scientific reports

Check for updates

OPEN

RRT-guided experience generation for reinforcement learning in autonomous lane keeping

Tamás Bécsi

Reinforcement Learning has emerged as a significant component of Machine Learning in the domain of highly automated driving, facilitating various tasks ranging from high-level navigation to control tasks such as trajectory tracking and lane keeping. However, the agent's action choice during training is often constrained by a balance between exploitation and exploration, which can impede effective learning, especially in environments with sparse rewards. To address this challenge, researchers have explored combining RL with sampling-based exploration methods such as Rapidly-exploring Random Trees to aid in exploration. This paper investigates the effectiveness of classic exploration strategies in RL algorithms, particularly focusing on their ability to cover the state space and provide a quality experience pool for learning agents. The study centers on the lane-keeping problem of a dynamic vehicle model handled by RL, examining a scenario where reward shaping is omitted, leading to sparse rewards. The paper demonstrates how classic exploration techniques often cover only a small portion of the state space, hindering learning. By leveraging RRT to broaden the experience pool, the agent can learn a better policy, as exemplified by the dynamic vehicle model's lane-following problem.

Motivation

Reinforcement Learning (RL) as part of Machine Learning is gaining more and more attention in the field of highly automated driving, having several utilizations, from high-level navigation through local motion planning to control tasks, such as trajectory tracking or lane keeping[1,2]. Nowadays, these solutions generally use artificial neural networks as the agents' function approximators, forming the so-called Deep Reinforcement Learning (DRL) problem. The framework for such systems utilizes the Markov Decision Process (MDP)[3], placing an acting agent in an interacting environment providing state information to the agent, based on which it chooses an action, leading to the next state. During this process, the agent collects rewards, and its final goal is to maximize its gains. To do so, the agent performs various episodes on the environment, acting based on the policy learned from previous experiences.

The agent's action choice during the training is a trial-and-error process, where it maintains a balance between utilizing its best practice, exploitation, and exploration, which is searching for new possible good actions. Keeping this balance and the fact that the agent chooses only one action at a time prevents it from efficiently finding strategies, especially for sparse rewards. To handle this issue, multiple research utilizes search methods or heuristics to aid exploration, like the Monte Carlo Tree Search[4] used by the AlphaGo[5] architecture. Among the sampling-based methods, Rapidly-exploring Random Trees (RRT)[6] is another popular choice for aiding exploration.

This paper aims to investigate multiple questions regarding the lane-keeping problem of a dynamic vehicle model handled by Reinforcement Learning: First, it shows that in case reward shaping is omitted, and the reward is sparse, the fast learning agents struggle to adapt to an optimal policy. The second research question is how effective the classic exploration is, whether it can cover the state space and provide a quality experience pool for the agent to learn. The answer to the second question leads to the paper's motivation. As shown later, the classic one-step exploration covers only a narrow part of the state space, resulting in inefficient training. The final question is whether all nodes of an unguided RRT could serve as a better experience pool than the one resulting from the ϵ-greedy exploration.

Related work

Many approaches in the literature combine sampling-based heuristics, especially RRT, with different reinforcement learning agents to provide a better or more efficient motion planner. Reinforcement learning

Department of Control for Transportation and Vehicle Systems, Faculty of Transportation Engineering and Vehicle Engineering, Budapest University of Technology and Economics, Budapest 1111, Hungary. email: becsi.tamas@kjk.bme.hu

itself can be an alternative to these methods. Xiaofei et al. propose a DDQN-based path planning algorithm for the global static path planning problem of an amphibious unmanned surface vehicle (USV)[7] using action mask operation to improve the action selection by combining some known experiences and also path smoothing to generate the final path. Comparing their results to the ones from RRT shows the efficiency of their approach. Zhang et al. use paradigms from classic heuristics to train the path planning of a Double DQN (DDQN) guided robot in a grid environment by applying RRT-based intuition for the initialization of the agent and A*-based principles for the generation of the reward function[8] achieving a more stable training process and a more efficient behavior. A possible combination of RL and sampling-based motion planning methods is the PRM-RL concept[9], which adopts a hierarchical approach to long-range navigation tasks. This method leverages a probabilistic route mapping (PRM) algorithm alongside a trained RL agent to assess the feasibility of connections between points in configuration space by substituting a collision-free straight line for the actual path. Such integration enhances the efficiency of both RL agents and sampling-based planners in addressing long-range navigation challenges. The fusion of Deep Reinforcement Learning and Rapidly-exploring Random Trees has found application in various contexts. Hollenstein et al. introduce the Planning for Policy Search (PPS) algorithm[10], which serves as a learning pattern generator for DRL algorithms, employing an Affine Quadratic Regulator-based cost function to determine the nearest tree node. A local planner based on Model Predictive Control (MPC) is then employed to generate the next node for connection to the existing tree. This approach demonstrates efficacy in the OpenAI control environment, characterized by continuous action spaces. Deep reinforcement learning can also be used to improve the effectiveness of the classic path planning algorithms. Li et al.[11] proposes an extension to the RRT algorithm, where the sampling algorithm of the RRT is guided by a DQN agent, which, in the end, improves the performance of the algorithm, improving its convergence speed and path quality. Chiang et al.[12] train an RL algorithm to navigate obstacles in a motion planning task and subsequently utilize the learned agents to collect additional data for training a reachability estimator. This estimator accurately measures the time required to reach a state and, when integrated into the RRT, acts as a cost function, guiding tree growth towards promising regions. This RL-RRT integration results in shorter paths with reduced planning efforts. Meng et al. propose the application of deep learning methodologies to augment sampling-based planning techniques, leading to the development of a novel risk-bounded near-optimal path planning algorithm termed neural risk-aware RRT (NR-RRT)[13]. The approach involves maintaining a deterministic risk contours map to perceive probabilistic nonconvex obstacles, with a neural network sampler introduced to forecast the next most promising safe state. Finally Sadhu et al. present the Simultaneous Learning and Planning Algorithm (SLPA)[14] for robot navigation, combining Backward Q-learning (BQL) and RRT*. This hybrid approach outperforms RRT* in optimizing path lengths by balancing the exploration-exploitation trade-off through RRT*. While SLPA demonstrates superior performance, scalability concerns may arise due to its utilization of BQL in tabular form, lacking the generalization capabilities offered by function approximations.

Contribution of the paper

According to the literature review above, combining heuristic path or motion planning methods with reinforcement learning may serve multiple purposes. One is to train an agent that can increase the effectiveness of determining the new sampling points of the RRT algorithm, which provides a faster and more efficient planner. The other way is to use a guided local RRT search during the exploration phase of the RL algorithm, which provides more rewarding or successful episodes that result in a better experience pool and, in the end, increase the efficiency of the training of an RL agent. Also, principles from RRT can even be used to construct the training logic or the reward.

In summary, either the RRT helps the exploration of the DRL, or the DRL agent controls the sampling of the RRT. Contrary to these approaches, this paper seeks the answer to the efficiency of the exploration of RL algorithms. To do so, it utilizes RRT as a standalone experience generator for the agent to provide a more covered state space, completely omitting the DRL-s own exploration process. In this regard, the central insight of the paper is illustrated later by Fig. 4, showing how the classic exploration-exploitation technique covers a small portion of the actual state space. As a result of the broader and more homogenous experience pool, the agent can learn a better policy. To illustrate this phenomenon, this paper uses a dynamic vehicle model's lane following problem, choosing DQN and standard RRT as the agent and heuristics.

Markov decision process formulation

This section introduces the parts of the MDP used for this reinforcement learning problem: the environment consisting of the vehicle dynamics model and the state representation, the reward, and the action space. The track is provided by a simple track generator[15]. The curvature extremums of the track are not adjusted against feasibility for the given vehicle dynamics circumstances. Hence, there could be unsolvable episodes, though they do not affect the training outcome.

Vehicle dynamics

Many RL-based motion planning research uses simple kinematic models for training. On the one hand, this is a reasonable choice, as this kind of Machine learning needs a vast amount of simulation, where the computational requirements of the models are a crucial factor in training time. On the other hand, kinematic models neglect too many aspects of dynamic maneuvers. Therefore, one has to find a trade-off in model complexity between computation time and accuracy[16]. Achieving this balance is essential for ensuring the trained model can generalize well beyond the training scenarios and handle real-world complexities effectively.

A nonlinear single-track vehicle model containing a dynamic wheel model is developed to accurately calculate the vehicle's motion. Single-track models are extensively utilized in vehicle simulations and model-based computations because they provide adequate precision across a broad range of driving scenarios while

maintaining moderate complexity [17].The multi-body model (see Fig. 1) consists of three elements: the vehicle chassis and two virtual wheels, which rigidly connect the front and rear axles. The main parameters are defined according to the vehicle to be modeled, which are the mass m and moment of inertia θ of the chassis, the horizontal distances between the vehicle's CoG and the front and rear wheel centers l_f and l_r, the CoG 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 wheel models' parameters also have a strong influence, from which the most important ones are the coefficient of friction $\mu_{[f/r]}$. The Magic Formula defines the transmittable amount of force between road and tires[19]. More details on the model can be found in[18].

The model used here accurately simulates driving situations, even under high accelerations, near the boundaries of adhesion and stability. It achieves this with moderate computational demands compared to comprehensive four-wheel 3D models. However, it does not account for vertical dynamic impacts like uneven roads or load shifts due to slopes. Additionally, it cannot simulate scenarios where the friction conditions of all four wheels are crucial, such as cases involving μ-split, where different road surfaces affect the left and right sides of the vehicle. However, in this case, these situations can be neglected.

State representation

According to Aradi[2], the literature uses three types of state representation for the lane-keeping task: sensor, intermediate, or ground truth level information. On the sensor level, Lee et al.[20] uses lidar simulation, while others[21–23] use front-facing camera image. Intermediate-level representations usually use spatial maps[24] or grid-based representations[8,25] . Some approaches also incorporate lookahead information to anticipate the future trajectory evolution[15]. Ground-truth level solutions used in this research use directly extracted parameters containing vehicle state information and topological data in a vectorized form. Kinematic models can use the vehicle's position, speed, and angle on the track. In contrast, knowing other parameters for the agent to act properly for dynamic models and comfort aspects is important. Therefore, the current study's state vector S_t passed at time t to the agent has the form given in (1):

$$S_t = \{d, v_x, v_y, \psi, \dot{\psi}, \lambda, \beta, L, A_{t-1}\} \tag{1}$$

$$L = \{\theta_1, \theta_2, \ldots, \theta_{10}\} \tag{2}$$

The first three parameters are the vehicle's lateral position from the lane center d and its speed components v_x, v_y in the ego coordinate system. ψ is the yaw angle relative to the tangent of the track, while $\dot{\psi}$ is the vehicle's yaw rate, i.e., the angular velocity along the Z axis.

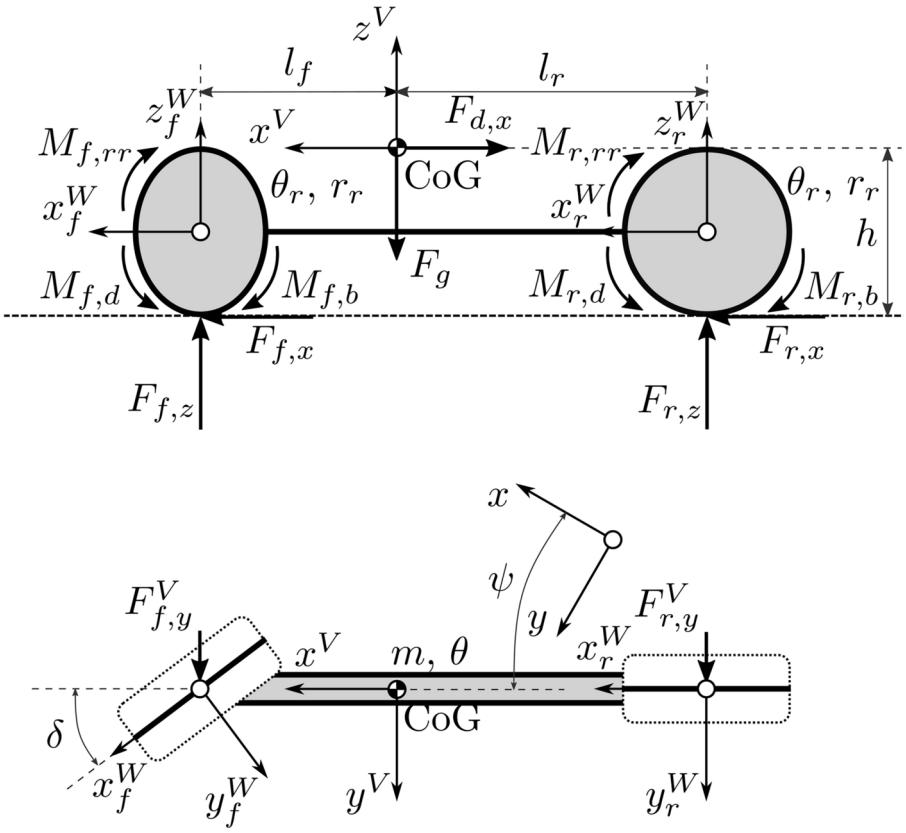

Fig. 1. Nonlinear single track vehicle model[18].

Longitudinal slip, denoted by the symbol λ, represents the difference between a vehicle's actual longitudinal velocity and the velocity of its wheels. It measures how much the wheels slip or slide relative to the road surface. Longitudinal slip is a crucial parameter in understanding tire-road interaction and is used in vehicle dynamics and control systems to optimize traction, stability, and braking performance. In automated driving, accurate estimation of longitudinal slip is essential for designing control algorithms that ensure safe and effective vehicle operation. Mathematically, it can be expressed as the ratio of the difference between the vehicle's longitudinal velocity (v_x) and the velocity of the wheels (v_{wheel}) to the velocity of the wheels:

$$\lambda = \frac{v_{\text{wheel}} - v_x}{v_{\text{wheel}}}. \tag{3}$$

Lateral slip β refers to the difference between the direction in which a vehicle is pointing (its heading) and the direction it is moving. It is a measure of the sideways slip of the vehicle. It can be expressed as the arctangent of the ratio of the lateral velocity v_y to the longitudinal velocity v_x of the vehicle:

$$\beta = \arctan\left(\frac{v_y}{v_x}\right). \tag{4}$$

The agent needs its current position and information on the upcoming curvatures to plan on the track. This could be done by passing a parametrized curve of the path ahead. The actual state representation solves this issue with a set of ten angles L shown in (2), where each θ_i value represents the angle between the vehicle's heading and the tangent of the lane centerline at a specified distance. Figure 2 illustrates the elements of the state vector. All parameters were scaled to fall within the interval $[-1;1]$ to aid the convergence of the underlying neural network.

Reward strategy

As its name suggests, the primary objective of lane-keeping is to control the vehicle or mobile robot between the boundaries of the previously defined track, leading to the straightforward reward design focusing on this aspect only. On the contrary, simply punishing lane departure or rewarding the mobile robot by keeping track is quite a sparse reward, as it holds no information for the agent about how well it performs. Naturally, this could lead to more extended and more challenging training. This is why most of the papers in the literature utilize reward shaping, considering mainly two parameters in the reward: the lateral distance from the centerline of the track d and the relative yaw angle to the tangent of this centerline ψ, punishing their deviation from zero. This can either be linear[26] (5), or nonlinear to the yaw[27] (6) or to the lateral distance[20] (7).

$$R_{lk} = -(c * d + \psi) \tag{5}$$

$$R_{lk} = v(\cos\psi - d) \tag{6}$$

$$R_{lk} = \tanh\frac{|c-d|}{c} \tag{7}$$

In the equations above, v stands for the longitudinal speed of the vehicle, while c is some constant in each research. Passenger comfort or cargo safety indicates a secondary goal: to make the resulting trajectory of the vehicle as smooth as possible. In reward design terms, this can be achieved by penalizing frequent changes in the steering angle[28], lateral acceleration[29,30], or jerk[31,32].

Though incorporating yaw and lateral deviation speeds up a learning process, they also guide the agent based on human knowledge to learn a policy that could deviate from the optimal. According to Russell and Norvig[33]: "As a general rule, it is better to design performance metrics according to what one actually wants to be achieved in the environment, rather than according to how one thinks the agent should behave." Inspecting this phenomenon in automated driving, Knox et al.[34] point out that this problem exists in this area, calling it unsafe reward shaping. Indeed, minimizing yaw angle and lateral distance while not violating lane boundaries could lead to the rigorous keeping of the centerline, which prevents the vehicle from driving on more ideal curves and being able to take sharper turns by cutting them.

Along the lines of the arguments mentioned above, the reward system focusing on the original task is used in the article, rewarding only the keeping of the lane. Furthermore, in an actual application, a high slip value is also detrimental regarding passenger comfort and tire wear. Therefore, the final reward (10) comes from two objectives, the lane-keeping reward (8) and the slip-based one (9), which is only actuated above a certain threshold.

$$R_{lk} = \begin{cases} 1 & \text{if vehicle stays in track} \\ -1 & \text{if vehicle leaves track} \end{cases} \tag{8}$$

$$R_s = \begin{cases} 0 & \text{if } \ |\lambda| < 0.3 \\ -\sqrt{\lambda} & \text{otherwise} \end{cases} \tag{9}$$

$$R = R_{lk} + R_s \tag{10}$$

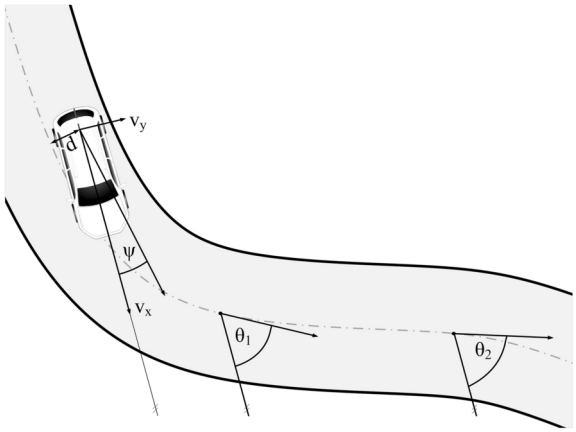

Fig. 2. State representation and lookahead angles.

Action-space

For this agent, where the action space is discrete, this research utilizes a set of steering angles. The values shown in (11) are in radian. Earlier research usually used 3 or 5 discrete values for such decisions, which is sufficient when utilizing a kinematic model or when the model neglects passenger safety. However, in order to minimize slip, this choice is rational.

$$A = \{-0.5, -0.34, -0.17, 0, 0.17, 0.34, 0.5\}[rad] \tag{11}$$

Algorithms

Deep reinforcement learning

Contrary to supervised learning, where the training data is collected and provided to the agent, Reinforcement learning is a different paradigm in artificial intelligence, where agents learn optimal behaviors through interactions with their environment. The emergence of deep reinforcement learning signifies a crucial advancement, leveraging deep neural networks to handle complex and high-dimensional state spaces. At its core, reinforcement learning involves an agent making sequential decisions to maximize cumulative rewards within a given environment. Doing so, it utilizes the Markov Decision Process as a modeling paradigm where the problem is described by the (*S*, *A*, *R*, *P*) tuple, where *S* represents the set of states, *A* is the set of actions, *P* denotes the state transition probability function, and *R* is the reward function. The MDP dynamics adhere to the Markov property, ensuring that the future state depends only on the current state and action, not on the sequence of events leading to it. Solving an MDP involves finding a policy, a strategy that dictates the agent's actions in each state, to maximize the expected cumulative reward over time.

Deep Q-network algorithm

The Deep Q-Network (DQN) is a fundamental algorithm in deep reinforcement learning. It combines Q-learning with deep neural networks to handle high-dimensional state spaces. The goal of DQN is to learn a Q-function, denoted as $Q(s, a; \theta)$, where s is the state, a is the action, and θ represents the parameters of a deep neural network. The Q-value is updated using the temporal difference (TD) error:

$$\delta = (r + \gamma \max_{a'} Q(s', a'; \theta^{-})) - Q(s, a; \theta), \tag{12}$$

where *r* is the immediate reward after taking action *a* in state *s*, s' is the next state, γ is the discount factor, $(Q(s, a; \theta^{-})$ represents the target Q-value computed using a target network with parameters θ^{-}. The loss function is defined to minimize the TD error:

$$\mathscr{L}(\theta) = \mathbb{E}\left[(\delta)^2\right] \tag{13}$$

DQN uses experience replay to break the temporal correlation between consecutive samples. The agent stores experience (s, a, r, s') in a replay buffer and samples a minibatch during each update. DQN uses a target network with parameters θ^{-} to stabilize training to compute the target Q-values. The target network is periodically updated with the main Q-network's parameters. During action selection, DQN uses an ϵ-greedy strategy, where with probability ϵ, a random action is chosen, and with probability $1 - \epsilon$, the action with the maximum Q-value is selected.

Rapidly exploring random trees

The Rapidly Exploring Random Trees algorithm is a probabilistic, sampling-based approach used for motion planning in robotics, introduced by LaValle in 1998[6]. RRT rapidly explores the configuration space

by incrementally growing a tree of feasible paths from an initial state towards a goal state. It has been widely applied in robotics for motion planning tasks and is known for its simplicity, efficiency, and adaptability to high-dimensional state spaces. It is biased towards unexplored areas, allowing it to explore diverse regions of the configuration space quickly. It also generates random configurations in the state space, attempting to connect them to the existing tree. While not guaranteed to find the optimal path, RRT is often effective in providing feasible solutions quickly, making it suitable for dynamic and complex environments.

The combined method

As mentioned before, one of the key problems in RL is the exploration-exploitation dilemma, combined with the fact that any pure RL algorithm can only use its present knowledge on its preferred action in a given state. Hence, the exploration is probabilistic and only applied on a one-step level. As a result, the actual action space is only narrowly covered by the experiences, which may withhold the agent from learning a robust strategy. The proposed method therefore, replaces the actual experience-gathering technique based on playing complete episodes linearly and applying some exploration technique with an RRT-based approach. Algorithm 1 describes this experience-gathering process in detail. The algorithm ensures the generation of a diverse set of experiences by running multiple episodes, each consisting of a random track and vehicle initial state.

Require: $POOL\_SIZE$ ▷ Expected size of the pool
Require: $EPISODE\_SIZE$ ▷ Experience count limit of one episode
$Pool \leftarrow \emptyset$ ▷ Initialize experience pool
while $|Pool| < POOL\_SIZE$ **do** ▷ Perform episodes until pool is full
  track ← GenerateRandomTrack() ▷ Generate a random track
  initialState ← GenerateRandomInitialState() ▷ Place the vehicle in a random position
  $RRT \leftarrow \text{GenerateRRT}(\text{initialState}, EPISODE\_SIZE)$ ▷ Generate an RRT
  for each node n in RRT **do**
    reward ← CalculateReward(n)
    for each (generated) $action$ of n **do**
      $nextnode \leftarrow n.nodes(action)$
      $Pool \leftarrow Pool \cup \{(n.\text{state}, action, \text{reward}, nextnode.\text{state})\}$
    end for
  end for
end while

Algorithm 1. Experience Pool Generation

The detailed steps of the algorithm are as follows: The algorithm starts by initializing an empty experience pool *Pool*. The main loop continues to execute until the pool contains $POOL\_SIZE$ experiences. Within each iteration of the loop, referred to as an episode, the following steps are performed:

1. Track and Initial State Generation: A random track is generated, and the vehicle is placed at a random state on this track.
2. RRT Generation: A Rapidly-exploring Random Tree (RRT) is constructed starting from the initial state. The RRT generation is limited by $EPISODE\_SIZE$, which dictates the maximum number of nodes in the tree.
3. Node Processing: For each node n in the RRT, a reward is calculated based on the node's state.
4. Action Exploration: Each action that is generated to the node n is considered. The resulting state, *nextnode*, after taking the action is identified.
5. Experience Storage: The experience, encapsulated as a tuple $(n.\text{state}, action, \text{reward}, nextnode.\text{state})$, is added to the experience pool *Pool*.During the sample collection process, the algorithm records the training samples and the full scenario snapshot of the given step. Subsequently, the Rapidly Exploring Random Trees algorithm identifies the closest point from the accumulated positions to a randomly generated point. It then takes a step from this identified point, employing a randomly selected steering angle, and records the resulting configuration as a training sample. This iterative procedure generates a set of highly explored samples within the state space. The exploration capability of the combined method on a closed track is illustrated in Fig. 3.

In particular, this method differs from traditional approaches in that it refrains from using neural network predictions for sample collection. It clearly separates the processes of sample collection and network training. Figure 4 illustrates the difference between the two exploration strategies, where, for the sake of better visualization, a two-dimensional subspace of the observation space is chosen, containing the normalized lateral position d and yaw angle ψ. The figure presents the collected samples based on the two strategies and shows that the RRT covers a larger, more uniformly distributed area.

Results

To compare the RRT-based and the ϵ-greedy sample collection, every parameter and factor was chosen as the same for all two methods, including the set of closed tracks, the number of collected samples, and the number of training epochs.

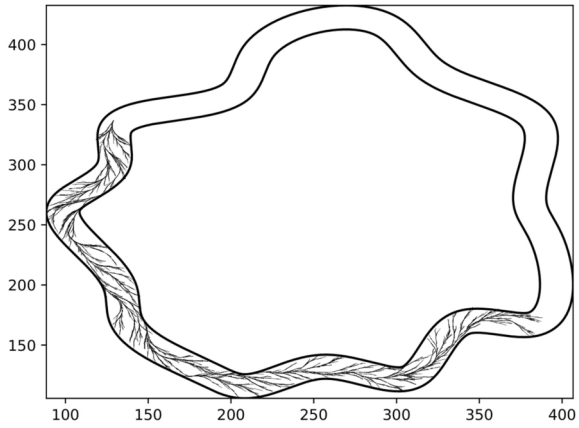

Fig. 3. Exploration on track.

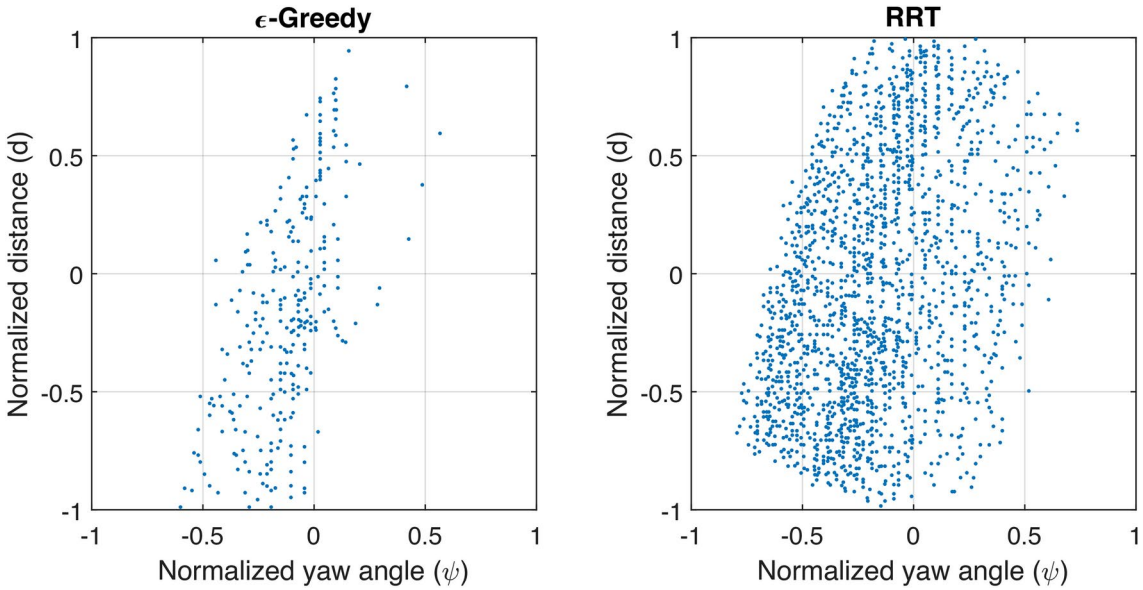

Fig. 4. State space coverage of the two different exploration methods.

Statistical comparisons were made through the analysis of performance indicators, with particular attention to two primary metrics. The first of these metrics is the success rate, which indicates the ratio of successful episodes to the total number of episodes. In parallel, the second indicator is the average step, representing the average number of steps taken in all episodes. In addition to measuring success rate, the average step metric reveals an agent's proficiency by accounting for significant variation in the number of steps between unsuccessful episodes. As previously mentioned, both agents were trained through 10, 000 epochs and were tested on the same set of 1000 tracks. It is important to note that two random seeds were used to generate the tracks. The first random seed generated the training episodes' tracks in the same order for both agents, ensuring a fair performance comparison. The second random seed was used to create the tracks for evaluation, allowing the assessment of the generalization capability of each strategy. Every episode was limited to 250 steps. The agents' success rates, average steps, and rewards are summarized in Table 1. The tracks involved were randomly generated and not validated against feasibility, which explains the classical agent's low success rate and highlights the importance of

	Avg. steps	Success rate	Avg. reward	Avg. reward/step
ϵ-Greedy	139.17	12.8%	76.99	0.55
RRT-DRL	238.60	90.6%	181.60	0.76

Table 1. Statistical results.

the composition of the experience pool for training. The results are generated with the two trained agents, where only exploitation occurs. It is essential to mention that the paper's main finding is depicted in Fig. 4. However, for the given setting, the results provided by Table 1 show that the classic exploration falls on the challenging tracks, reaching the final 250th step once in every eight trials, while the agent trained on the broader pool only fails once out of ten tries. The average steps taken also correspond to this performance. One last exciting phenomenon can be calculated from the reward and steps gathered, showing that the RRT-DRL agent's slip-based penalty was much lower on average.

Although these metrics provide insights into the agent's success, they fall short of capturing the quality of motion. In the context of a dynamic vehicle model, slip is a critical parameter. High slip values can cause an increase in tire temperature, leading to energy loss derived from the vehicle's kinetic energy. This results in a significant reduction in vehicle velocity. However, an excessive slip can accelerate tire wear and compromise passenger comfort. Therefore, finding an optimal slip value is important, acknowledging that it cannot be reduced to zero due to the inherent ground-tire interaction. The preferred method for slip measurement involves considering the vehicle's absolute velocity and the slip itself.

Figure 5 shows example paths traveled by the two agents on the same track. The agents only perform lateral control through steering while constant torque is applied to the driven wheels. However, this torque cannot always be transferred, as a lateral force acts on the tires during cornering. This lateral force can affect the longitudinal slip by changing the traction available for acceleration or braking. As the lateral force increases, the available traction for braking or accelerating may decrease, leading to changes in longitudinal slip. As the classic agent's maneuvers are not ideal, it uses more aggressive turns, resulting in higher lateral and lower longitudinal slip values. On one hand, this affects the vehicle speed; on the other hand, it results in higher wear on the tires. The more ideal path and the higher vehicle speed can be observed in the figure through the longer distance traveled by the RRT-based agent.

Based on these findings, it can be inferred that the vehicle will achieve faster completion of a track segment while experiencing reduced tire wear and diminished loss. Considering the dynamics of the ground-tire interaction, it can further be deduced that operating at lower slip values preserves greater traction reserve within the tires. Consequently, the vehicle exhibits enhanced maneuverability in an unforeseen emergency, ensuring safer travel despite its elevated velocity.

A complex combination of turns presents the difference between the two agents in Fig. 6. It can be seen that the RRT-based algorithm goes much closer to the ideal curve when executing the turns.

Conclusions

It can be observed from the results that the combination of Rapidly Exploring Random Trees with Deep Reinforcement Learning significantly improves the performance of the agent in terms of both success rate and average step count. By employing RRT for sample collection, the agent explores a broader and more uniform area of the state space, resulting in a more robust policy. This approach demonstrates improved motion quality, as evidenced by reduced longitudinal slip and smoother path trajectories.

Furthermore, the RRT-based agent exhibits superior maneuverability, as illustrated by its ability to maintain higher speeds while traversing complex turns. This enhanced maneuverability is attributed to the agent's ability to generate more optimal trajectories, which minimize tire wear and energy loss. Consequently, the RRT-DRL agent offers improved performance and enhanced safety and efficiency in dynamic driving scenarios.

In conclusion, integrating RRT with DRL presents a promising autonomous vehicle motion planning approach, offering significant advancements in exploration, learning, and decision-making. Further research and experimentation in this direction could lead to even more significant improvements in autonomous driving technology, paving the way for safer, more efficient, and more reliable autonomous vehicles.

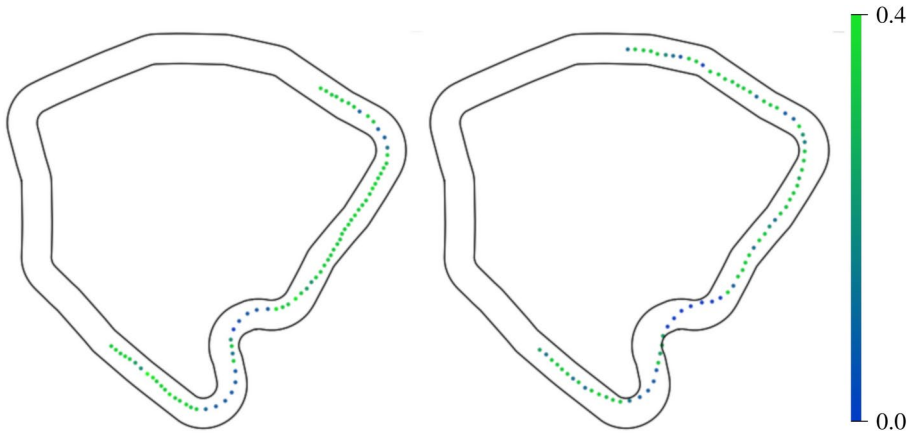

Fig. 5. Longitudinal slip and paths of the two agents.

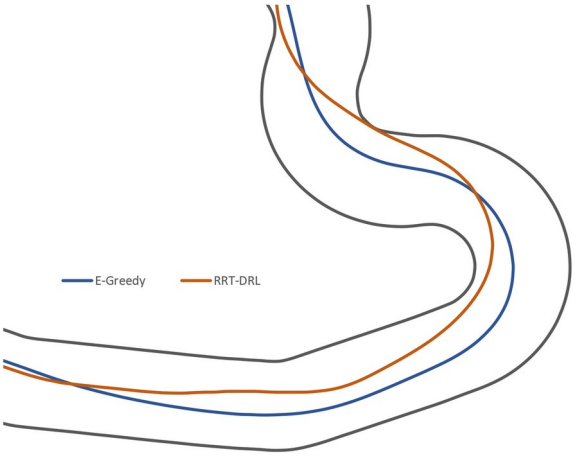

Fig. 6. Comparison of path traveled in a complex series of turns.

Data availability

No training datasets were used and/or analyzed during the current study; only model-generated ones were used. All parameters are shared in the article. Software is available from the corresponding author upon reasonable request.

Received: 14 May 2024; Accepted: 23 September 2024
Published online: 14 October 2024

References

1. Kiran, B. R. *et al.* Deep reinforcement learning for autonomous driving: A survey. *IEEE Trans. Intell. Transport. Syst.* **23**, 4909–4926. https://doi.org/10.1109/TITS.2021.3054625 (2022).
2. Aradi, S. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Trans. Intell. Transport. Syst.* **23**, 740–759. https://doi.org/10.1109/TITS.2020.3024655 (2022).
3. Sutton, R. S., Barto, A. G. et al. *Introduction to Reinforcement Learning*, vol. 135 (MIT Press, 1998).
4. Kocsis, L. & Szepesvári, C. Bandit based monte-carlo planning. In *European Conference on Machine Learning*, 282–293 (Springer, 2006).
5. Silver, D. *et al.* Mastering the game of go without human knowledge. *Nature* **550**, 354–359 (2017).
6. LaValle, S. Rapidly-exploring random trees: A new tool for path planning. Research Report 9811 (1998).
7. Xiaofei, Y. *et al.* Global path planning algorithm based on double DQN for multi-tasks amphibious unmanned surface vehicle. *Ocean Eng.* **266**, 112809. https://doi.org/10.1016/j.oceaneng.2022.112809 (2022).
8. Zhang, F., Gu, C. & Yang, F. An improved algorithm of robot path planning in complex environment based on double dqn. In *Advances in Guidance, Navigation and Control* (eds. Yan, L., Duan, H. & Yu, X.) 303–313. https://doi.org/10.1007/978-981-15-8155-7_25 (Springer Singapore, 2022) .
9. Faust, A. et al. Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 5113–5120. https://doi.org/10.1109/ICRA.2018.8461096 (2018).
10. Hollenstein, J. J., Renaudo, E., Saveriano, M. & Piater, J. Improving the exploration of deep reinforcement learning in continuous domains using planning for policy search. **2010**, 12974 (2020).
11. Li, Z., Huang, J., Fei, Y. & Shi, R. A novel exploration mechanism of RRT guided by deep q-network. *Unmanned Syst.* **12**, 447–456. https://doi.org/10.1142/S2301385024420068 (2024).
12. Chiang, H.-T.L., Hsu, J., Fiser, M., Tapia, L. & Faust, A. Rl-rrt: Kinodynamic motion planning via learning reachability estimators from rl policies. *IEEE Robot. Autom. Lett.* **4**, 4298–4305. https://doi.org/10.1109/LRA.2019.2931199 (2019).
13. Meng, F., Chen, L., Ma, H., Wang, J. & Meng, M.Q.-H. Nr-rrt: Neural risk-aware near-optimal path planning in uncertain nonconvex environments. *IEEE Trans. Autom. Sci. Eng.* **21**, 135–146. https://doi.org/10.1109/TASE.2022.3215562 (2024).
14. Sadhu, A. K., Shukla, S., Sortee, S., Ludhiyani, M. & Dasgupta, R. Simultaneous learning and planning using rapidly exploring random tree* and reinforcement learning. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, 71–80. https://doi.org/10.1109/ICUAS51884.2021.9476861 (2021).
15. Kvári, B., Hegedüs, F. & Bécsi, T. Design of a reinforcement learning-based lane keeping planning agent for automated vehicles. *Appl. Sci.* **10**. https://doi.org/10.3390/app10207171 (2020).
16. Hegedüs, F., Bécsi, T., Aradi, S. & Gáspár, P. Model based trajectory planning for highly automated road vehicles. IFAC-PapersOnLine 20th IFAC World Congress. **50**, 6958–6964. https://doi.org/10.1016/j.ifacol.2017.08.1336 (2017).
17. Schramm, D., Hiller, M. & Bardini, R. *Vehicle dynamics: Modeling and simulation* (Springer, Berlin, 2014).
18. Hegedüs, F., Bécsi, T., Aradi, S. & Gáspár, P. Motion planning for highly automated road vehicles with a hybrid approach using nonlinear optimization and artificial neural networks. *Strojniški vestnik J. Mech. Eng.* **65**, 148–160. https://doi.org/10.5545/sv-jme.2018.5802 (2019).
19. Pacejka, H. B. Chapter 8—Applications of transient tire models. In *Tire and Vehicle Dynamics*, 3rd edn. 355–401. https://doi.org/10.1016/B978-0-08-097016-5.00008-5 (Butterworth-Heinemann, 2012).
20. Lee, J., Kim, T. & Kim, H. J. Autonomous lane keeping based on approximate Q-learning. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 402–405. https://doi.org/10.1109/URAI.2017.7992762 (IEEE, 2017).
21. Li, G., Ji, Z., Li, S., Luo, X. & Qu, X. Driver behavioral cloning for route following in autonomous vehicles using task knowledge distillation. *IEEE Trans. Intell. Veh.* **8**, 1025–1033. https://doi.org/10.1109/TIV.2022.3198678 (2023).
22. Xu, N., Tan, B. & Kong, B. Autonomous driving in reality with reinforcement learning and image translation. **1801**, 05299 (2019).

23. Li, D., Zhao, D., Zhang, Q. & Chen, Y. Reinforcement learning and deep learning based lateral control for autonomous driving. **1810**, 12778 (2018).
24. Folkers, A., Rick, M. & Büskens, C. Controlling an autonomous vehicle with deep reinforcement learning. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2025–2031. https://doi.org/10.1109/IVS.2019.8814124 (2019).
25. Ghadi, M. A grid-based framework for managing autonomous vehicles' movement at intersections. *Period. Polytech. Transport. Eng.* **52**, 235–245. https://doi.org/10.3311/PPtr.24397 (2024).
26. Toromanoff, M., Wirbel, E. & Moutarde, F. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7151–7160. https://doi.org/10.1109/CVPR42600.2020.00718 (2020).
27. Jaritz, M., de Charette, R., Toromanoff, M., Perot, E. & Nashashibi, F. End-to-end race driving with deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2070–2075. https://doi.org/10.1109/ICRA.2018.8460934 (2018).
28. Tang, Y. Towards learning multi-agent negotiations via self-play. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2427–2435. https://doi.org/10.1109/ICCVW.2019.00297 (2019).
29. Wang, P., Chan, C. Y. & De La Fortelle, A. A reinforcement learning based approach for automated lane change maneuvers. In *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2018-June, 1379–1384. https://doi.org/10.1109/IVS.2018.8500556 (Institute of Electrical and Electronics Engineers Inc., 2018).
30. Ronecker, M. P. & Zhu, Y. Deep Q-network based decision making for autonomous driving. In *2019 3rd International Conference on Robotics and Automation Sciences (ICRAS)*, 154–160. https://doi.org/10.1109/ICRAS.2019.8808950 (IEEE, 2019).
31. Zhu, M., Wang, Y., Hu, J., Wang, X. & Ke, R. Safe, Efficient, and Comfortable Velocity Control based on Reinforcement Learning for Autonomous Driving (2019).
32. Saxena, D. M., Bae, S., Nakhaei, A., Fujimura, K. & Likhachev, M. Driving in dense traffic with model-free reinforcement. *Learning* **1909**, 06710 (2019).
33. Russell, S. & Norvig, P. *Artificial Intelligence: A Modern Approach*, 4th edn (Pearson, 2020).
34. Knox, W. B., Allievi, A., Banzhaf, H., Schmitt, F. & Stone, P. Reward (MIS) design for autonomous driving. *Artif. Intell.* **316**, 103829. https://doi.org/10.1016/j.artint.2022.103829 (2023).

Acknowledgements

This research was supported by the European Union within the framework of the National Laboratory for Autonomous Systems. (RRF-2.3.1-21-2022-00002). The research reported in this paper is part of project no. BME-NVA-02, implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021 funding scheme. T.B. was supported by BO/00233/21/6, János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

Author contributions

All roles related to the manuscript were done by Tamás Bécsi: Conceptualization and problem formulation, Software and training, Result analysis, and visualization; T.B. also wrote the whole article.

Declarations

Competing interests

The author declares no competing interests.

Additional information

Correspondence and requests for materials should be addressed to T.B.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.