et al., 2017). In recent years, reinforcement learning (RL) was also used successfully for motion planning of car-like mobile robots. In (Tai et al., 2017) authors offer a method for path planning of a mobile robot to reach a specified target position without having a-priori map information, while (Chen et al., 2017) deals with motion planning in pedestrian-rich environments. In (Li et al., 2019) continuous lateral control for racetrack simulation is taught, in (Paxton et al., 2017) the authors combine the MTCS method with RL techniques for simple maneuvers.

The main drawback of classical optimization based techniques despite their outstanding performance is the necessity of computationally intensive on-line optimization considering complex vehicle dynamics. With the application of reinforcement learning it is however possible to teach an artificial neural network how to drive a vehicle model with same level complexity in an optimal way. With this approach the computationally demanding tasks can be carried out off-line (Plessen, 2019). The motivation of the paper is to create a trajectory planning and tracking algorithm especially for road vehicles that can provide dynamically feasible motions under real-time constraints.

The DDPG planner presented trains itself for the optimal trajectory planning problem with predefined initial and end states as described in Section 3.1 , without considering any obstacles, though regard to dynamics described in Section 3.2. The output of the system is a detailed trajectory curve which can be followed by a lateral controller. The evaluation of the resulted control loop considers angle and distance errors, and side slip as a measure of feasibility.

2 PLANNER DESIGN WITH DEEP REINFORCEMENT LEARNING

2.1 Reinforcement Learning

In problems like the one discussed in this paper, the training of the Artificial Neural Network (ANN) lacks training data, hence the machine learning process needs to generate its own experiences through trial-and-error forming a reinforcement learning framework. In this area, the learner and decision maker algorithm is called the *agent*. Everything outside the agent is called the *environment*. The environment shall provide the following information to the agent:

- state (output)
- action (input)
- reward (output)

The learning process consists of episodes, which is a solution attempt for the original problem with a given set of initial parameters, and generally an episode consists of a series of steps. The agent interacts with the environment, and based on the state information provided, it selects actions, resulting in a new state representing the new situation in every step. Furthermore, the environment provides information about how well the agent does its job as a scalar value, called the reward.

An overview of the developed trajectory designer can be seen in Fig. 1: In each episode, the agent receives the initial conditions and the target for trajectory planning and calculates the interior points of the trajectory, then we drive a vehicle along the planned route (control loop), while its performance is evaluated. The reward value after the evaluation is received by the learning agent. Thereafter the process starts from the beginning. This is a one-step return learning task, meaning that an episode consists of one step and does not considers the next state (gray in Fig. 1) which reduces the complexity of the learning.

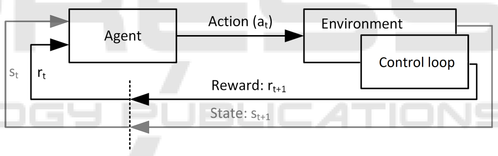

Figure 1: Agent-environment interaction in reinforcement learning.

2.2 Deep Deterministic Policy Gradient Method

In our previous studies, we trained reinforced learning agents in vehicular tasks (Bécsi et al., 2018)(Fehér et al., 2018)(Aradi et al., 2018) where the agents control the environment through discreet actions, but most vehicle control tasks and the motion planning environment must be controlled by continuous actions. We have chosen a relatively easy-to-implement but well-performing learning agent for this continuous approach, called Deep Deterministic Policy Gradient (DDPG). It is a model-free, off-policy actor-critic algorithm using deep function approximators that can learn policies in high-dimensional, continuous action spaces (Lillicrap et al., 2015). It is based on the deterministic policy gradient (DPG) algorithm (Silver et al., 2014). The actor $\mu(s|\theta^{\mu})$ specifies the current policy by deterministically mapping states to a specific action and the critic $Q(s,a)$ use the Bellman equation. The actor is updated by the a following rule:

$$\nabla_{\theta\mu} J \approx E_{S_t \sim \rho^{\beta}}[\nabla_{\theta\mu} Q(s,a|\theta^{Q})|_{s=s_t, a=\mu(St|\theta^{\mu})}] \quad (1)$$

3 TRAINING ENVIRONMENT

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

- Feasible conditions based trajectory generator module
- Nonlinear planar single track vehicle model with dynamic wheel model
- Longitudinal and lateral control
- Reward calculation

3.1 Trajectory Generation

The trajectory planning task works with the inputs of: the vehicle state at the start and also the desired end state. Based on these information, the learning agent determines the intermediate points of the trajectory.

We give an example case for the training, where the initial state vector (2) of position and yaw angle are fixed to the position of the vehicle, and a constant speed of ($90km/h$) is chosen as a typical speed for main roads . The final state (3) is evenly distributed random vector drawn from a set of states, that are bit wider than the feasible targets (3). Too many sample from unfeasible target end-states could lengthen the learning process and hence need to be avoided, though some is beneficial to learn the boundaries.

$$\begin{bmatrix} x_s & y_s & \psi_s & v_s \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 0 & 25\mathrm{m/s} \end{bmatrix}^T \quad (2)$$

$$\begin{bmatrix} x_e \\ y_e \\ \psi_e \\ v_e \end{bmatrix} = \begin{bmatrix} 3 * v_{start} \\ \mathrm{rand}(-y_{max}, y_{max}) \\ 0.1 * \psi_{max} + \mathrm{rand}(0,1) * 1.3 * \psi_{max} \\ v_{start} \end{bmatrix} \quad (3)$$

$$R_{min} = 0.1207 * v_{start}{}^{2.4736} \quad (4)$$

$$y_{max} = R_{min} - \sqrt{R_{min}{}^2 - x_e{}^2} \quad (5)$$

$$\psi_{max} = -2 * \arctan(y_e/x_e)) \quad (6)$$

The planned trajectory is validated by a dynamic vehicle model. The feasible final state can be determined by an empirical formula (4) as a rule of a thumb, which gives the smallest arc radius that an average vehicle can take at fix speed under normal conditions.

Determining the initial and the end state , the learning agent determines y coordinate of two intermediate points, placed equally between the initial and the end points along the x coordinate. A spline is inserted based on the four holding points, taking into account the initial and end gradients, which gives the desired trajectory.

3.2 Vehicle Model

In order to provide an accurate prediction of the vehicle's behavior at fair computational requirements, a nonlinear planar single track vehicle model containing a dynamic wheel model as well is applied. This model can deliver feasible results even in case of high-dynamic driving maneuvers, but is simple enough to keep its run time at a suitable level (Hegedüs et al., 2017b).

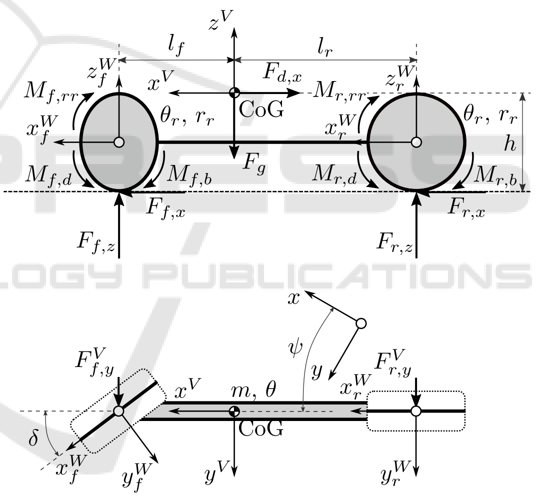

Figure 2: Nonlinear single track vehicle model.

The multi-body model (Fig. 2) consists of the vehicle chassis and two virtual wheels connected rigidly representing the front and rear axles. The main parameters are the mass m and moment of inertia θ of the chassis, the horizontal distances between the vehicle's center of gravity and the front and rear wheel centers l_f and l_r, the center of gravity height of the vehicle h, the moments of inertia $\theta_{[f/r]}$ as well as the radii $r_{[f/r]}$ of the front and rear wheels. The parameters of the wheel models have also a vital influence, the most important ones are the coefficient of friction $\mu_{[f/r]}$ and the parameters of the Magic Formula slip curves $C_{[f/r],[x/y]}$, $B_{[f/r],[x/y]}$, $E_{[f/r],[x/y]}$ which influence the transmittable amount of force between road and tires (Pacejka, 2012).

The inputs of the model are the steering angle of front wheel δ (the rear wheel is considered unsteered) and the total driving M_d and braking M_b torques applied at the wheels (no powertrain is modelled). The driving torque is distributed to the front and rear axles $M_{[f/r],d}$ by time-varying distribution factor ξ_M. For the braking torque, ideal distribution $M_{[f/r],b}$ is considered which maintains equal brake slips.

The chassis can move longitudinally x and laterally y and rotate ψ about its vertical axis (yaw movement). The wheels can only rotate $\phi_{[f/r]}$ about their own horizontal axes, and their longitudinal and lateral slips $s_{[f/r],[x/y]}$ are modelled dynamically. In the following superscripts are used to distinguish dynamic quantities in ground-fixed (no superscript), vehicle-fixed (V) and wheel-fixed (W) coordinate systems and dot notation ($\dot{\_}$) is used for time derivatives.

The dynamic equations for the chassis are derived in the ground-fixed inertial coordinate system using Newton's second law for translation and rotation as follows:

$$\ddot{x} = \frac{1}{m}(F_{f,x} + F_{r,x} + F_{d,x}), \tag{7}$$

$$\ddot{y} = \frac{1}{m}(F_{f,y} + F_{r,y} + F_{d,y}), \tag{8}$$

$$\ddot{\psi} = \frac{1}{\theta}(l_f F^V_{f,y} - l_r F^V_{r,y}), \tag{9}$$

where $F_{[f/r],[x/y]}$ are tire forces. The aerodynamic drag forces are calculated as:

$$F^V_{d,x} = \frac{1}{2} c_D A_f \rho_A \dot{x}^V \sqrt{\dot{x}^V + \dot{y}^V}, \tag{10}$$

$$F^V_{d,y} = \frac{1}{2} c_D A_f \rho_A \dot{y}^V \sqrt{\dot{x}^V + \dot{y}^V}, \tag{11}$$

where c_D is the drag coefficient and A_f is the frontal area of the vehicle, and ρ_A is the mass density of air.

The tire forces can be derived considering the motion of the wheels. The front and rear wheels are modelled equally, so only the equations for the front one are presented. The dynamic equations of the front wheel using Newton's second law for rotation and dynamic slip equations from (Pacejka, 2012) are the following:

$$\ddot{\phi}_f = \frac{1}{\theta_f}\Big(M_{f,d} - r_f F^W_{f,x} - M_{f,b} - M_{f,rr}\Big), \tag{12}$$

$$\dot{s}_{f,x} = \frac{1}{l_{f,x}}\Big(r_f \dot{\phi}_f - \dot{x}^W_f - |\dot{x}^W_f| s_{f,x}\Big), \tag{13}$$

$$\dot{s}_{f,y} = \frac{1}{l_{f,y}}\Big(-\dot{y}^W_f - |\dot{x}^W_f| s_{f,y}\Big), \tag{14}$$

where $\dot{x}^W_f$ and $\dot{y}^W_f$ are the longitudinal and lateral velocities of wheel center. The longitudinal and lateral slip dependent relaxation lengths are:

$$\begin{aligned} l_{f,[x/y]} =& max\Bigg(l_{f,[x/y],0}\left[1 - \frac{B_{f,[x/y]}C_{f,[x/y]}}{3}|s_{f,[x/y]}|\right], \\ & l_{f,[x/y],min}\Bigg), \end{aligned} \tag{15}$$

where $l_{f,[x/y],0}$ is the values at standstill and $l_{f,[x/y],min}$ are the values at at wheel spin or wheel lock. The rolling resistance torque $M_{f,rr}$ is calculated according to standard SAE J2452. The longitudinal and lateral tire forces are calculated by the Magic Formula:

$$\begin{aligned} \tilde{F}^W_{f,[x/y]} =& \mu_f F^W_{f,z} \sin\{C_{f,[x/y]} \arctan(B_{f,[x/y]}\tilde{s}_{f,[x/y]} - \\ & E[B_{f,[x/y]}\tilde{s}_{f,[x/y]} - \arctan(B_{f,[x/y]}\tilde{s}_{f,[x/y]})])\}. \end{aligned} \tag{16}$$

For the force calculation, damped slip values are used to improve the stability of numerical solution:

$$\tilde{s}_{f,x} = s_{f_x} + \frac{k_{f,x}}{B_{f,x}C_{f,x}\mu_f F^W_{f,z}}(r_f \dot{\rho}_f - \dot{x}^W_f), \tag{17}$$

$$\tilde{s}_{f,y} = s_{f,y}, \tag{18}$$

where $k_{f,x}$ is the velocity dependent damping factor calculated as:

$$k_{f,x} = \begin{cases} \frac{1}{2}k_{f,x,0}\Big(1 + \cos\Big(\pi\frac{|\dot{x}^W_f|}{v_{low}}\Big)\Big), \text{if } \dot{x}^W_f \leq v_{low} \\ 0, \text{if } \dot{x}^W_f > v_{low}, \end{cases} \tag{19}$$

with $k_{f,x,0}$ being the damping value at zero velocity and v_{low} being the velocity at which damping is switched off. The superposition of longitudinal and lateral forces is considered using the fiction ellipse method:

$$\begin{aligned} F^W_{f,x} &= \text{sign}(\tilde{s}_{f,x})\sqrt{\frac{(\tilde{F}^W_{f,x}\tilde{F}^W_{f,y})^2}{(\tilde{F}^W_{f,y})^2 + (\frac{\tilde{s}_{f,y}}{\tilde{s}_{f,x}}\tilde{F}^W_{f,x})^2}} \\ F^W_{f,y} &= \text{sign}(\tilde{s}_{f,y})\sqrt{\frac{(\tilde{F}^W_{f,x}\tilde{F}^W_{f,y})^2}{(\tilde{F}^W_{f,x})^2 + (\frac{\tilde{s}_{f,x}}{\tilde{s}_{f,y}}\tilde{F}^W_{f,y})^2}} \end{aligned} \tag{20}$$

The presented wheel model enables the usage of explicit ODE (Ordinary Differential Equation) solvers (e.g. the 4th order Runge-Kutta method) with a moderate step size of approximately 1 ms. The model was originally implemented in Python, but even with this time step the run time was infeasible considering the large amount of iterations in the learning process. Because of this, the vehicle model as well as the solver was implemented in C which resulted in a tenfold increase in speed approximately.

3.3 Longitudinal and Lateral Control

For driving along the vehicle on the trajectory we developed a longitudinal and lateral control. The beginning of the episode, the vehicle does not start with 0 km/h, therefore in order to get stabilized states the vehicle modeluses a warm up distance to reach the initial state. For longitudinal control tasks a simple PID can effectively handle the problem. The Stanley method (Thrun et al., 1970) is used for the lateral control.

$$\delta = -\left(\psi + \arctan\left(k * \frac{y}{v}\right)\right) \tag{21}$$

where ψ is the yaw error at the front axle, y is the lateral error at the front axle, v is the vehicle speed (computed at the front axle, it's direction is parallel to the front wheel) and k is the gain factor.

At the output of the Stanley controller, speed-sensitive saturation was applied.

3.4 Reward Calculation

In each training step, the agent receives the state vector (initial conditions of the trajectory) and determines its actions, the intermediate points. To calculate the reward, the vehicle goes along the trajectory using the internal lateral and longitudinal controls. Each episode of the training process lasts as long as the vehicle does not reach the end of the trajectory unless a terminating condition stops it.

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

- The lateral distance error is more than 10 meters
- The longitudinal or lateral slip is higher than 0.1
- The maximum number of steps is more than 2500
- The (Yaw) angle error is more than 0.2 radians

Besides terminating conditions the sum slip and the angular and distance deviation requirements describe the quality features of the performance of the agent. The episode reward consists of three weighted components.

$$R_{episode} = s_w * R_{slip} + d_w * R_{dist} + a_w * R_{angle} \tag{22}$$

The environment defines 10 checkpoints (cp) equally distributed on the trajectory. The distance (R_{dist}) and the angle (R_{angle}) rewards were calculated at the checkpoints and the slip reward (R_{slip}) was calculated at all time step. The subreward values are defined to be in range $[0,3]$ and are calculated as follows:

$$R_{slip} = 3 - \sum_{step=1}^{max\ step} -abs(max(s_{[f/r],[x/y]}))/10 \tag{23}$$

$$R_{angle} = 3 - \sum_{cp=1}^{10} -abs(\psi) * 2 * pos \tag{24}$$

$$R_{dist} = 3 - \sum_{cp=1}^{10} -abs(y/3) * 2 * pos \tag{25}$$

Where ψ is the yaw error at the front axle, y is the lateral error and the pos the vehicle position on the trajectory. The inital value and the equations are determined by experience. When a terminating condition rises, the episode is stopped, and the agent is given a negative reward ($R \approx -10$). The environment includes a reset method to restore the vehicle to its initial position.

4 RESULTS

Reinforcement learning algorithms usually needs a lot of iteration. The success of the training process depends on many parameters. It is highly affected by the hyperparameters of the training algorithm, and - in the recent case - the efficiency of the longitudinal and lateral control, the feasible conditions from the trajectory generator module and the consistent reward function are also influential.

In the present case, the most significant hyperparameters are the actor and the critic network learning rates (α_a), (α_c), the action bound factors (a_{fn}), (a_{ff}) and the Ornstein-Uhlenbeck noise parameters (μ), (σ), (θ).

The hyperparameters of the neural network kept 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 are summarized in Table 1. The following figures show the result after 80000 episodes training. After approx. 40000 the agent started to produce trajectories of good quality. Fig. 3 shows the trend of the max Q-value, smoothed by moving average using window length of 21 episodes. The diagram shows that the max q-value are stabilized. The critic network learned the reward function well. The evaluation of the learning is almost perfect based on the Density plot (Fig. 4) which shows the estimated Q values versus the actual rewards, as sampled from test episodes. The diagram shows strong positive correlation.

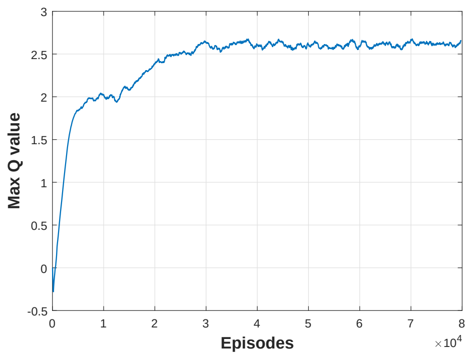

Figure 3: Training Q-values.

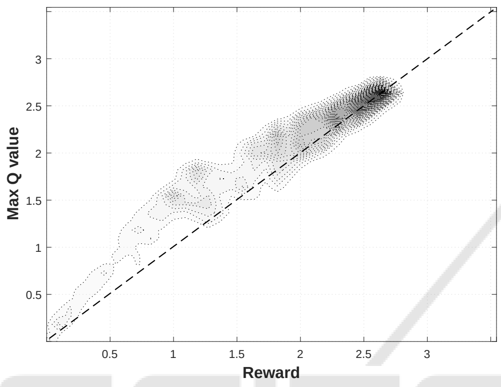

Figure 4: Density plot.

For the performance evaluation of the learned agent we differentiate two type of situations represented by different trajectory types. The first one (Fig. 6) is when the vehicle needs to turn the second one (Fig. 5) which is the avoiding situation. In the first case, the target angle is a higher value, however in case of the second, it is close to zero. The target angle of the turning maneuver depends on the distance.

Table 1: Hyperparameters.

Actor network	
Learning rate (α)	0.0001
Batch size	64
Hidden F.C. layer structure	[128,100,64]
Activation function	relu
Output scale factor	[2,4]
Critic network	
Learning rate (α)	0.001
Discount factor (γ)	0.99
Hidden F.C. layer structure	[128,64]
Activation function	relu
Ornstein-Uhlenbeck parameters	
(μ)	[0,0]
(σ)	0.3
(θ)	0.15

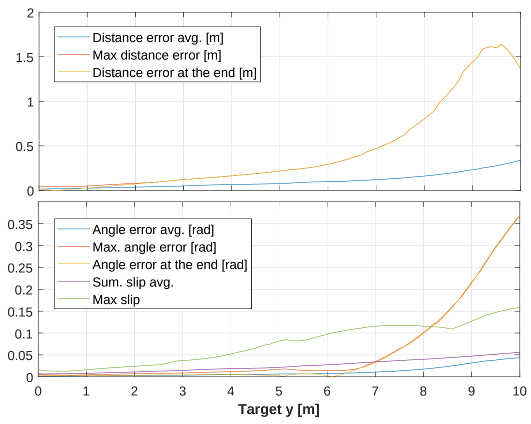

Figure 5: Avoidance maneuver performance.

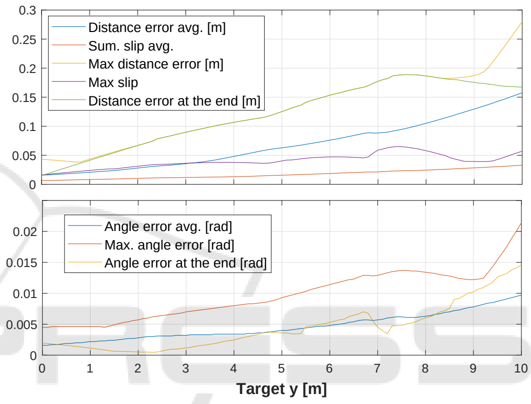

Figure 6: Turning maneuver performance.

Considering the previously defined maximum speed the evaluation of these cases were carried out at vehicle speed of $90 km/h$ with a test set that is slightly broader than the area which was considered theoretically feasible beforehand. The diagrams show that the planner solves these situations well. The diagrams also show that, the harder the situation gets, the error of the realization also increases. Especially at the case of the avoidance maneuver the theoretical bounds tend to be valid (see Fig. 5). Additionally the critic network gives a previous estimate about the physical feasibility of planned trajectory besides the learned optimal trajectory planner (actor network). It can have pragmatic meaning in terms of a decision model of control system of autonomous vehicles.

5 CONCLUSIONS

The paper presents a possible motion planner approach, with the application of Deep Deterministic Policy Gradient based reinforcement learning combined with classic control solutions. The results showed that the combination of AI and classic approach can make a good tool for designing effective solutions for autonomous vehicle control, where additional information on the feasibility and stability can be acquired.

The algorithm shows convergence during the learning hence, the trained agent is basically capable to generate effective trajectories. The visual inspection of the situations shows that the overall behavior of the agent fulfills the requirements.

Further research will focus on extending the environment with a variable speed solution and real-world tests.

ACKNOWLEDGEMENTS

The research reported in this paper was supported by the Higher Education Excellence Program of the Ministry of Human Capacities in the frame of Artificial Intelligence research area of Budapest University of Technology and Economics (BME FIKPMI/FM).

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. Supported by The UNKP-18-3 New National Excellence Program of The Ministry of Human Capacities

REFERENCES

Aradi, S., Becsi, T., and Gaspar, P. (2018). Policy gradient based reinforcement learning approach for autonomous highway driving. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 670–675. IEEE.

Bécsi, T., Aradi, S., Fehér, Á., Szalay, J., and Gáspár, P. (2018). Highway environment model for reinforcement learning. *IFAC-PapersOnLine*, 51(22):429–434.

Chen, Y. F., Everett, M., Liu, M., and How, J. P. (2017). Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE.

Fehér, Á., Aradi, S., and Bécsi, T. (2018). Q-learning based reinforcement learning approach for lane keeping. In *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*. IEEE.

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2015). Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074. IEEE.

Hegedüs, F., Bécsi, T., and Aradi, S. (2017a). Dynamically feasible trajectory planning for road vehicles in terms of sensitivity and robustness. *Transportation Research Procedia*, 27:799 – 807. 20th EURO Working Group on Transportation Meeting, EWGT 2017, 4-6 September 2017, Budapest, Hungary.

Hegedüs, F., Bécsi, T., Aradi, S., and Gápár, P. (2017b). Model based trajectory planning for highly automated road vehicles. *IFAC-PapersOnLine*, 50(1):6958 – 6964. 20th IFAC World Congress.

Li, D., Zhao, D., Zhang, Q., and Chen, Y. (2019). Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]. *IEEE Computational Intelligence Magazine*, 14(2):83–98.

Li, X., Sun, Z., He, Z., Zhu, Q., and Liu, D. (2015). A practical trajectory planning framework for autonomous ground vehicles driving in urban environments. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1160–1166.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Liu, L., Hu, J., Wang, Y., and Xie, Z. (2017). Neural network-based high-accuracy motion control of a class of torque-controlled motor servo systems with input saturation. *Strojniski Vestnik-Journal of Mechanical Engineering*, 63(9):519–529.

Minh, V. T. and Pumwa, J. (2014). Feasible path planning for autonomous vehicles. *Mathematical Problems in Engineering*, 2014.

Pacejka, H. B. (2012). Chapter 8 - applications of transient tire models. In Pacejka, H. B., editor, *Tire and Vehicle Dynamics (Third Edition)*, pages 355 – 401. Butterworth-Heinemann, Oxford, 3rd edition edition.

Palmieri, L., Koenig, S., and Arras, K. O. (2016). Rrt-based nonholonomic motion planning using any-angle path biasing. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2775–2781. IEEE.

Paxton, C., Raman, V., Hager, G. D., and Kobilarov, M. (2017). Combining neural networks and tree search for task and motion planning in challenging environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6059–6066. IEEE.

Plessen, M. G. (2019). Automating vehicles by deep reinforcement learning using task separation with hill climbing. In *Future of Information and Communication Conference*, pages 188–210. Springer.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *ICML*.

Singh, S., Majumdar, A., Slotine, J.-J., and Pavone, M. (2017). Robust online motion planning via contraction theory and convex optimization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5883–5890. IEEE.

Tai, L., Paolo, G., and Liu, M. (2017). Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 31–36. IEEE.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., and Mahoney, P. (1970). *Stanley: the robot that won the DARPA Grand Challenge*, volume 23, pages 1–43. Wiley InterScience.

Vorobieva, H., Minoiu-Enache, N., Glaser, S., and Mammar, S. (2013). Geometric continuous-curvature path planning for automatic parallel parking. In *2013 10th IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pages 418–423.

Yim, Y. U. and Oh, S.-Y. (2004). Modeling of vehicle dynamics from real vehicle measurements using a neural network with two-stage hybrid learning for accurate long-term prediction. *IEEE Transactions on Vehicular Technology*, 53(4):1076–1084.