Article

Constrained Reinforcement Learning for Vehicle Motion Planning with Topological Reachability Analysis

Shangding Gu [1,*], Guang Chen [1,2], Lijun Zhang [2], Jing Hou [2], Yingbai Hu [1] and Alois Knoll [1]

1 Department of Informatics, Technical University of Munich, 80333 Munich, Germany
2 School of Automotive Studies, Tongji University, Shanghai 201804, China
* Correspondence: shangding.gu@tum.de

Abstract: Rule-based traditional motion planning methods usually perform well with prior knowledge of the macro-scale environments but encounter challenges in unknown and uncertain environments. Deep reinforcement learning (DRL) is a solution that can effectively deal with micro-scale unknown and uncertain environments. Nevertheless, DRL is unstable and lacks interpretability. Therefore, it raises a new challenge: how to combine the effectiveness and overcome the drawbacks of the two methods while guaranteeing stability in uncertain environments. In this study, a multi-constraint and multi-scale motion planning method is proposed for automated driving with the use of constrained reinforcement learning (RL), named RLTT, and comprising RL, a topological reachability analysis used for vehicle path space (TPS), and a trajectory lane model (TLM). First, a dynamic model of vehicles is formulated; then, TLM is developed on the basis of the dynamic model, thus constraining RL action and state space. Second, macro-scale path planning is achieved through TPS, and in the micro-scale range, discrete routing points are achieved via RLTT. Third, the proposed motion planning method is designed by combining sophisticated rules, and a theoretical analysis is provided to guarantee the efficiency of our method. Finally, related experiments are conducted to evaluate the effectiveness of the proposed method; our method can reduce 19.9% of the distance cost in the experiments as compared to the traditional method. Experimental results indicate that the proposed method can help mitigate the gap between data-driven and traditional methods, provide better performance for automated driving, and facilitate the use of RL methods in more fields.

Keywords: motion planning; automated driving; reinforcement learning; reachability analysis

check for updates

Citation: Gu, S.; Chen, G.; Zhang, L.; Hou, J.; Hu, Y.; Knoll, A. Constrained Reinforcement Learning for Vehicle Motion Planning with Topological Reachability Analysis. *Robotics* **2022**, *11*, 81. https://doi.org/10.3390/robotics11040081

Academic Editor: Zhangguo Yu

Received: 22 June 2022
Accepted: 8 August 2022
Published: 16 August 2022

1. Introduction

The industrial demand for automated driving technology is increasing, and this technology has made remarkable progress in providing promising transportation to our lives [1,2]. However, vehicle motion planning for automated driving needs further development which considers multiple constraints in sparse information environments; in these environments, macro information is known and micro information is unknown and uncertain, especially for safe, effective, and precise motion planning. In automated driving, a real traffic environment is full of uncertainty due to information incompleteness, traffic disturbances, and other factors such as the incomplete information provided by vehicle sensors and the disturbances of obstacles, which lead to complex traffic environments that may be difficult to predict in real time [3].

Various of traditional motion planning methods (non-data-driven methods) use heuristic algorithms [4], sampling-based methods [5], and rule-based methods [6,7] for automated driving. Although these methods perform well via solid mathematical models in terms of robustness, interpretability, and stability, they still need considerable human knowledge in advance; therefore, modeling the complex and uncertain environments using these models is difficult, and these models might not perform well under unknown and uncertain environments. However, reinforcement learning (RL) (data-driven methods) [8–10] does not need a lot of human knowledge in advance and may perform better in unknown and uncertain environments via trial-and-error learning, which can model complex environments using data-driven methods [11]. For example, AlphaGo shows superior performance via RL [12] as compared to humans in the Game of Go, which is one of the most challenging games that exist; Gu et al. provided a multi-agent RL method that could teach multiple robots how to run in a multi-agent system while considering the balance between the robots' rewards and safety, in which an agent not only needs to consider its own reward and other agents' rewards, but also its safety and other agents' safety in unstable environments [9].

Indeed, agents using RL methods can learn how to automatically navigate a vehicle via exploration and exploitation [13,14]. Therefore, combining RL with traditional methods is significant as it can achieve better performance in uncertain environments for vehicle motion planning. However, how to combine traditional methods with RL and consider safe, effective, and precise motion planning while leveraging the advantages of these two methods and overcoming their shortcomings is a challenge. In this study, a strategy is proposed in which RL is constrained, thus leveraging traditional methods such as the topological path search (TPS) and the trajectory lane model (TLM) to achieve safe, effective, and precise motion planning (named RLTT).

Several key issues need to be solved in this study: First, the routing points need to be considered in a dynamic model of vehicles, which can make motion planning more reasonable and closer to actual environments. Second, how to transform between RL and the topological path needs to be resolved because RL is a trial-and-error algorithm in which the search for paths for large-scale areas with respect to the search time is difficult. On the contrary, the method of TPS can easily and effectively plan a path for large-scale areas. Third, how to build a dynamic model for automated driving and provide dynamic constraints that can render safer, more effective, and more precise motion planning should be considered.

To settle the mentioned problems, we first propose a planning hierarchy framework with three levels, as shown in Figure 1. The first level is for macro-scale planning via TPS, and the second level is the RL method used for micro-scale planning; the differences between macro-scale and micro-scale planning can be found in reference [15]. The third level is TLM, which can make motion planning more precise and closer to actual environments. Second, we take into account that solving the interpretability and instability problems of deep learning (DL) is difficult because DL's ability is known, but its operations are unknown. Thus, from the perspective of the traditional methods, TPS and TLM are leveraged to ensure safety and to constrain the RL search space, where motion planning can be stabilized by combining their advantages and overcoming their shortcomings. Third, for uncertain environments such as an uncertain obstacle, RL is used to explore dynamic and uncertain environments, and then multiple constraints are considered and the safety range is set through a safe TPS buffer.

The contributions of our proposed method and model are as follows:

- The RLTT method along with a theoretical analysis is proposed for automated driving, in which a novel planning hierarchy framework of three levels and a multi-scale planning based on TPS, RL, and TLM are introduced.
- Multiple constraints of vehicles are considered, such as the dynamic constraints of vehicles, smooth constraints, and safety constraints, thereby making motion planning more reasonable and closer to actual scenarios.
- Uncertain environments are also considered in the proposed planning method, which achieves superior performance as compared to related works.
- Safe and efficient motion planning for automated driving is achieved. The RLTT method, by combining traditional methods and RL, can perform well under sparse information environments and multi-scale planning environments.

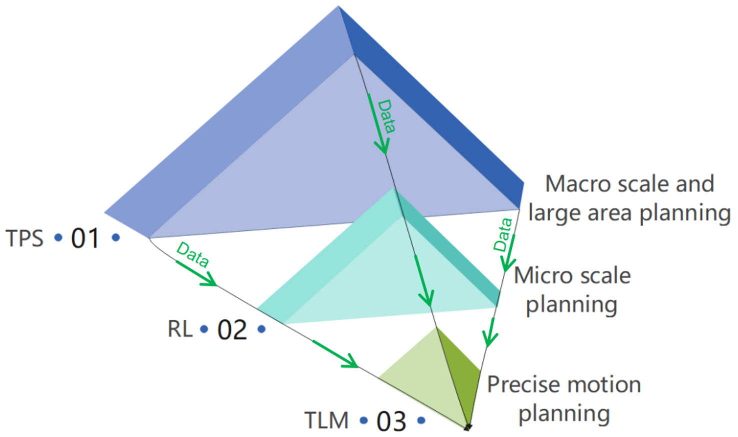


Figure 1. A framework of the RLTT method: a planning hierarchy framework of three levels for motion planning with data flow.

The remainder of this paper is organized as follows: Related works are introduced in Section 2; the dynamic model of a vehicle is provided in Section 3; TLM is presented in Section 4; the method of TPS is introduced in Section 5; the RLTT method for automated driving is introduced in detail in Section 6; related experiments are described in Section 7; the conclusion of the paper is given in Section 8.

2. Related Work

The planning for unmanned vehicles has attracted a lot of attention in the automated driving community [16], especially from the perspective of data-driven methods in complex environments which are unknown and uncertain.

For example, in [17], Bernhard and Knoll considered uncertain information using neural networks for automated vehicle planning. Nevertheless, they assumed all information about other vehicles to be known, and that this assumption might not be suitable for real environments. Zhang et al. [18] developed a novel bi-level actor–critic method for multi-agent coordination. Although their method achieved great success in making quick decisions regarding automated driving in highway merge environments, this method could not guarantee the safety of automated vehicles. Nick et al. [19] introduced an algorithm for clustering traffic scenarios, in which they combined convolutional neural networks and recurrent neural networks to predict traffic scenarios for the ego vehicle's decision-making and planning. However, their method might be unstable for some extreme traffic scenarios because of the uncertain and imperfect information on some traffic situations, such as intersection environments. In [20], a safe motion planning for autonomous vehicles based on RL and Lyapunov functions was proposed, where the reward function was optimized with respect to agent safety, and the balance between reward and safety was analyzed by optimizing the probability of collision. Similarly, uncertain environments were also not considered.

Chen et al. [21] developed an end-to-end method for automated driving using RL and a sequential latent environment model, which is a semantic bird-eye mask; their methods were more interpretable than other machine learning methods to some extent. However, their method may still need to further consider sparse information environments and multiple constraints for automated driving. Tang et al. [22] introduced a motion planning method for automated driving using a soft actor-critic method [23], in which different strategies were balanced via the weights of safety, comfort, and efficiency. Zhu et al. [24] developed a motion planning method to consider the factor of pedestrian distraction based on a rule-based method and a learning-based method. Their experimental results demonstrated that the learning-based method may be better at handling the unsafe action problem at unsignalized mid-block crosswalks than the rule-based method; nonetheless, the learning-based method may generate unreasonable actions. Wen et al. [3] provided a safe RL method for autonomous driving based on constrained policy optimization, in which the risk return was considered as a hard constraint during the learning process, and a trust region constraint was used to optimize the policy. Although they achieved better performance than CPO [25] and PPO [26] in some scenarios, their method considered risk as an optimization objective based on neural network learning. This learning process might encounter violations of the safety constraint, which will not allow for safety to be achieved with high confidence. In contrast to their methods, our method is based on a geometry reachability analysis and vehicle dynamic verification, which can guarantee safety during automated driving.

Shai et al. [27] presented a sophisticated strategy based on RL which considers negotiation strategies when the ego vehicle drives in complex environments. Their method can be divided into two strategies: one strategy that can be learned, and another strategy associated with hard constraints that cannot be learned (e.g., safety constraints). Although they tried their best to ensure the safety of automated driving, their method still required an improvement in adaptability for more complex environments such as a complex intersection with multiple heterogeneous vehicles and pedestrians. Sarah M. Thornton [28] proposed a method that leveraged the Markov decision process (MDP) and dynamic programming to control the vehicle speed for safety, and this method considered uncertain pedestrians at a crosswalk. However, the method might require improvements in order to consider more uncertain environments, and the search space may need to be reduced for efficient planning. Codevilla et al. [29] used condition imitation learning for a high-level command input to achieve automated driving in simulation environments and for a 1/5-scale robotic truck in real environments. Although both experiments evaluated the effectiveness of their method, the method may need to address the need of automated vehicles for human guidance in sparse information environments.

Moreover, there are many alternative methods available for solving robot motion planning problems, such as a probabilistic Chekov method for robots that plans via chance constraints [30], a probabilistic collision constraint planning based on Gaussian probability distribution functions [31], artificial potential fields for motion planning [32], symptotically optimal planning for robots on the basis of a rapidly exploring random tree (RRT) [33], direct sampling for robot path planning derived from RRT [34], a fast marching method for path planning [35], etc.; although the above methods have shown impressive progress in robot motion planning research, the methods may need to be further developed for autonomous driving and consider the features of vehicles and autonomous driving environments.

In this paper, the proposed RLTT is different from the above-mentioned methods because it can achieve safe and efficient automated driving in sparse information environments while considering multi-constraint and multi-scale motion planning.

In RLTT, TLM is developed on the basis of the trajectory unit, which was first proposed for unmanned surface vehicles (USVs) [36–38]. However, USVs are different from automated vehicles. The freedom of control, navigation environments, and vehicle shapes are different [15,39]. The trajectory unit may not be suitable for automated vehicles, therefore developing TLM for automated vehicles is necessary. Moreover, TLM is different from a lattice planner[40] because a lattice planner is achieved using sample and fit data, which may require a huge amount of time and more computing power. In contrast, TLM is achieved via a dynamic model of vehicles in advance. In addition, TPS is proposed to constrain the RL search space and provide routing points for RL navigation, which can improve RL efficiency. Finally, the hierarchy framework is proposed by integrating TPS, RL, and TLM to develop the RLTT method, which can make the proposed method and model more unified.

3. Problem Formulation

In this study, a constrained RL problem is considered for motion planning; in particular, uncertain constraints F_{un}, dynamic constraints F_{dy}, safety constraints F_{sa}, and smooth constraints F_{sm} are considered.

3.1. Uncertain Constraints F_{un}

For the convenience of description, we have provided the following definitions for (1): $O_{obstacle}$ denotes the obstacles; O_{shape} denotes the shape of the obstacles; $O_{position}$ denotes the position of the obstacles; O_{part} denotes partial information about the obstacles' positions and shapes.

Definition 1. *In uncertain environments, since information about an obstacle shape cannot be fully observed, $O_{observe}$ is used to denote partial information about the obstacles that the agent observes.*

$$O_{shape} \cup O_{position} = O_{obstacle} \supseteq O_{part} \supseteq O_{observe}. \tag{1}$$

3.2. Dynamic Constraints F_{dy}

Dynamic constraints F_{dy} can be defined as the dynamic model of vehicles, which is briefly shown in Figure 2. In this paper, a kinematics model, steer command, and heading angle are considered. The kinematics model is briefly introduced in this study [41,42]. In general, the steer command range is set to approximately $-30° \sim 30°$, or to $-40° \sim 40°$ for other settings, and the setting of the steering command of our method is suitable for and can be applied to real vehicle experiments.

The driving speed at the axle center of the rear axle is v_r. (X_r, Y_r) represents the coordinates at the axis of the rear axle, and φ represents the heading angle.

$$v_r = \dot{X}_r cos\varphi + \dot{Y}_r sin\varphi. \tag{2}$$

The kinematic constraints of the front and rear axles are as follows: δ_f represents the front wheel steering angle, which is similar to the steering command, and it is used to name the steering command here.

$$\begin{cases} \dot{X}_f sin(\varphi + \delta_f) - \dot{Y}_f cos(\varphi + \delta_f) = 0 \\ \dot{X}_r sin\varphi - \dot{Y}_r cos\varphi = 0, \end{cases} \tag{3}$$

where l indicates the distance between the rear axle and the front axle (wheelbase), w represents the yaw rate, and R denotes the turning radius of the vehicle. The geometric relationship between the front and rear wheels is as follows:

$$\begin{cases} X_f = X_r + l cos\varphi \\ Y_f = Y_r + l sin\varphi, \end{cases} \tag{4}$$

$$\begin{cases} R = v_r / w \\ \delta_f = arctan(l/R). \end{cases} \tag{5}$$

According to the analysis above, the kinematic constraints can be briefly summarized in Equation (6):

$$\begin{bmatrix} \dot{X}_r \\ \dot{Y}_r \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} cos\varphi \\ sin\varphi \\ 0 \end{bmatrix} * v_r + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} * w. \tag{6}$$

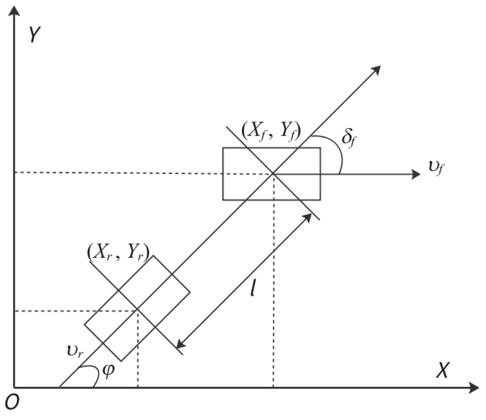


Figure 2. A briefly diagram of the vehicle dynamic constraints.

3.3. Safety Constraints F_{sa}

The safety distance is D_{safety}, and the distance between the point of motion planning P_{point} and obstacles $O_{obstacle}$ is D_p.

Definition 2. *The safety constraints (F_{sa}) are represented by Equation (7):*

$$D_{safety} \leq D_p. \tag{7}$$

3.4. Smooth Constraints F_{sm}

The motion planning points are represented by the set P, $\sum_{n=1}^{N} p_n \subseteq P$. The two adjacent points are p_1 and p_2. The two-steer angle difference of any two adjacent points is δ_{normal}. The limitation of the two-steer angle difference of any two adjacent points is $\delta_{limitation}$, where $\delta_{limitation}$ is set to smoothen the steer angle.

Definition 3. *The smooth constraints F_{sm} can be represented by Equation (8):*

$$\delta_{normal} \leq \delta_{limitation}. \tag{8}$$

In conclusion, we need to find a proximal optimal motion planning set $\boldsymbol{P_{op}}$ for vehicles that can also simultaneously satisfy the constraints F_{un}, F_{dy}, F_{sa}, and F_{sm}.

Definition 4. *The objective needs to satisfy the following constraints, which can be formulated as follows:*

$$\boldsymbol{P_{op}} \subseteq \left(F_{un} \cap F_{dy} \cap F_{sa} \cap F_{sm}\right). \tag{9}$$

4. Trajectory Lane Model

The trajectory lane model (TLM) can be considered as a bridge that connects the routing planning and the dynamic constraints of a vehicle, thus allowing the motion planning to become closer to actual environments, as shown in Algorithm 1. TLM is constructed according to the dynamic model of the vehicle, and relative TLM rules are introduced to achieve effective and precise motion planning.

Algorithm 1: Generating Trajectory Lane.

```
Input: The acceleration $a$, velocity $v$, position $(x, y)$, front wheel steering angle $\delta$,
       differential time $dt$, interval time $T$, heading angle $\psi$, wheelbase$frlen$, $i$
       $\leftarrow 0$, $t \leftarrow 0$;
Output: Trajectory lane model m;
for $i$ do
    $i \leftarrow i + 1$;
    $x' \leftarrow x + v * cos(\psi)dt$;
    $y' \leftarrow y + v * sin(\psi)dt$;
    $\psi' \leftarrow \psi + (v/frlen) * \delta * dt$;
    $v' \leftarrow v + a * dt$;
    $x \leftarrow x', y \leftarrow y', \psi \leftarrow \psi', v \leftarrow v'$;
    $t \leftarrow t + dt$;
    if $t <= T$ then
        Break.
    end
end
```

4.1. Rules Design

Based on the analysis above, the following rules have been designed:

- Rule one: The trajectories of each of the actions are equal in length. This is for the continuous, regularized, and easily spliced trajectories.
- Rule two: Each trajectory has only one steer command, except for the start and end steer commands; this is for the smooth steer. The relative angle (steering angle) between two adjacent points is not greater than one degree; this is for generating a smooth trajectory.
- Rule three: The trajectory of each action has the same speed; this is for those with equal length. At the start and end of the trajectory, the condition is the same.

4.2. Analysis of TLM

According to the rules above and the dynamic model of a vehicle, TLM can be constructed. The introduction of the construction of the TLM m can be found in Algorithm 1, in which several types of TLMs are constructed on the basis of the constraints of the action and the state space.

(1) TLM—type one: This is the simplest type, which only has eight direction actions, and it can be seen in Figure 3a. This kind of TLM does not have sufficient actions to achieve motion planning, and the model can not achieve curve turning.

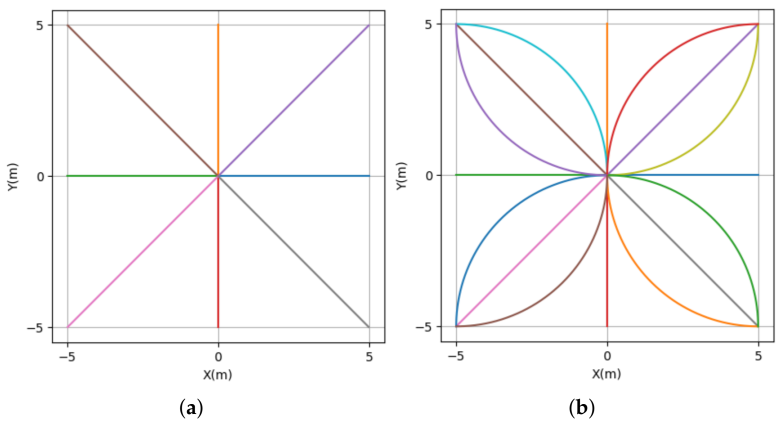


Figure 3. Straight-line trajectory lane and semicircle trajectory lane models. (**a**) Straight-line trajectory in 8 action directions. (**b**) Semicircle trajectory in 16 action directions.

(2) TLM—type three: This type of TLM, with 16 direction actions, has more actions than the first type, which has eight direction actions (as shown in Figure 3b). It can comprise the second type, including four direction-action positive semicircles and four direction-action negative semicircles (as shown in Figure 4b). This type of TLM still does not have sufficient actions to achieve motion planning, and sometimes this type of TLM for motion planning is locally optimal.

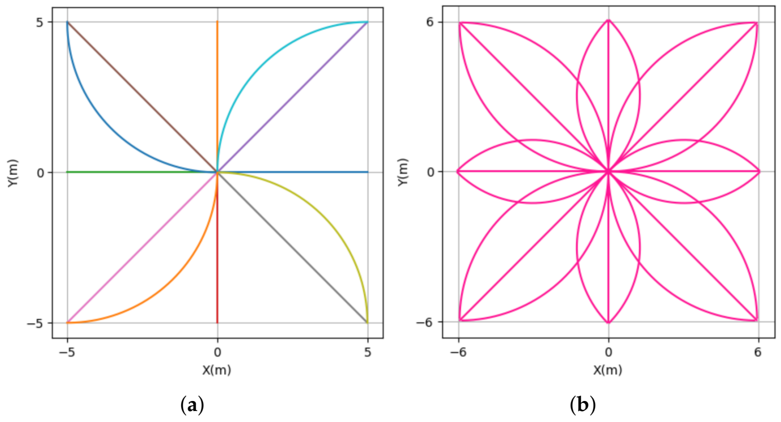


Figure 4. Semicircle trajectory lane model. (**a**) Negative semicircles in 12 directions based on 8 actions; (**b**) A 24 direction-action TLM.

(3) TLM—type four: This type of TLM, with 24 direction actions, has more actions than the second type and can comprise the second type and 8 direction-action trajectories near the maximum rudder angle along the x and y axes; this can be seen in Figure 4b. This type of TLM is similar to the third type, and sometimes the type of TLM for motion planning is also locally optimal.

(4) TLM—type five: This type of TLM, with 40 direction actions, has more actions than the third type and can comprise the third type and 8 direction-action smooth semicircles (Figure 5a); this can be seen in Figure 5b. To some extent, this type of TLM has sufficient actions to achieve motion planning and can achieve proximal optimal motion planning. Figure 6 shows the 40 direction-action TLM with one example of a smooth circle. A theoretical analysis is provided in Lemma 1, which can prove that type four is the proximal optimal trajectory; in Theorem 1, completeness of trajectory space is proved.

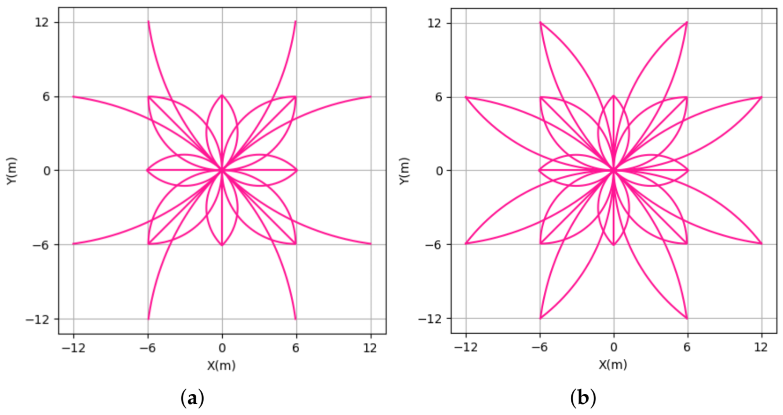


Figure 5. Semicircle trajectory lane model. (**a**) A 32 direction-action TLM. (**b**) A 40 direction-action TLM.

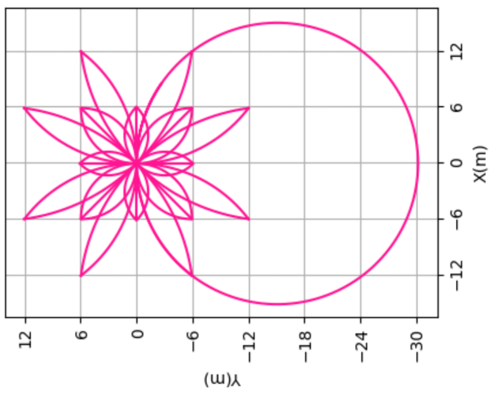


Figure 6. A 40 direction-action TLM, with one example of a smooth circle.

Lemma 1. *The proximal optimal type of TLM for motion planning can be considered to be TLM's type five, where the TLM has 40 search directions for motion planning.*

Proof. Taking type one of TLM as an example, it can be transformed into the optimal distance problem.

A trajectory is $T = \{p_1, p_2, \cdots, p_i, \cdots, p_n\}$, a point i is $p_i = \{t_i, x_i, y_i, \delta_i, v_i, a_i\}$, the distance between any two adjacent points is $d_{i,i+1} = \sqrt{(x_{i+1} - x_i) + (y_{i+1} - y_i)}$, the set of trajectories in the space is expressed as $\boldsymbol{T} = \{T^1, T^2, \cdots, T^j, \cdots, T^m\}$.

The trajectory T_j can be presented as $y_i^j = A^j(x_i^j)^2 + B^j x_i^j + C^j$, A^j, B^j, and C^j are the correlation coefficients. According to the law of a triangle, the sum of two sides is greater than the third side. j denotes the trajectory type, j_h denotes the h trajectory of the j trajectory type, i^{j_h} denotes the i point of the h trajectory of the j trajectory type.

$$\begin{aligned} d^{j_0}_{i^{j_0},i^{j_0}+1} &= \sqrt{(x^{j_0}_{i^{j_0}+1} - x^{j_0}_{i^{j_0}})^2 + (y^{j_0}_{i^{j_0}+1} - y^{j_0}_{i^{j_0}})^2} \\ &= \sqrt{(x^{j_0}_{i^{j_0}+1} - x^{j_0}_{i^{j_0}})^2 + \left[(A^{j_0}(x_{i^{j_0}+1})^2 + B^{j_0}x^{j_0}_{i^{j_0}+1} + C^{j_0}) - (A^{j_0}(x^{j_0}_{i^{j_0}})^2 + B^{j_0}x_{i^{j_0}} + C^{j_0})\right]^2} \\ &\forall \quad 0 \le i^{j_0} \le n^{j_0}, 0 \le j^h \le m^{j_h}, \end{aligned} \tag{10}$$

$$\begin{aligned} d^{j_1}_{i^{j_1},i^{j_1}+1} &= \sqrt{(x^{j_1}_{i^{j_1}+1} - x^{j_1}_{i^{j_1}})^2 + (y^{j_1}_{i^{j_1}+1} - y^{j_1}_{i^{j_1}})^2} \\ &= \sqrt{(x^{j_1}_{i^{j_1}+1} - x^{j_1}_{i^{j_1}})^2 + \left[(A^{j_1}(x^{j_1}_{i^{j_1}+1})^2 + B^{j_1}x^{j_1}_{i^{j_1}+1} + C^{j_1}) - (A^{j_1}(x^{j_1}_{i^{j_1}})^2 + B^{j_1}x^{j_1}_{i^{j_1}} + C^{j_1})\right]^2} \\ &\forall \quad 0 \le i^{j_1} \le n^{j_1}, 0 \le j^h \le m^{j_h}, \end{aligned} \tag{11}$$

According to the law of a triangle, the sum of two sides is greater than the third side.

$$\begin{aligned} d^{j_{0,1}}_{i^{j_0},i^{j_0}+1,i^{j_1},i^{j_1}+1} &= \sqrt{(x^{j_1}_{i^{j_1}+1} - x^{j_0}_{i^{j_0}})^2 + (y^{j_1}_{i^{j_1}+1} - y^{j_0}_{i^{j_0}})^2} \\ &= \sqrt{(x^{j_1}_{i^{j_1}+1} - x^{j_0}_{i^{j_0}})^2 + \left[(A^{j_1}(x^{j_1}_{i^{j_1}+1})^2 + B^{j_1}x^{j_1}_{i^{j_1}+1} + C^{j_1}) - (A^{j_0}(x^{j_0}_{i^{j_0}})^2 + B^{j_0}x^{j_0}_{i^{j_0}} + C^{j_0})\right]^2} \\ &\forall \quad 0 \le i^{j_0} \le n^{j_0}, 0 \le i^{j_1} \le n^{j_1}, 0 \le j^h \le m^{j_h}. \end{aligned} \tag{12}$$

$$d^{j_{0,1}}_{i^{j_0},i^{j_0}+1,i^{j_1},i^{j_1}+1} < d^{j_1}_{i^{j_1},i^{j_1}+1} + d^{j_0}_{i^{j_0},i^{j_0}+1}, \tag{13}$$

$$\boldsymbol{D}^{j_0} = \sum_{i}^{n^{j_0}} d^{j_0}_{i^{j_0},i^{j_0}+1}, \tag{14}$$

$$\boldsymbol{D}^{j_1} = \sum_{i}^{n^{j_1}} d^{j_1}_{i^{j_1},i^{j_1}+1}, \tag{15}$$

$$\boldsymbol{D}^{\boldsymbol{j_{0,1}}} = \sum_{i}^{n^{j_{0,1}}} d^{j_{0,1}}_{i^{j_{0,1}}, i^{j_{0,1}}+1}, \tag{16}$$

$$\boldsymbol{D}^{\boldsymbol{j_{0,1}}} < \boldsymbol{D}^{\boldsymbol{j_0}} + \boldsymbol{D}^{\boldsymbol{j_1}}. \tag{17}$$

In addition, this is similar to other types of TLMs and curve constraints (or steering smooth constraints), and the fifth type of TLM can be proven to be the proximal optimal TLM for motion planning. With an excessive number of trajectories, the search complexity will be large, and the fifth type of TLM will be sufficient for the trajectory probability. Completeness of trajectory space is subsequently proven.

Based on the Minkowski inequality and its sum [43], we can observe that $\boldsymbol{D}^{\boldsymbol{j_{0,1}}}$ is the proximal optimal trajectory, and the fifth type is the proximal optimal TLM. □

Theorem 1. *Completeness of trajectory space: any target point in space $\mathbb{R}^n$ can be reached using TLM. The number of optimal trajectory circles to cover the convex hull P' is $N^{\frac{d}{\sqrt{\beta_i}}^2}$, in which the number of vertices of the convex hull P' is N, the diameter of a convex hull P' is d, and the radius of any circle of the proximal optimal TLM is β_i.*

Proof. It can be transformed into the completeness of the quadratic equation problem.

$$\sum_{i=1}^{m} \lambda_i z_i \quad \text{where} \quad \lambda_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{m} \lambda_i = 1, \tag{18}$$

$$\text{conv}(P') := \big\{ \text{ convex combinations of } z_1, \ldots, z_m \in P' \text{ for } m \in \mathbb{N} \big\}. \tag{19}$$

Based on Caratheodory's theorem [44], we can observe that for any point p_i in a convex hull P', p_i can be covered with a set of less than n_d points in the convex hull P', n_d is the dimension of the convex hull P', and this will depend on the dimension of the convex hull P', $P' \in R^n$, whose proof can be seen in reference [44]. Derived from Caratheodory's theorem, an approximate Caratheodory's theorem is proposed [45], which does not depend on the dimension of the convex hull P'. We prove the space completeness of the trajectories of the TLM based on the approximate Caratheodory's theorem.

x_i ($i \subseteq \mathbb{N}$) is any point that belongs to a convex set T', with the set being bounded by a circle (its diameter is within 1) and an integer k. The proximal optimal TLM, which can be regarded as three different circles, is shown as Figure 7a; these three circles can cover any reachable areas under vehicle dynamic constraints, implying that the proximal optimal TLM can achieve the completeness of trajectory space for motion planning.

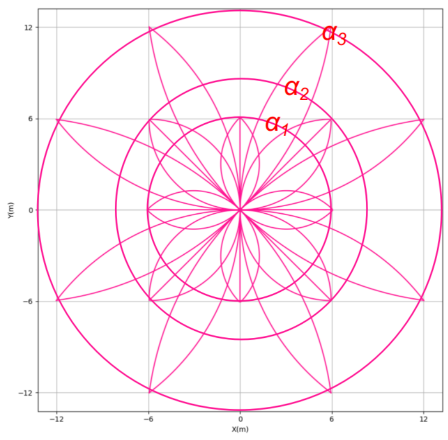
Next, we will prove the completeness of the trajectory space by using any circle from our proximal optimal TLM; this is shown in Figure 7b. The set of any convex hull P' is T', the number of vertices of the convex hull P' is N, the diameter of a convex hull P' is d, and the radius of any circle of the proximal optimal TLM is β_i. λ_i is the probability of point p_i, and x is a point within the reachable areas.

$$\mathbb{P}\{p = p_i\} = \lambda_i, \quad i = 1, \ldots, m, \tag{20}$$

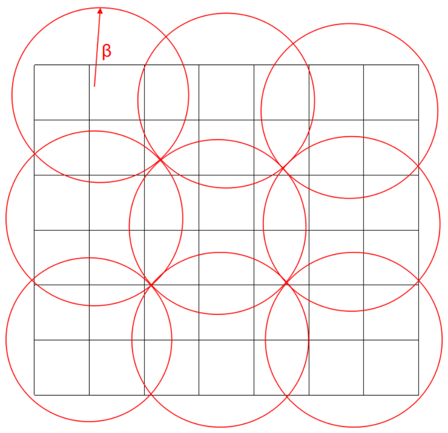
$$\mathbb{E}p = \sum_{i=1}^{m} \lambda_i p_i = x. \tag{21}$$

Based on the strong law of large numbers, we can obtain the following equation:

$$\frac{1}{k} \sum_{j=1}^{k} p_j \rightarrow x. \tag{22}$$



(**a**)



(**b**)

Figure 7. (**a**) Different circles of proximal optimal TLM that cover different points. (**b**) Covering areas that satisfy the vehicle's dynamic constraints using the proximal optimal TLM.

According to the weak law of large numbers, we can compute the variance of $\frac{1}{k}\sum_{j=1}^{k} p_j$,

$$\begin{aligned}\mathbb{E}\|p_j - x\|_2^2 &= \mathbb{E}\|p - \mathbb{E}p\|_2^2 \\ &\leq \mathbb{E}\|p\|_2^2 \leq d,\end{aligned} \tag{23}$$

moreover, we can observe that

$$\begin{aligned}\mathbb{E}\left\|x - \frac{1}{k}\sum_{j=1}^{k} p_j\right\|_2^2 &= \frac{1}{k^2}\mathbb{E}\left\|\sum_{j=1}^{k}(p_j - x)\right\|_2^2 \\ &= \frac{1}{k^2}\sum_{j=1}^{k}\mathbb{E}\|p_j - x\|_2^2.\end{aligned} \tag{24}$$

Therefore,

$$\mathbb{E}\left\|x - \frac{1}{k}\sum_{j=1}^{k} p_j\right\|_2^2 \leq \frac{d}{k}. \tag{25}$$

For the random variables, we can have

$$\left\|x - \frac{1}{k}\sum_{j=1}^{k} p_j\right\|_2^2 \leq \frac{d}{k}. \tag{26}$$

Thus, for any radius β_i of any TLM circle that satisfies the vehicle constraints, we can observe that

$$\left\|x - \frac{1}{k}\sum_{j=1}^{k} x_j\right\|_2 \leq \frac{d}{\sqrt{k}} \leq \beta_i, \tag{27}$$

which is derived from [44], and there are N^k ways to choose k with repetition to cover the cardinality of N. Thus,

$$N \leq N^k = N^{(\frac{d}{\beta_i})^2}. \tag{28}$$

The number of TLM circles to cover reachable areas is $N^{(\frac{d}{\beta_i})^2}$, implying that we can use the $N^{(\frac{d}{\beta_i})^2}$ TLM circles to cover any convex hull P'; the area is also reachable by using the proximal optimal trajectory circle, which finishes the proof. □

5. Topological Path Search for Automated Driving

For a large-scale and efficient path search, a topological map meant for high path search efficiency is constructed [46,47]. During the path searching process, the open geospatial consortium (OGC) is used to make a topological map [48]. OGC defines a simple feature model that is suitable for efficiently storing and accessing geographic features in relational databases.

A spatial map comprises nodes, lines, and a surface, which are three elements for constructing a map: a node is a point that does not have spatial features such as shape and size; a line comprises a series of nodes, whose spatial features include shape, length, etc.; the surface comprises closed rings, whose spatial features include shape, area, etc. In TPS, a node denotes the vehicle, a line denotes the routing path, a surface denotes the obstacles.

Spatial relationship refers to the spatial characteristic relationship between geographical entities, which is the basis of spatial data organization, query, analysis, and reasoning. It includes topological, metric, and direction relationships. Numerous topological relation expression models have been proposed to represent spatial relationships. In this study, the nine-cross model [48] is used to represent the space topological relationship used for the construction of a topological map. This is introduced through Function (29), where A and B represent two different objects, $(\eth)$ denotes the edge of an object, $(^\circ)$ denotes the internal part of an object, $(^-)$ denotes the external part of an object, and $(\bigcap)$ denotes the intersection of two objects.

Based on Function (29), the topological relation predicates are defined, which are Crosses, Disjoint, Within, Contains, Equals, Touches, Intersects, and Overlaps. In this paper, we leverage Intersects, Disjoint, Touches, and Contains to analyze the topological relationship shown in Figure 8. For example, $[FF*FF****]$ denotes the Disjoint; $[FT*******]$, $[F**T*****]$, and $[F***T****]$ denote Touch; $[T*F**F***]$ and $[T*****FF*]$ denote Contains/Within; $[T********]$, $[*T*******]$, $[***T*****]$, and $[****T****]$ denote Intersects via the nine-cross model. For two objects in space, a and b, the intersection value (ϕ) is denoted by F (False), the value $-(\phi)$ is denoted by T (True); the intersection is represented by 0 when the point is the intersection, by 1 when the line is the intersection, and by 2 when the area is the intersection. $*$ means that f, 0, 1, or 2 can be selected. More specifically, a. Disjoint b means two geometric bodies (a and b) have no common intersection, thus forming a set of disconnected geometric shapes.

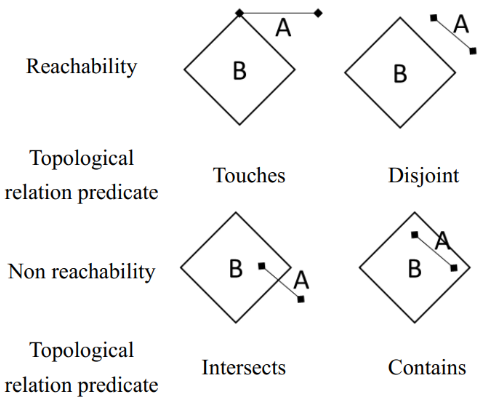


Figure 8. Schematic diagram of the topological path reachability.

After constructing the topological map, the Dijkstra algorithm [49] is used to achieve an optimal routing planning, and the Euclidean metric is used to measure the distance between two points, which is provided by Function (30). The algorithm is described in Algorithm 2, in which Function (29) is used to judge the topological relationship between obstacles O_n, the start point S, and the end point E; Function (29) is used to judge the topological position relationship between the path point and each point O_{ij} of the ith obstacle O_i.

In addition, the safety buffer is set according to topological relationships. Moreover, the routing path can be achieved by the Dijsktra algorithm using Function (30) based on the constructed topological map; the Dijskra algorithm can achieve a more optimal path than other heuristic algorithms, e.g., the A* algorithm, even though it requires more computing time. However, in this study, the Dijskra algorithm based on TPS could quickly find the optimal path and satisfy motion planning constraints.

$$\begin{pmatrix} A^\circ \bigcap B^\circ & A^\circ \bigcap \eth B & A^\circ \bigcap B^- \\ \eth A \bigcap B^\circ & \eth A \bigcap \eth B & \eth A \bigcap B^- \\ A^- \bigcap B^\circ & A^- \bigcap \eth B & A^- \bigcap B^- \end{pmatrix}, \tag{29}$$

$$d(u,v) = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}. \tag{30}$$

According to the size of the vehicle maneuver, the size of the safety buffer is set at 8 (m), which indicates that the distance between the path and the obstacles is always 8 (m), thus making the route planning safe and suitable for motion planning.

Algorithm 2: The method of TPS for automated driving.

```
Input: The set of obstacles $O_n$; the start point $S$ and the end point $E$, the number of
       the $i$th obstacle of all points $OP_i$, path point $MP$, $i \leftarrow 0$;
Output: Macro-scale routing point $MP_n$;
for $i$ do
    $i \leftarrow i + 1$;
    $j \leftarrow 0$;
    for $j$ do
        $j \leftarrow j + 1$;
        Compute (29) to judge the relationships between the $MP$ and $O_{ij}$ of the $i$th
          obstacle $O_i$;
        if $j$ = $OP_i$ then
            Break;
        end
    end
    if $j$ = $O_n$ then
        Break;
    end
end
Set the safety buffer and store the topological map;
Call Dijkstra algorithm and compute function (30) for topological path $MP$, and
  add $MP$ to $MP_n$;
return $MP_n$.
```

6. A Motion Planning Method for Automated Driving

TLM and TPS were constructed in the previous sections. In this section, we will discuss how to achieve motion planning for automated driving. RL is briefly introduced [8], the RLTT method for automated driving is then presented in detail, and multiple constraints are considered in the RLTT method.

TPS is introduced for large and macro-scale motion planning, in which both driving safety and optimal path planning are achieved via a safety buffer, a topological map, and the Dijkstra algorithm. A representation of TPS constraints is given in Figure 9a. For micro-scale motion planning, in which the environment contains uncertain and imperfect information, building a model and learning experiences via a data-driven method is easier than traditional methods; in contrast, when the environment is uncertain, the environment is difficult to model via traditional methods. Figure 9b shows the representation of constraints through RL. TLM can be used for dynamic constraints and smooth constraints, in which the kinematic model of vehicles and steer constraints as well as the relative angle between any two adjacent points are also limited; the representation of TLM is given in Figure 9c. Considering this type of framework and environment is useful because the perfect environment information cannot be obtained in a real environment even if advanced sensors are used. The proposed method could aid in the advancement of automated driving.

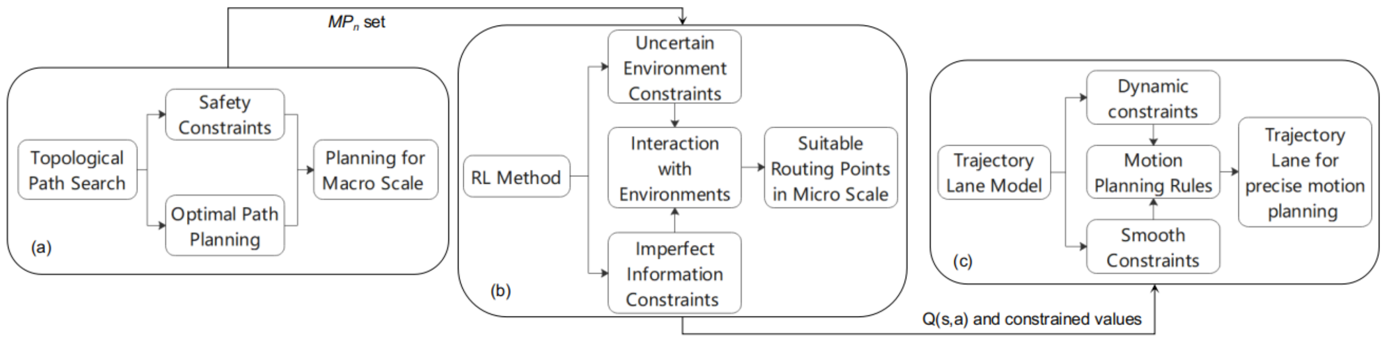

MP_n set
Topological Path Search
Safety Constraints
Optimal Path Planning
Planning for Macro Scale
(a)
RL Method
Uncertain Environment Constraints
Interaction with Environments
Imperfect Information Constraints
Suitable Routing Points in Micro Scale
(b)
Trajectory Lane Model
Dynamic constraints
Motion Planning Rules
Smooth Constraints
Trajectory Lane for precise motion planning
(c)
Q(s,a) and constrained values

Figure 9. TPS, RL, and TLM for automated driving. (**a**) Representation of TPS constraints. (**b**) Representation of RL constraints. (**c**) Representation of TLM constraints.

6.1. *RL Methods*

RL can be typically regarded as MDP, and MDP is presented by a five-object tuple (S, A, P, R, γ), where S denotes the state space, A denotes the action space, P shows the probability of transition, R represents the reward once one action is performed, and γ denotes the discounted factor of the reward. In this study, RL is regarded as the partial observation MDP, where RL makes an agent interact with the environments through trial-and-error learning in order to obtain more data about the environments, and the agent can then learn better about the environments as well as perform better. In particular, RL can deal with uncertainty problems better than traditional methods. Notably, the most likely action can be selected in the future environment on the basis of previous exploration. It can reduce the uncertainty about the environment, which means that more knowledge can be acquired about the environment, and more certain information can be obtained for future decision making.

6.2. *A Motion Planning Method*

First, the TLM and TPS are introduced into the framework, and Q learning is developed based on the TLM and TPS, providing the constrained RL for safe motion planning.

Second, the transition function f of Q learning is integrated into TLM and TPS. More specifically, for the transition function f, the input is action a, the current position is P_c, the angle is A, and the terminal is P_T; the output is the next position P_n, the reward is r, done is T, and the next angle is A_n. The transition process is as follows: If the current position P_c has collided with an obstacle $O_{obstacle}$, then return the state s, reward r, done, and angle A; if the current position P_c has reached the terminal P_T, then return state s, reward r, done, and angle A; at the same time, if angle A and action a are equal to angle A_t and action a_t of the trajectory, respectively, we can obtain the next position $P_n \leftarrow$ current position P_c, next angle $A_n \leftarrow$ angle A, and obtain the related reward r. If the next position P_n has collided with an obstacle $O_{obstacle}$, then return the state s, reward r, done T, and angle A; if the next position P_n has reached the terminal P_T, then return the state s, reward r, done T, and angle A. Iteratively proceed according to this logic, obtain the related path information, and return each position's next position P_n, reward r, done T, and next angle A_n for further motion planning.

Third, the algorithm of Q learning is presented for the path value, which can be seen in Algorithm 3. Finally, according to the distance of the safety buffer, select the routing points and call Algorithm 4 for the automated driving motion planning.

Algorithm 3: Q learning method for the path value.

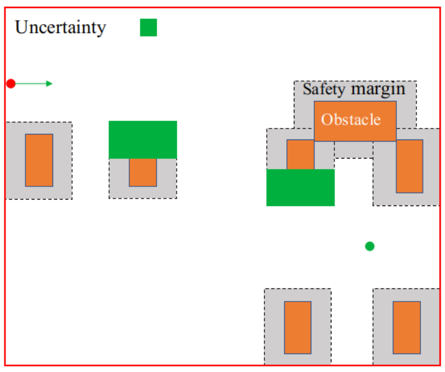
```
Input: The start point $S_{ij}$ and end point $E_{ij}$ of each segment from the $MP_n$ set
       $(i = 1, 2, \ldots, n, j = 1, 2, \ldots, n')$, $S'$ represents the reachable points;
Output: The $Q(s, a)$ in each segment;
Initialize the Q value $(s, a)$, $\forall\ s \in S$, $a \in A$, set parameter $\alpha$, $\gamma$ ;
for episode $z = 0$ to $Z$ do
    Select the action $a$ according to the initial state $s$ and $\epsilon$-greedy strategy;
    while $s$ is not the terminal do
        Conduct $a$, then get reward $r$, next angle $\psi'$ and next state $s'$ according to
          transition strategy $f$, angle $\psi$, and state $s$;
        if next state $s' \in S'$ then
            $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma maxQ(s', a') - Q(s, a)]$;
            $s \leftarrow s'$, $a \leftarrow a'$;
        end
    end
    if $s$ is terminal then
        Break;
    end
end
return $Q(s, a)$.
```

Algorithm 4: Motion planning for automated driving via RLTT.

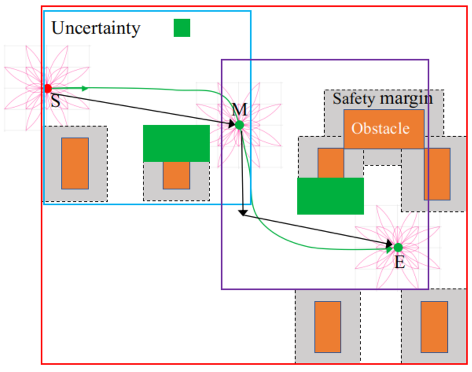
```
Input: The start point $S_{ij}$ and end point $E_{ij}$ of each segment from the $MP_n$ set
       $(i = 1, 2, \ldots n, j = 1, 2, \ldots n')$, and vehicle environments (trajectory)
       generated via Algorithms 1 and 2;
Output: The command steer of each point $\delta$; the position of each point in each
        segment $(X_{ij}, Y_{ij})$; the heading angle $\psi$ of each point during the segment;
Initialize the initial state $s$ (including $S_{ij}$), heading angle $\psi$, learning rate $\alpha$, greedy
  strategy parameter $\epsilon$, call Q learning, and return Q value via Algorithm 3;
while trajectory $t$ not get done do
    Append state $s$ into the path and select the action $a$ according to the initial state
      $s$ and the greedy strategy;
    Add state $s$ into a specific position via Algorithm 3 and append position,
      action, and angle into motion planning table $f1$;
    Conduct $a$, then obtain reward $r$, next angle $\psi'$, and next state $s'$ according to
      the transition strategy $f$, angle $\psi$, and state $s$;
end
for $i$ = 0 to the length of motion planning steps do
    Call trajectory lane model;
    if angle and action of motion planning $f1$ are equal to the angle and action of
      trajectory $m$ then
        Record the trajectory $m$;
    end
end
return Continuous motion planning trajectory $m$.
```

6.3. Motion Planning Examples

A motion planning example of automated driving via RLTT is illustrated in this section. The problem of motion planning in certain and perfect information environments may be easy to solve using the traditional methods such as the sample-based method. However, if uncertain environments and vehicle constraints are considered, the traditional methods may not perform well. For example, in Figure 10a, the red circle point represents the start point, the green circle point represents the end point, and the green arrow represents the dynamic constraints, including the heading constraints and steer constraints, among others. When the environment is 65% certain and 35% uncertain, for example, 35% of the obstacles expand randomly upward or downward by 35%. More specifically, the position and shape of 65% of the obstacles are certain, and the position and shape of 35% of the obstacles are uncertain. When the vehicle is navigating, 35% of the obstacles change at random. The RLTT method can perform well in the environments because it can efficiently and rapidly explore the unknown world in micro-scale motion planning (blue box (from the S point to the M point) and purple box (from M point to E point)). More specifically, it leverages TPS for large and macro-scale routing planning (red box (represented by the black arrow from the S point to the E point)) and TLM for micro and precise motion planning (deep pink line). An example is given in Figure 10b, with the motion planning trajectory represented by the green arrow from the S point to the E point. Furthermore, we provide the complexity analysis of RLTT in Remark 1.



(**a**)



(**b**)

Figure 10. Certain and perfect information environments for vehicles. (**a**) does not consider dynamic constraints, (**b**) considers dynamic constraints.

Remark 1. *The proximal optimal search time for motion planning with a constrained RL is as follows:*

The search time for TPS is $\mathcal{O}(M^2)$, where M represents the number of obstacle points; for the RL search time, the complexity is $\mathcal{O}(T)$, where T is the total number of steps [50].

More particularly, the convergence of our method can be guaranteed based on the Banach fixed-point theorem [51]; for the detailed convergence proof of the constrained RL, see [52].

7. Experiments

Several experiments were conducted in the same environments, and the results are analyzed here in detail. First, different RL methods for path search were tested. Second, the RLTT method for motion planning was achieved for automated driving. Third, comparison experiments in certain and uncertain environments have been carried out. In addition, comparison experiments between RLTT and RL algorithms were provided. Fourth, comparison experiments between RLTT and traditional algorithms were conducted to evaluate the effectiveness of our proposed methods. Finally, RLTT for automated driving motion planning in uncertain corridor environments was carried out to demonstrate our method's applicability.

Moreover, all experiments were run on an Ubuntu 18.04 system (Dell PC), in which the CPU was an Intel Core i7-9750H CPU 2.60 GHz $\times$ 12, the GPU was an Intel UHD Graphics

630 (CFL GT2), the Memory was 31.2 GB; all comparison experiments were conducted under the same environment conditions, and the computation time in the experiments included training time and test time. The number of initial iteration steps was 10,000, the learning rate was 0.1, the epsilon value of the maximizing optimal value was 0.9. For the reward settings, we set them on the basis of the safety cost, distance cost, and steering cost of the vehicle. If the vehicle collided with obstacles, the reward was -10; if the vehicle found the target position with the correct attitude, the reward was 10; the reward for each step the vehicle took along the straight line was -0.1; for each step the vehicle took along the diagonal direction, the reward was -0.14; for each step the vehicle took along the oblique curve, the reward was -0.157; the reward was -0.16 for every turn in the steering action; if the vehicle took each step along the diagonal and straight direction, the reward was -0.196.

7.1. Different RL Methods for Path Search

Related RL methods for conducting the path planning for automated driving have been performed, which can be seen in Figure 11. The figure shows that the dynamic programming (DP) value and policy iterations [8] achieved fewer steps, implying that the two algorithms can obtain the shortest path for automated driving. Nonetheless, DP value and policy iterations may be unsuitable for automated driving in uncertain and unknown environments because the probability and reward transition might not be known, and the above two methods typically need information regarding the model of environments.

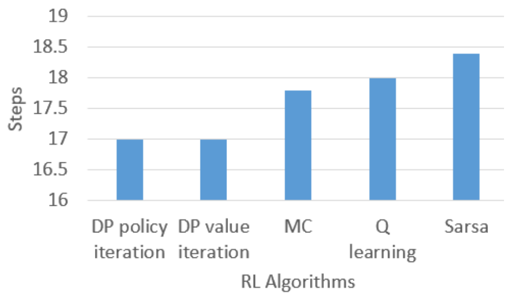


Figure 11. Different RL methods for a path search.

In addition, the Monte Carlo (MC) RL method [8] achieved almost the same number of steps as the Q learning [53] for automated driving. However, the MC method sometimes falls into a deadlock during exploration and cannot obtain the desired path due to the complex environments. The Sarsa method [54] obtains more steps than Q learning because of its conservative strategy. Because RLTT is a multi-scale method and the safety buffer constraints of the first level via TPS have been further considered, it is better to develop Q learning and integrate it into the RLTT framework for automated driving. In the next section, the experiments of the RLTT method for automated driving are introduced and discussed.

7.2. A RLTT Method for Motion Planning

7.2.1. Macro-Scale Motion Planning

In this section, macro-scale planning (generally, the planning area can be set from approximately 100 to 2000 (m)) was accomplished via TPS (Figure 12a,b), which represents the convex obstacle and concave-convex obstacle environments, respectively, in which the macro-scale planning is represented by the deep pink line and the obstacles are represented by the red rectangles. The start point and end point are S and S_3 in Figure 12a and S' and S'_3 in Figure 12b, respectively. The safety buffer was set at 8 m, as shown in Figure 12a,b (denoted by B and the yellow arrow).

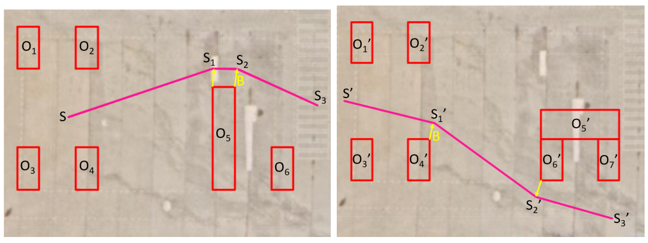


Figure 12. Macro-scale planning. (**a**) Convex obstacle environments. (**b**) Concave-convex obstacle environments.

7.2.2. Micro-Scale Motion Planning

The micro-scale motion planning (generally, the planning area can be set at approximately 100 m) was achieved, as shown in Figures 13 and 14, where the environment information is unknown. The start point is the S point and the end point is the E point. The obstacle is denoted by O. RL was constrained by TPS and TLM, and it was used to dynamically explore the suitable point from the S to the E points. TLM was used to splice the RL points, where the smooth and dynamic constraints were considered; the two bottom obstacles randomly changed within a certain range. In particular, obstacles O_3 and O_4 randomly changed within the O_{31} and O_{41} ranges (as shown in the light blue rectangles); therefore, the motion planning is different in Figures 13 and 14.

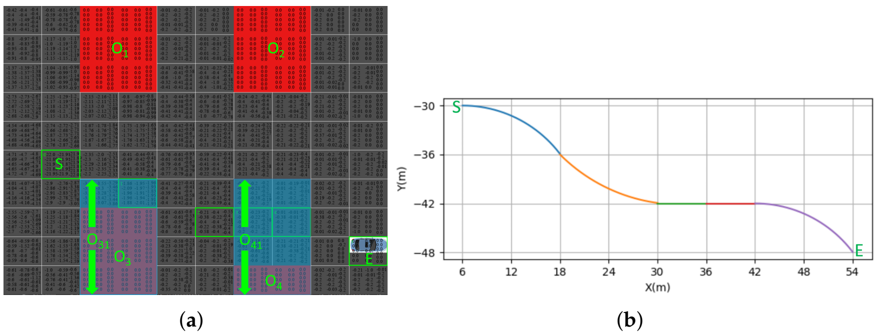


Figure 13. Micro-scale planning and random obstacle (red rectangle) experiment one. (**a**) Grid experiment environments and RLTT method for planning (green rectangle), where the two bottom obstacle shapes randomly change within a certain range. (**b**) TLM motion planning (color curve) according to RLTT planning.

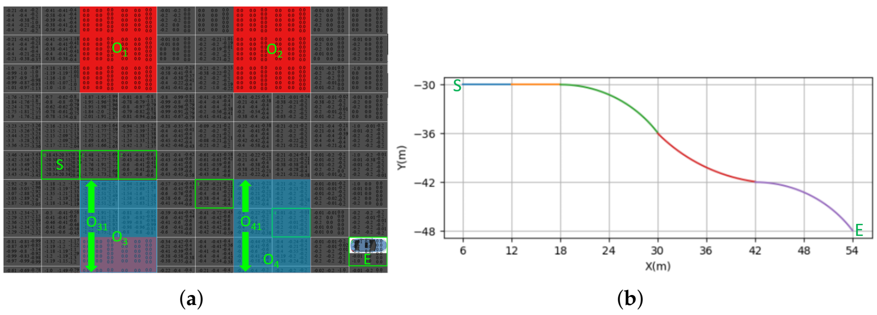


Figure 14. Micro-scale planning and random obstacle experiment two. (**a**) Grid experiment environments and RLTT method for planning, where the two bottom obstacle shapes randomly change within a certain range. (**b**) TLM motion planning according to RLTT planning.

7.3. Comparison Experiments in Certain and Uncertain Environments

Related experiments were conducted in different environments, which were either certain (known environment information) or uncertain environments (unknown environment information), as shown in Figure 15. The environments were the same except for the random obstacles. The average distance of five uncertain experiments was 54.87 (m), and the average distance of five certain experiments was 71.98 (m). The experiments showed that the RLTT method could deal with the uncertain environments well.

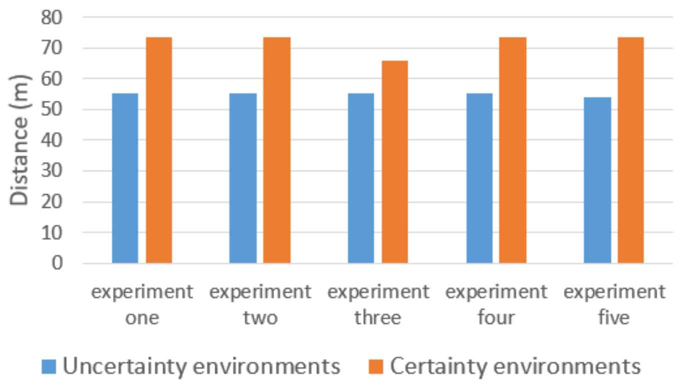


Figure 15. Motion planning for automated driving in uncertain and certain environments.

7.4. Comparison Experiments between RLTT and Q Learning Algorithms

We conducted five experiments using the RLTT method and the Q learning algorithm in unknown environments. The Q learning algorithm is a classic and commonly used algorithm for vehicle navigation. The computation time of each experiment and the averaged computation time of the five experiments using the RLTT method and the Q learning algorithm are shown in Figures 16 and 17, respectively.

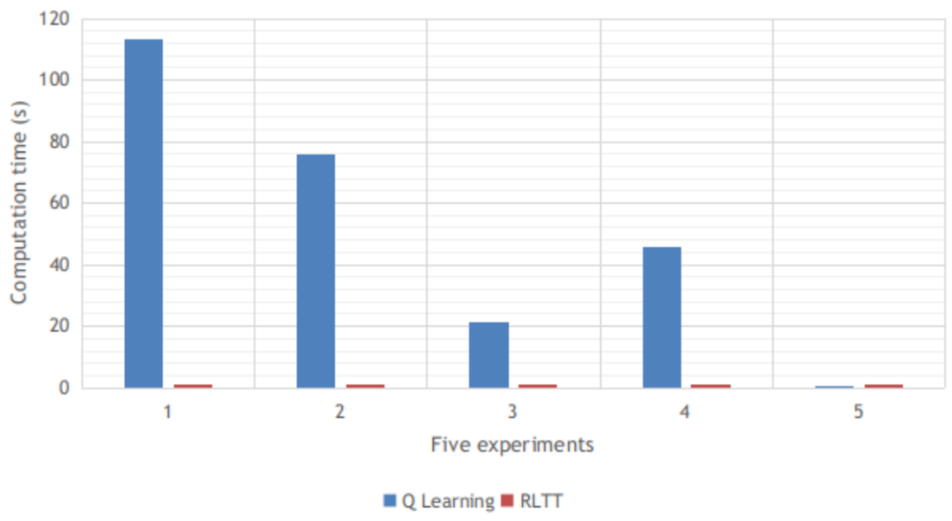


Figure 16. Computation time for each experiment on motion planning using RLTT and the Q learning algorithm.

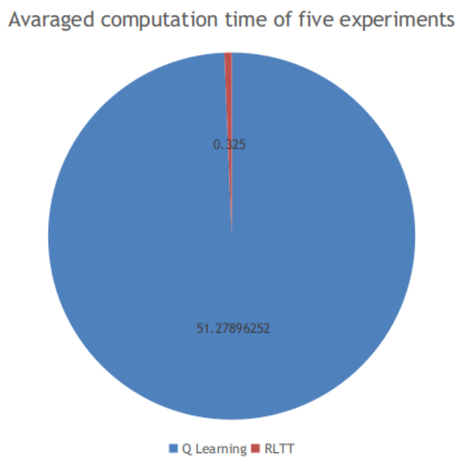


Figure 17. Averaged computation time for five experiments on motion planning using RLTT and the Q learning algorithm.

More specifically, the averaged computation times for the RLTT method and Q learning for motion planning were 0.325 (s) and 52.279 (s), respectively. In addition, the trajectory distance when using Q learning is usually greater than that when the RLTT method is used. We randomly selected one of the five experiments and computed the trajectory distance. The distance when using the RLTT method was 95.834 (m), whereas that when using the Q learning algorithm was 126.735 (m). The results indicate that the performance of the RLTT method was better than the classic RL algorithm, and the RLTT method achieved a shorter trajectory distance and used less time for the path search as compared to the classic RL algorithm.

7.5. Comparison Experiments Between RLTT and Traditional Algorithms

This section shows the experiments comparing the RLTT and traditional algorithms for automated driving motion planning. The adopted traditional algorithm is a hybrid A* algorithm [55], which was proposed by Dolgov et al. It was developed based on the A* algorithm [39] by leveraging continuous coordinates to guarantee the kinematic feasibility of the trajectories and the conjugate gradient descent in order to improve the quality of the trajectories on the geometric workspace. The hybrid A* algorithm has shown better performance than the rapidly exploring random trees algorithms [56,57] and the parameterized curve algorithms [58] in terms of fast global convergence and optimal global values. It is very useful and widely used for automated driving since the hybrid A* algorithm can be smooth and practical for autonomous driving in real-world and unknown environments [55].

The distance of motion planning using the RLTT algorithm was 147.4 (m), which considered dynamic constraints under the environments with sparse information; on the other hand, in the same environments, the distance of motion planning using the hybrid A* algorithm while considering dynamic constraints was 184.0 (m). The experimental results indicate that our proposed method outperforms the hybrid A* algorithm in terms of distance.

7.6. Uncertain Corridor Scenarios

We referred to the experimental settings of reference [59] to carry out an experiment on uncertain corridor scenarios. Figure 18a shows real traffic environments: uncertain corridor scenarios. Figure 18b shows motion planning via RLTT for an automated vehicle in real and uncertain traffic scenarios. The red areas denote uncertain objects, where there is a corridor, some vehicles, and pedestrians crossing the corridor uncertainly. The vehicles on the main road (as indicated by the light blue arrow) can safely cross the uncertain corridor scenarios; the distance of motion planning was 61.8 (m).

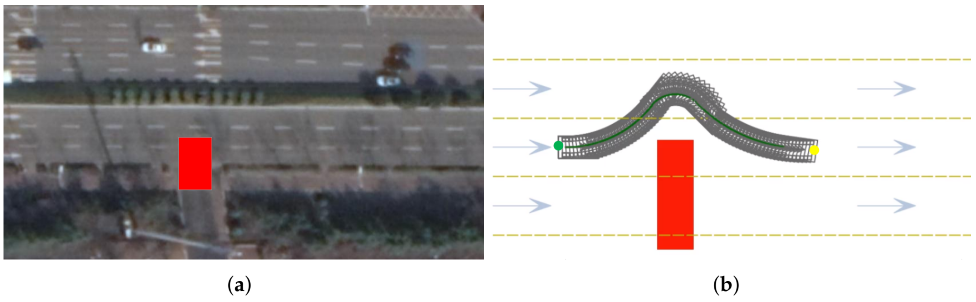


Figure 18. RLTT for traffic scenarios: driving corridors. (**a**) Real traffic scenario: uncertain corridor scenario. (**b**) Motion planning for an automated vehicle in uncertain corridor scenarios.

8. Conclusions

In this study, a constrained RL method along with a theoretical analysis were developed based on the TLM and TPS methods in order to achieve a multi-constraint and multi-scale safe motion planning for automated driving in sparse information environments. The dynamic constraints of a vehicle as well as the smooth constraints, safety constraints, and distance optimization constraints were taken into account. In addition, related experiments were conducted to evaluate the effectiveness of our proposed method. Experimental results indicate that the proposed method is extendable and can be applied to other types of vehicle navigation and control, such as ground robots for parking maneuvers and logistics environments. We hope that our RLTT method inspires new research in robotics development. In the future, we plan to carry out more experiments and aim to improve the proposed method by performing more complex tasks.

Author Contributions: Methodology, S.G.; formal analysis, S.G.; investigation, S.G.; resources, S.G.; data curation, S.G.; writing—original draft preparation, S.G.; writing—review and editing, J.H., Y.H., and G.C.; visualization, S.G.; supervision, G.C. and A.K.; project administration, L.Z. and G.C.; funding acquisition, G.C. and A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was financially supported by the German Research Foundation (DFG) and the Technical University of Munich (TUM) within the framework of the Open Access Publishing Program. This work was partially supported by the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 945539 (Human Brain Project SGA3), and by the National Natural Science Foundation of China (No. 61906138).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets generated during this study are available from the corresponding author upon reasonable request.

Acknowledgments: We would like to thank Jiancheng Long for his very constructive suggestions; we would also like to thank Xiao Wang for the fruitful discussions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ye, Y.; Zhang, X.; Sun, J. Automated vehicle's behavior decision making using deep reinforcement learning and high-fidelity simulation environment. *Transp. Res. Part C Emerg. Technol.* **2019**, *107*, 155–170. [CrossRef]
2. Chen, G.; Chen, K.; Zhang, L.; Zhang, L.; Knoll, A. VCANet: Vanishing-Point-Guided Context-Aware Network for Small Road Object Detection. *Automot. Innov.* **2021**, *4*, 400–412. [CrossRef]
3. Wen, L.; Duan, J.; Li, S.E.; Xu, S.; Peng, H. Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization. In Proceedings of the 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), Rhodes, Greece, 20–23 September 2020; pp. 1–7.
4. Min, H.; Xiong, X.; Wang, P.; Yu, Y. Autonomous driving path planning algorithm based on improved A* algorithm in unstructured environment. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2021**, *235*, 513–526. [CrossRef]

5. Williams, G.; Drews, P.; Goldfain, B.; Rehg, J.M.; Theodorou, E.A. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Trans. Robot.* **2018**, *34*, 1603–1622. [CrossRef]
6. Likmeta, A.; Metelli, A.M.; Tirinzoni, A.; Giol, R.; Restelli, M.; Romano, D. Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving. *Robot. Auton. Syst.* **2020**, *131*, 103568. [CrossRef]
7. Hang, P.; Lv, C.; Huang, C.; Cai, J.; Hu, Z.; Xing, Y. An Integrated Framework of Decision Making and Motion Planning for Autonomous Vehicles Considering Social Behaviors. *IEEE Trans. Veh. Technol.* **2020**, *69*, 14458–14469. [CrossRef]
8. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
9. Gu, S.; Kuba, J.G.; Wen, M.; Chen, R.; Wang, Z.; Tian, Z.; Wang, J.; Knoll, A.; Yang, Y. Multi-agent constrained policy optimisation. *arXiv* **2021**, arXiv:2110.02793.
10. Gu, S.; Yang, L.; Du, Y.; Chen, G.; Walter, F.; Wang, J.; Yang, Y.; Knoll, A. A Review of Safe Reinforcement Learning: Methods, Theory and Applications. *arXiv* **2022**, arXiv:2205.10330.
11. Brunke, L.; Greeff, M.; Hall, A.W.; Yuan, Z.; Zhou, S.; Panerati, J.; Schoellig, A.P. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annu. Rev. Control. Robot. Auton. Syst.* **2021**, *5*, 411–444. [CrossRef]
12. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef]
13. Tamar, A.; Wu, Y.; Thomas, G.; Levine, S.; Abbeel, P. Value iteration networks. In *Advances in Neural Information Processing Systems*; Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2016; Volume 29.
14. Kaelbling, L.P. The foundation of efficient robot learning. *Science* **2020**, *369*, 915–916. [CrossRef]
15. Zhou, C.; Gu, S.; Wen, Y.; Du, Z.; Xiao, C.; Huang, L.; Zhu, M. The review unmanned surface vehicle path planning: Based on multi-modality constraint. *Ocean. Eng.* **2020**, *200*, 107043. [CrossRef]
16. Claussmann, L.; Revilloud, M.; Gruyer, D.; Glaser, S. A review of motion planning for highway autonomous driving. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 1826–1848. [CrossRef]
17. Bernhard, J.; Knoll, A. Robust stochastic bayesian games for behavior space coverage. In Proceedings of the Robotics: Science and Systems (RSS), Workshop on Interaction and Decision-Making in Autonomous-Driving, Virtual Session, 12–13 July 2020.
18. Zhang, H.; Chen, W.; Huang, Z.; Li, M.; Yang, Y.; Zhang, W.; Wang, J. Bi-level actor-critic for multi-agent coordination. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 7325–7332.
19. Harmening, N.; Biloš, M.; Günnemann, S. Deep Representation Learning and Clustering of Traffic Scenarios. *arXiv* **2020**, arXiv:2007.07740.
20. Zhang, L.; Zhang, R.; Wu, T.; Weng, R.; Han, M.; Zhao, Y. Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 5435–5444. [CrossRef]
21. Chen, J.; Li, S.E.; Tomizuka, M. Interpretable End-to-End Urban Autonomous Driving With Latent Deep Reinforcement Learning. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 5068–5078. [CrossRef]
22. Tang, X.; Huang, B.; Liu, T.; Lin, X. Highway Decision-Making and Motion Planning for Autonomous Driving via Soft Actor-Critic. *IEEE Trans. Veh. Technol.* **2022**, *71*, 4706–4717. [CrossRef]
23. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning Research, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
24. Zhu, H.; Han, T.; Alhajyaseen, W.K.; Iryo-Asano, M.; Nakamura, H. Can automated driving prevent crashes with distracted Pedestrians? An exploration of motion planning at unsignalized Mid-block crosswalks. *Accid. Anal. Prev.* **2022**, *173*, 106711. [CrossRef]
25. Achiam, J.; Held, D.; Tamar, A.; Abbeel, P. Constrained policy optimization. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–1 August 2017; pp. 22–31.
26. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv: 1707.06347.
27. Shalev-Shwartz, S.; Shammah, S.; Shashua, A. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv* **2016**, arXiv:1610.03295.
28. Thornton, S. Autonomous Vehicle Speed Control for Safe Navigation of Occluded Pedestrian Crosswalk. *arXiv* **2018**, arXiv: 1802.06314.
29. Codevilla, F.; Miiller, M.; López, A.; Koltun, V.; Dosovitskiy, A. End-to-end driving via conditional imitation learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 1–9.
30. Dai, S.; Schaffert, S.; Jasour, A.; Hofmann, A.; Williams, B. Chance constrained motion planning for high-dimensional robots. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 8805–8811.
31. Thomas, A.; Mastrogiovanni, F.; Baglietto, M. Probabilistic Collision Constraint for Motion Planning in Dynamic Environments. *arXiv* **2021**, arXiv:2104.01659.
32. Mohanan, M.; Salgoankar, A. A survey of robotic motion planning in dynamic environments. *Robot. Auton. Syst.* **2018**, *100*, 171–185. [CrossRef]

33. Webb, D.J.; Van Den Berg, J. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 5054–5061.
34. Gammell, J.D.; Srinivasa, S.S.; Barfoot, T.D. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 2997–3004.
35. Janson, L.; Schmerling, E.; Clark, A.; Pavone, M. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *Int. J. Robot. Res.* **2015**, *34*, 883–921. [CrossRef]
36. Du, Z.; Wen, Y.; Xiao, C.; Zhang, F.; Huang, L.; Zhou, C. Motion planning for unmanned surface vehicle based on trajectory unit. *Ocean. Eng.* **2018**, *151*, 46–56. [CrossRef]
37. Zhu, M.; Xiao, C.; Gu, S.; Du, Z.; Wen, Y. A Circle Grid-based Approach for Obstacle Avoidance Motion Planning of Unmanned Surface Vehicles. *arXiv* **2022**, arXiv:2202.04494.
38. Gu, S.; Zhou, C.; Wen, Y.; Xiao, C.; Knoll, A. Motion Planning for an Unmanned Surface Vehicle with Wind and Current Effects. *J. Mar. Sci. Eng.* **2022**, *10*, 420. [CrossRef]
39. Gu, S.; Zhou, C.; Wen, Y.; Zhong, X.; Zhu, M.; Xiao, C.; Du, Z. A motion planning method for unmanned surface vehicle in restricted waters. *Proc. Inst. Mech. Eng. Part M J. Eng. Marit. Environ.* **2020**, *234*, 332–345. [CrossRef]
40. McNaughton, M.; Urmson, C.; Dolan, J.M.; Lee, J.W. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 4889–4895.
41. Rajamani, R. *Vehicle Dynamics and Control*; Springer: New York, NY, USA, 2011.
42. Gong, J.; Jiang, Y.; Xu, W. *Model Predictive Control For Self-Driving Vehicles*; Beijing Institute of Technology Press: Beijing, China, 2014.
43. Gardner, R. The brunn-minkowski inequality. *Bull. Am. Math. Soc.* **2002**, *39*, 355–405. [CrossRef]
44. Meurant, G. *Handbook of Convex Geometry*; Elsevier: Amsterdam, The Netherlands, 2014.
45. Vershynin, R. *High-Dimensional Probability: An Introduction with Applications in Data Science*; Cambridge University Press: Cambridge, UK, 2018; Volume 47.
46. Zhou, C.; Gu, S.; Wen, Y.; Du, Z.; Xiao, C.; Huang, L.; Zhu, M. Motion planning for an unmanned surface vehicle based on topological position maps. *Ocean. Eng.* **2020**, *198*, 106798. [CrossRef]
47. Gu, S.; Zhou, C.; Wen, Y.; Xiao, C.; Du, Z.; Huang, L. Path Search of Unmanned Surface Vehicle Based on Topological Location. *Navig. China* **2019**, *42*, 52–58.
48. Herring, J.R. *OpenGIS Implementation Specification for Geographic Information-Simple Feature Access—Part 1: Common Architecture*; Open Geospatial Consortium: Wayland, MA, USA, 2006; p. 95.
49. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
50. Jin, C.; Allen-Zhu, Z.; Bubeck, S.; Jordan, M.I. Is Q-learning provably efficient? In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 4868–4878.
51. Latif, A. Banach contraction principle and its generalizations. In *Topics in Fixed Point Theory*; Springer: Cham, Switzerland, 2014; pp. 33–64.
52. Melo, F.S. *Convergence of Q-Learning: A Simple Proof*; Tech. Rep.; Institute of Systems and Robotics: Lisbon, Portugal, 2001.
53. Greenwald, A.; Hall, K. Correlated-Q learning. In Proceedings of the Twentieth International Conference on International Conference on Machine Learning, Washington, DC, USA, 21–24 August 2003; pp. 242–249.
54. Zhao, D.; Wang, H.; Shao, K.; Zhu, Y. Deep reinforcement learning with experience replay based on SARSA. In Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, Greece, 6–9 December 2016; pp. 1–6.
55. Dolgov, D.; Thrun, S.; Montemerlo, M.; Diebel, J. Practical search techniques in path planning for autonomous driving. *Ann. Arbor* **2008**, *1001*, 18–80.
56. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
57. Plaku, E.; Kavraki, L.E.; Vardi, M.Y. Discrete search leading continuous exploration for kinodynamic motion planning. In *Robotics: Science and Systems III*; MIT Press: Cambridge, MA, USA, 2007; pp. 326–333.
58. Cremean, L.B.; Foote, T.B.; Gillula, J.H.; Hines, G.H.; Kogan, D.; Kriechbaum, K.L.; Lamb, J.C.; Leibs, J.; Lindzey, L.; Rasmussen, C.E.; et al. Alice: An information-rich autonomous vehicle for high-speed desert navigation. *J. Field Robot.* **2006**, *23*, 777–810. [CrossRef]
59. Wang, C.; Li, F.; Wang, Y.; Wagner, J.R. Haptic Assistive Control with Learning-Based Driver Intent Recognition for Semi-Autonomous Vehicles. *IEEE Trans. Intell. Veh.* **2021**. [CrossRef]