RLTT: Multi-Constraint and Multi-Scale Motion Planning for Autonomous Driving with Constrained Reinforcement Learning

Journal:	*IEEE Transactions on Vehicular Technology*
Manuscript ID	VT-2021-01924
Suggested Category:	Regular Paper
Date Submitted by the Author:	04-Jun-2021
Complete List of Authors:	Gu, Shangding; Technical University of Munich, Informatics Chen, Guang; Tongji University; Technical University of Munich Zhang, LIjun; Tongji University Hou, Jing; Tongji University Hu, Yingbai; Technical University of Munich Knoll, Alois; Technical University of Munich
Keywords:	Motion Planning, Autonomous Driving, Constrained Reinforcement Learning, Multiple Constraints

RLTT: Multi-Constraint and Multi-Scale Motion Planning for Autonomous Driving with Constrained Reinforcement Learning

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

***Abstract*—Mostly, the rule-based traditional motion planning methods have superior performance with prior knowledge of the macro-scale environment while suffering the problem under an unknown and uncertain environment. To address this issue, deep reinforcement learning (DRL) is a solution that can deal with micro-scale unknown and uncertain environments well. Nevertheless, it is unstable and lacks interpretability. Therefore, it raises a new challenge: how to combine the effectiveness and overcome the drawbacks of the two methods guaranteeing stability under uncertain environments. In this paper, a multi-constraint and multi-scale motion planning method for autonomous driving with constrained reinforcement learning is proposed (named RLTT), which is based on reinforcement learning (RL), topological path search (TPS), and trajectory lane model (TLM). Firstly, a dynamic model of vehicles is formulated, and then TLM is developed based on the dynamic model, which constrains RL action and state space. Secondly, the macro-scale path planning is solved through TPS, and in the micro-scale range, the discrete routing points are achieved via RLTT. Thirdly, the motion planning method is designed by combining our proposed sophisticated rules. Finally, the related experiments are conducted to evaluate the efficiency of the proposed method, the results of experiments indicate that the method can help reduce the gap between the data-driven methods and traditional methods, provide better performance for autonomous driving, and facilitate the application of RL methods for more fields.**

***Index Terms*—Motion Planning, Autonomous Driving, Constrained Reinforcement Learning, Multiple Constraints**

I. Introduction

AUTONOMOUS driving technology is increasing in industrial demand which has shown impressive progress provided a promising transportation in our society [1]–[5]. However, the vehicle motion planning for autonomous driving still needs to be further developed considering multi constraints in sparse information environments, where the macro information is known and the micro information is unknown and uncertain, especially for safe, effective and precise motion planning. As for autonomous driving, the real traffic environment is full of uncertainty because of the information incompleteness and traffic disturbances, etc., which result in complex traffic environments that are hard to be predicted real time.

There are various of traditional motion planning methods (non-data-driven methods) that use heuristic algorithms [6], sampling-based methods [7], and rule-based methods [8], [9] for autonomous driving. Although these methods achieved impressive performance via solid math models regarding robustness, interpretability and stability, they need lots of human knowledge in advance, and thereby are hard to model the complex and uncertain environments, and might not conduct well under unknown and uncertain environments. On the other hand, reinforcement learning (RL) (data-driven methods) [10] does not need lots of human knowledge in advance and maybe better conduct tasks in unknown and uncertain environments [11] via trial-and-error learning, and can model complex environments through data-driven methods.

Specifically, agents using RL methods can learn how to navigate a vehicle automatically via exploration [12], [13]. Therefore, it is necessary to combine RL and traditional methods to achieve better performance in uncertain environments for vehicle motion planning.

How to combine traditional methods with RL to achieve safe, effective, and precise motion planning leveraging the two kinds of methods' advantages, and overcoming the shortcomings is a challenging problem. In this paper, RL is constrained with traditional methods such as topological path search (TPS) and trajectory lane model (TLM) to achieve safe, effective and precise motion planning (named RLTT). To the best of authors knowledge, this is the first time that the motion planning for autonomous driving based on RL, TPS and TLM considering multi-constraint and multi-scale motion planning in sparse information environments.

There are several key challenge points that need to be solved in this paper: firstly, the routing points need to be considered into a dynamic model of vehicles, which can render motion planning more reasonable and closer to the actual environments; secondly, how to transform between RL and topological path needs to be resolved, because RL is a trial-and-error algorithm which is hard to search paths for large-scale areas in term of the time consumption. On the other hand, the method of TPS can easily and effectively plan a path for the large-scale areas; thirdly, how to build a dynamic model for autonomous driving and provide dynamic constraints to render motion planning safer and more effective, precise.

Considering the above problems, firstly, the planning hier-

This work was supported by the China Scholarship Council under Grant 202006950010; the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 945539 (Human Brain Project SGA3); the National Natural Science Foundation of China (No. 61906138). (*Corresponding author: Guang Chen)*

Authors Affiliation: [1]Tongji University, Shanghai, China; [2]Chair of Robotics, Artificial Intelligence and Real-time Systems, Technical University of Munich, Munich, Germany.

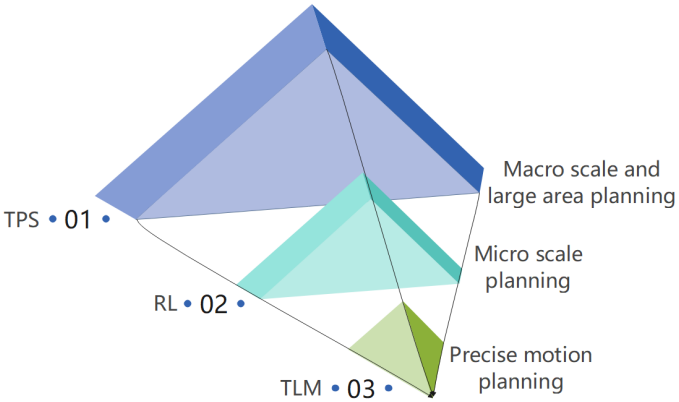

Fig. 1: The framework of RLTT method: planning hierarchy framework of three levels for motion planning.

archy framework of three levels is proposed shown in Fig. 1, in which the first level is for large scale planning via the TPS; the second level is the RL method that is used for micro-scale planning; the third level is TLM which can render motion planning more precise, closer actual environments and has better performance. Secondly, aiming at the problem associate with interpretable and the unstable issues of deep learning (DL), it is hard to be solved, because we know DL can do what, but we can't explain the reason clearly. Thus, we try to develop TPS and TLM for replacing DL from the perspective of traditional methods, where the motion planning can be of interpretability and stability, their own advantages are combined and their own shortcomings are overcome. Thirdly, for uncertain environments, such as the uncertain obstacle, RL is developed to explore the dynamical and uncertain environments; moreover, the multi constraints are taken into account and the safety of range is set through a safe buffer of TPS.

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

- A novel sophisticated RLTT method is proposed for autonomous driving, where the novel planning hierarchy framework of three levels and multi-scale planning based on TPS, RL and TLM are introduced.
- The multiple constraints of vehicles are taken into account, such as the dynamic constraints of vehicles, smooth constraints, safety constraints, etc., which can render motion planning more reasonable and closer to the actual situation.
- The uncertain environments are also considered in proposed planning method which achieves superior performance than related work.
- The interpretable, safe and efficient motion planning for autonomous driving is achieved; and the RLTT method via the combination of traditional methods and RL method can work well under sparse information environments and multi-scale planning environments.

The remainder of this paper is organized as follows: the related work is introduced in Section II; the dynamic model of vehicles is provided in Section III; the TLM is present in Section IV; the method of TPS is introduced in Section V; RLTT method for autonomous driving is introduced in detail in Section VI; the related experiments are conducted in Section VII; conclusion of the paper is introduced in Section VIII.

II. Related Work

The navigation for unmanned vehicles has attracted the attention of many researchers [2], [14]–[17], especially the motion planning in sparse information environments which are unknown and uncertain environments.

For example, in [18], Bernhard and Knoll consider the the uncertain information using neural networks for autonomous driving navigation under sparse information environments. Nevertheless, they assume all information of other vehicles is known and the assumption might not be suitable in real environments. Zhang *et al.* [19] develop a novel bi-level actor-critic method for multi-agent coordination, the method has achieved great success in making decisions on autonomous driving in highway merge environments. Nonetheless, the method can not guarantee the safety of autonomous vehicles in some environments. Nick *et al.* [20] introduce an algorithm for clustering of traffic scenarios, in which they combine Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to predict the traffic scenarios for the ego vehicle making decision and navigation. Nonetheless, the method might be unstable for some extreme traffic scenarios due to the uncertain and imperfect information of traffics, such as intersection environments. Chen *et al.* [1] develop an end-to-end method for autonomous driving using RL and a sequential latent environment model which is a semantic bird-eye mask, and the method is interpretable compared with other machine learning methods to some extent. However, the method may need to further take into account the sparse information environments and multi constraints for autonomous driving.

Shai *et al.* [21] present a sophisticated strategy that can consider the negotiation strategies when the ego vehicle drives in complex environments. The method can be divided into two strategies: one strategy can be learned and another strategy can not be learned and is hard constraints. Although they have tried the best to prompt the safety for autonomous driving, the method may need to be improved adaptability for more complex environments, such as the complex intersection considering multi heterogeneity vehicles and pedestrians. Sarah M. Thornton [22] proposes a method leveraging the Markov decision process (MDP) and dynamic programming (DP) to control the vehicle speed for safety, in which the method considers the uncertain pedestrian in a crosswalk. However, the method might need to be further developed to consider more uncertain environments and the search space may need to be reduced for efficient navigation. Codevilla1 *et al.* [23] utilize condition imitation learning for high-level command input to achieve autonomous driving in simulation environments and on a 1/5 scale robotic truck in real environments. Although both experiments have evaluated the effectiveness of the proposed method, the method may need to address human guidance of autonomous vehicles for sparse information environments.

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

In RLTT, TLM is developed based on trajectory unit, which is firstly proposed for unmanned surface vehicles (USVs) [24]. However, USVs are different from autonomous vehicles: the free degrees of control the navigation environments and vehicles' shape are different [25]; the trajectory unit may not be suitable for autonomous, so it is necessary to propose TLM for autonomous vehicles; TLM is also different from lattice planner [26], because lattice planner is achieved via sample and fit data, which may spend lots of time and require more computing power, but TLM is achieved via kinetics and dynamics model of vehicles. Additionally, TPS is proposed to constraint RL search space and provide routing points for RL navigation, which can prompt RL efficiency. Finally, the hierarchy framework is proposed integrating TPS, RL and TLM to develop RLTT method, which can render the proposed method and algorithm more unified.

III. Problem Formulation

In this paper, the constrained multi-objective problem is considered for motion planning, specifically, uncertain constraints F_{un}, dynamic constraints F_{dy}, safety constraints F_{sa} and smooth constraints F_{sm} are considered.

A. Uncertain constraints F_{un}

For the convenience of description, we have following definitions shown (1): $O_{obstacle}$ denotes the obstacles; O_{shape} denotes the shape of obstacles; $O_{position}$ denotes the position of obstacles; O_{part} denotes the part information of obstacle position and shape. In uncertain environments, the information of the obstacle shape can not be fully observed, $O_{observe}$ denotes the partial information of obstacles.

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

B. Dynamic constraints F_{dy}

Dynamic constraints F_{dy} can be represented as the dynamic model of vehicles. In this paper, the kinematics model, steer command, heading angle are been considered. The kinematics model is briefly introduced in this paper, and the details can refer to the references [27], [28]. Generally, the steer command range is set around $30^\circ \sim 40^\circ$, and the setting of the steering command of our method is suitable and can be applied for 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 deflection 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 denotes the distance between the rear axle and front axle (wheelbase), w represents the yaw rate of the velocity, R denotes the turning radius of the vehicle, the geometric relationship between the front and rear wheels is as follows:

$$\begin{cases} X_f = X_r + lcos\varphi \\ Y_f = Y_r + lsin\varphi \end{cases} \tag{4}$$

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

According to the above analysis, the kinematics can be briefly summarized in equation (6):

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

C. 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. the safety constraints (F_{sa}) are represented as the equation (7):

$$D_{safety} <= D_p \tag{7}$$

D. Smooth constraints F_{sm}

The motion planning points is the set P, $\sum_{n=1}^{N} P_n \subseteq P$. Two adjacent points are P_1 and P_2 respectively. The steer angle of any two adjacent points is δ_{normal}. The limitation steer angle of any two adjacent points is $\delta_{limitation}$, where $\delta_{limitation}$ is set for to smooth the steer angle. The smooth constraints F_{sm} can be represented as the equation (8):

$$\delta_{normal} <= \delta_{limitation} \tag{8}$$

IV. Trajectory Lane Model

Trajectory lane model (TLM) can be seen as a bridge to connect the routing planning with dynamic constraints of a vehicle, and render the motion planning to be closer to the actual environments shown in **Algorithm 1**. TLM is constructed according to the dynamic model of the vehicle, and the relative TLM rules are introduced to help achieve effective and suitable TLM for effective and precise motion planning.

A. Rules Design

Rule one: the trajectory of each action has an equal length. This is for trajectory continuous, regularization and easy to splice trajectory.

Rule two: each trajectory has only one steer command except for the start and end steer command. this is for smooth steer, and the relative angle between two adjacent points is no more than one degree, this is for generating a smooth trajectory.

Rule three: the trajectory of each action has the same speed, that is for the equal length; at the start and end of the trajectory, the condition is the same.

B. Analysis of TLM

According to the above rules and dynamic model of vehicle, TLM can be constructed. The introduction of the construction of the TLM m can be found in algorithm 1, in which there are several kinds of TLM according to constraints of the action space.

Algorithm 1 Trajectory lane model m.

Input: The acceleration a, velocity v, position (x, y), front wheel deflection angle δ, differential time dt, interval time T, heading angle ψ, wheelbase$frlen$;

Output: trajectory lane model m;

1: Initial Q value (s, a), $\forall\ s \in S$, $a \in A$, set parameter α, γ, i = 0;
2: Repeat from step 3 to step 7 for i = 0 to interval time T;
3: $x' \leftarrow x + v * cos(\psi)dt$;
4: $y' \leftarrow y + v * sin(\psi)dt$;
5: $\psi' \leftarrow \psi + (v/frlen) * \delta * dt$;
6: $v' \leftarrow v + a * dt$;
7: $x = x', y = y', \psi = \psi', v = v'$;
8: **return** trajectory lane model $m \leftarrow (x, y, v, \psi, a, \delta)$;

1) TLM' type one: it is the simplest type, which only has four direction actions, and it can be seen in Fig. 2(a). The kind of TLM does not have enough actions to achieve motion planning, and usually the kind of TLM for motion planning is locally optimal.

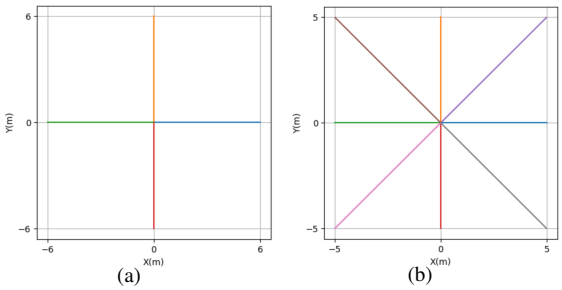

Fig. 2: Straight line trajectory lane model. (a) is the straight line trajectory in four actions directions. (b) is the straight line trajectory in eight actions directions.

2) TLM' type two: this kind of TLM has more actions than the first type, which has eight direction actions, and it can be seen in Fig. 2(b). The kind of TLM still does not have enough actions to achieve motion planning, and usually the kind of TLM for motion planning is also local optimal.

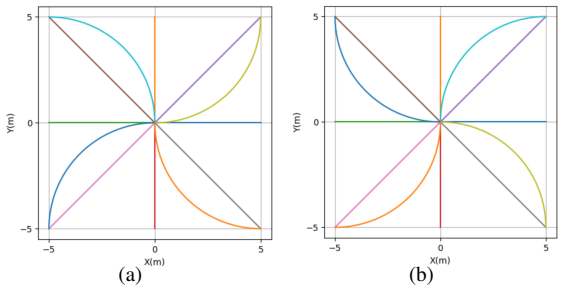

Fig. 3: Semicircle trajectory lane model. (a) is the positive semicircles in 12 directions. (b) is the negative semicircles in 12 directions based on eight actions.

3) TLM' type three: this kind of TLM has more actions than the second type, which has 16 direction actions (as shown in Fig. 4(a)) and can be consisted by the second type including 4 direction-action positive semicircles (in Fig. 3(a)) and 4 direction-action negative semicircles (in Fig. 3(b)). The kind of TLM still does not have enough actions to achieve motion planning, and sometimes the kind of TLM for motion planning is also local optimal.

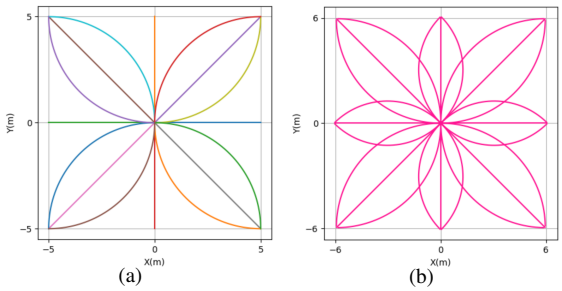

Fig. 4: Semicircle trajectory lane model. (a) is the 16 direction-action TLM. (b) is the 24 direction-action TLM.

4) TLM' type four: this kind of TLM has more actions than the third type, which has 24 direction actions and can be consisted by the third type and 8 direction-action trajectories near-maximum rudder angle along X and Y axes, and it can be seen in Fig. 4(b). The kind of TLM is same as the third type of TLM, and sometimes the kind of TLM for motion planning is also local optimal.

5) TLM' type five: this kind of TLM has more actions than the fourth type, which has 40 direction actions and can be consisted by the third, 8 direction-action smooth semicircles (Fig. 5(a)), and it can be seen in Fig. 5(b). To some extent, the kind of TLM has enough actions to achieve motion planning, and the kind of TLM for motion planning can achieve optimal motion planning. Fig. 6 shows the 40 direction-action TLM with one example of the smooth circles.

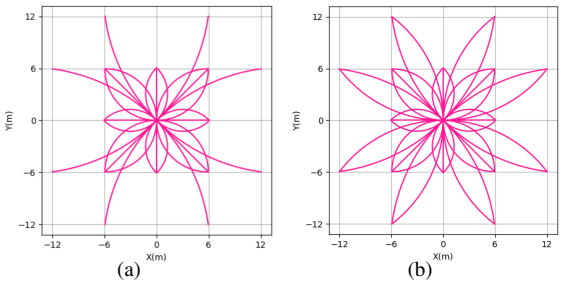

Fig. 5: Semicircle trajectory lane model. (a) is the 32 direction-action TLM. (b) is the 40 direction-action TLM.

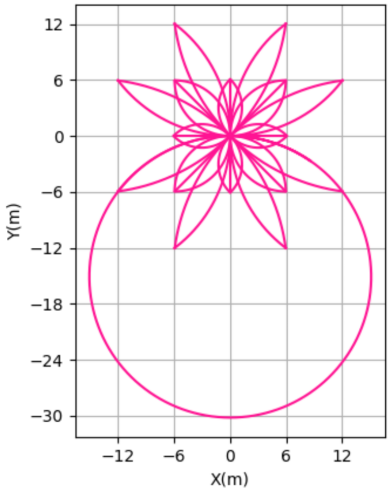

Fig. 6: 40 direction-action TLM and show one example of the smooth circles.

V. Topological Path Search for Autonomous Driving

For large-scale path search and efficient path search, the topological map may need to be built for path search because of the high efficiency of searching [29]. During the path searching process, OGC (open geospatial Consortium) is used to make a topological map [30]. OGC defines a simple feature model which is suitable for efficient storing and accessing geographic features in relational databases.

In the spatial map, the map is consisted of node, line and surface, which are three elements for constructing map: node is a point, which does not have spatial features such as shape and size; line is composed of a series of nodes, whose spatial features include shape, length, etc.; surface is composed of closed ring, whose spatial features include shape, area, etc. In TPS, node denotes the vehicle, line denotes the routing path, 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 relationship, metric relationship and direction relationship. Many topological relation expression models have been proposed to represent the spatial relationship. In this paper, the nine-cross model [30] is used to represent the spatial relationship for constructing a topological map, which is introduced through function (9), where A and B represent two different objects respectively, $(\eth)$ denotes the edge of an object, $(^\circ)$ denotes the internal of an object, $(^-)$ denotes the external of an object, $(\bigcap)$ denotes the intersection of two objects.

Based on Function (9), the topological relation predicates are defined, which are Crosses, Disjoint, Within, Contains, Equals, Touches, Intersects and Overlaps, respectively. In this paper, we use Intersects, Disjoint, Touches and Contains to analyze the topological relationship shown in Fig 7. 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 nine-cross model. For two objects a and b in space, the intersection value is (ϕ) denoted as F (False), value is $-(\phi)$ denoted as T (Truth), and the intersection is represented by 0 when the point is the intersection; with 1 when the line is the intersection; with 2 when the area is the intersection. $*$ means that f, 0, 1 or 2 can be selected. Specifically, a. Disjoint (b): two geometric bodies (a and b) have no common points, forming a set of disconnected geometric shapes.

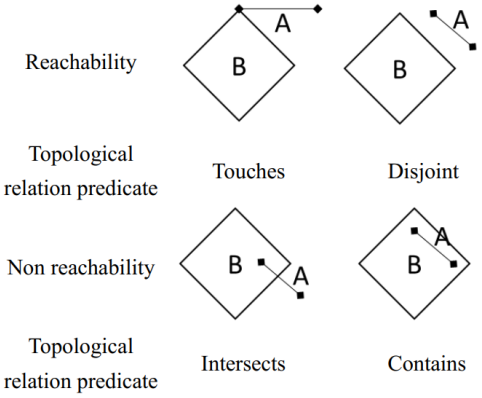

Fig. 7: Schematic diagram of topological path reachability.

After having the topological map, the Dijkstra algorithm [31], [32] is used to achieve the optimal routing planning, and the euclidean metric is utilized to judge the distance between two points, which is provided in function (10). The algorithm is described in **Algorithm 2**, in which the function (9) is used to judge the topological relationship between obstacles O_n, the start point S and the end point E; in addition, the safety buffer is set via topological relationships; finally, the routing path can be achieved by Dijsktra algorithm using function (10) based on topological map, Dijskra algorithm can achieve more optimal path compared with other heuristic algorithm, e.g. A* algorithm, although it spends more computing time. but in the paper, Dijskra algorithm using TPS is very quick to search optimal path and satisfy motion planning.

$$\begin{pmatrix} A^\circ \bigcap B^\circ & A^\circ \bigcap \eth B & A^\circ \bigcap B^- \\ \eth A \bigcap B^\circ & \eth A \bigcap \eth B & \eth A \bigcap B^- \\ A^- \bigcap B^\circ & A^- \bigcap \eth B & A^- \bigcap B^- \end{pmatrix} \tag{9}$$

$$d(u,v) = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2} \tag{10}$$

Additionally, the safety buffer is added to the topological path search. According to the size of the vehicle maneuver, the size of the safety buffer is set as 6 m, which means the vertical distance is always $6m$ between path and obstacles, and thus render the route planning safe and suitable for motion planning.

Algorithm 2 Method of TPS for autonomous driving.

Input: The set of obstacles, O_n; the start point S and the end point E;
Output: Macro scale routing point, P_n;
1: Repeat from step 2 to step 3 for i = 0 to the number of all obstacles;
2: Repeat step 3 for j = 0 to the number of one obstacle of all points;
3: Judge the topological position relationship between the path point and each point O_{ij} of the first obstacle O_i and utilize function (9);
4: Set the safety buffer and store the topological map;
5: Call Dijkstra algorithm and utilize function (10) for topological path search P_n;
6: **return** P_n;

VI. Motion Planning Method for Autonomous Driving

According to the above analysis of sections, the TLM and TPS have been constructed. In this section, how to achieve motion planning for autonomous driving is needed to be discussed. Firstly, the RL is introduced briefly, and the detailed introduction can refer to the reference [10]; secondly, the RLTT method for autonomous driving is presented in detail. The multi constraints which are considered in the RLTT method can be seen in Fig. 8.

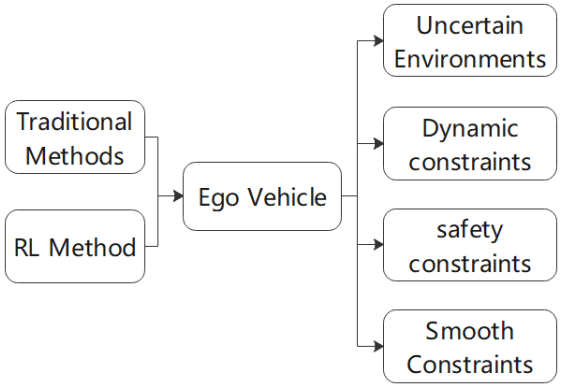

Fig. 8: Traditional methods and RL method for autonomous driving considering different constraints

TPS is introduced for large and macro-scale planning, where the safety and optimal path planning are both achieved via safety buffer, and topological map and Dijkstra; the representation of TPS constraints can be seen in Fig. 9(a). For micro-scale planning, the environment is full of uncertain and imperfect information, it is easier to build a model and learn experience via RL in a data-driven method compared with traditional methods; in contrast, when the environment is of uncertainty and the model is difficult to model via traditional methods. Fig. 9(b) shows the representation of the constraints through RL. TLM can be used for dynamic constraints and smooth constraints, in which the kinematics model of vehicles and steer constraints and the relative angle between any two adjacent points is also limited, the representation of TLM can be seen in Fig. 9(c).

It is useful to consider this kind of framework and this kind of environments, because we can not get the perfect environment information in a real environment, even if the sensors are very developed. The proposed method might help autonomous driving further develop.

A. RL Method

RL can be usually seen as a Markov Decision Process (MDP), and MDP is presented by five-tuple (S, A, P, R, γ), in which S denotes the state space, A denotes the action space, P shows the probability of transition, R represents the reward once one action is conducted, and γ denotes the discounted factor of the reward. In this paper, RL is regarded as Partial Observation Markov Decision Process (POMDP), where RL renders agent interact with environments through trial-and-error learning to get more data about the environments, interact more and then the agent can know environments more and the agent can do better. Specifically, the RL method can deal with uncertain problems better compared with traditional methods. In particular, the most likely action can be selected in the future environment according to the exploration. It can reduce the uncertainty about the environment, which means that more knowledge about the environment can be known, and we can gain more certain information for future action selection.

B. Motion Planning Method

There are lots of advantage RL methods that have been proposed, such as DQN [33], DDPG [34], A2C [35], PPO [36], etc. The methods utilize the strong nonlinear fitting ability of the neural networks to learn actions and states. However, the neural network is criticized because of its problem of interpretability and unstable for autonomous driving. In this work, we try to develop the traditional method that replaces neural networks, and can also work well integrating classic RL method (Q Learning) compared with the related methods.

Firstly, the TLM and TPS are introduced into the framework, as well as Q learning is developed based on the TLM and TPS and provides the constrained reinforcement leaning for safe motion planning. Secondly, the transition function of Q learning is integrated into TLM and TPS, which can be seen in **Algorithm 3**. Secondly, the algorithm of Q learning is presented for path value, which can be seen in **Algorithm 4**. Thirdly, according to the distance of the safety buffer, select the routing points, and call **Algorithm 5** for autonomous driving motion planning.

C. Motion Planning Example

A motion planning example for autonomous driving via RLTT is illustrated in this section. Fig. 10 which shows

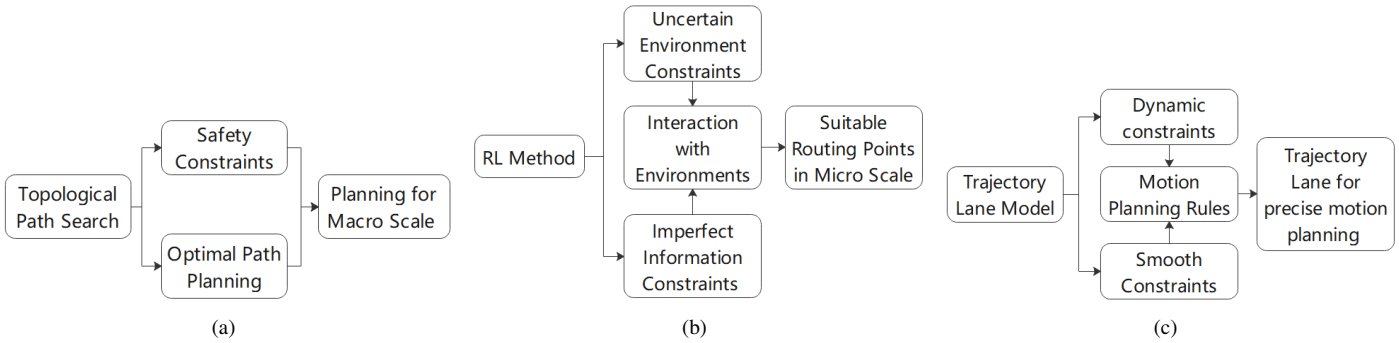

Fig. 9: TPS, RL and TLM for autonomous driving. (a) representation of TPS constraints. (b) representation of RL constraints. (c) representation of TLM constraints.

Algorithm 3 transition function f.

Input: action a, current position P_c and angle A;
Output: next position, reward, done, next angle;

Initialize action a, current position P_c and angle A;
If the current position P_c has collided the obstacle $O_{obstacle}$, return state s, reward r, done and angle A;
If the current position P_c has reached the terminal P_T, return state s, reward r, done and angle A ;
If angle A and action a are equal angle A_t and action a_t of trajectory respectively;
Next position $P_n \leftarrow$ current position P_c, next angle $A_n \leftarrow$ angle A and get related reward r;
If next position P_n has collided the obstacle $O_{obstacle}$, return state s, reward r, done T and angle A;
If next position P_n has reached the terminal P_T, return state s, reward r, done T and angle A;
return next position P_n, reward r, done T, next angle A_n;

Algorithm 4 Q learning method for path value.

Input: The start point S_{ij} and end point E_{in} each segment (j=1,2,...n);
Output: The $Q(s,a)$ and $\pi(s)$ during the segment;

Initial Q value (s, a), $\forall\ s \in S$, $a \in A$, set parameter α, γ, ;
Repeat from step 3 to step 8 until all $Q(s,a)$ converge;
Select the action a according to the initial state s and ϵ-greedy strategy;
Repeat from step 5 to step 8 until s is the terminal;
Conduct a, then get reward r, next angle ψ' and next state s' according to ϵ-greedy strategy, transition strategy f, angle ψ and state s, and record S';
If next state $s' \in S'$, get reward r;
$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max Q(s',a') - Q(s,a)]$;
$s = s'$, $a = a'$;
Get action strategy $\pi(s) \leftarrow \arg\max_a Q(s,a)$;
return $Q(s,a)$ and $\pi(s)$;

a certain and perfect information environments for robots without considering dynamic constraints and with considering dynamic constraints, respectively. 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, etc. The problem in Fig. 10 may be easy to solve using the traditional methods for vehicle motion planning, such as the sampled-based method.

If the uncertain of environments and vehicle constraints are both considered, the traditional method may not work well, for example, in Fig. 11(a), when the environment is 65% certain and 35% uncertain, specifically, the position and shape of the 65% obstacle are certain, the position and shape of the 35% obstacle are uncertain, when the vehicle is navigating, the 35% obstacles are changing randomly. The RLTT method can work well in the environments, due to utilizing RLTT method to explore the unknown world in micro-scale motion planning (blue box (from the S point to M point) and purple box (from M point to E point)) efficiently and rapidly; specifically, leveraging TPS for large and macro-scale routing planning (red box (represented as the black arrow from S point to E point)), and using TLM for micro and precise motion planning (deep pink line), the example can be seen in Fig. 11(b), the motion planning trajectory is represented as the green arrow from the S point to the E point.

VII. Experiments

In the experiments, several experiments have been conducted. Firstly, different RL methods for path search have been conducted. Then, the RLTT method for motion planning has been achieved for autonomous driving. Thirdly, comparison experiments in certain and uncertain environments have been carried out. In addition, comparison experiments between RLTT and RL algorithms are conducted. Finally, comparison experiments between RLTT and traditional algorithms are conducted to evaluate the effectiveness of our proposed methods.

A. Different RL Methods for Path Search

The related RL methods to conduct path planning for autonomous driving have been done, which can be seen in Fig. 12, where dynamic programming (DP) value iteration and DP policy iteration [10] have achieved fewer steps,

Algorithm 5 Motion planning for autonomous driving via RLTT.

Input: The start point $S_{ij}$and end point E_{in} each segment (j=1,2,...n), vehicle environments(trajectory transitions) via **Algorithm 1** and **Algorithm 2**;

Output: The command steer of each point δ; the position of each point during the segment (X_{ij},Y_{ij}); the heading angle of each point during the segment;

1: Initialize the initial state s, heading angle ψ, α, ϵ, call q learning algorithm and return q value via **Algorithm 3** and **Algorithm 4**;
2: Repeat from step 3 to step 6 until t get done or the step number is more than the certain number steps;
3: Append state s into the path, the path of vehicle environment $\leftarrow$ path, and select the action a according to the initial state s and greedy strategy;
4: Add transition state s into a specific position and append position, action, angle into motion planning table $f1$;
5: Conduct a, then get reward r, next angle ψ' and next state s' according to transition strategy f, angle ψ and state s;
6: Repeat step 7 for i = 0 to the length of motion planning steps; and call trajectory lane model m
7: If angle and action of motion planning $f1$ are equal to the angle and action of trajectory m, record the trajectory m;
8: **return** Continuous motion planning trajectory m;

which means the two algorithms can get the shortest path for autonomous driving. Nonetheless, DP value iteration and DP policy iteration may not be suitable for autonomous driving in uncertain and unknown environments because the probability and reward transition might not be known, and the above two methods usually need to know the model of environments.

In addition, Monte Carlo (MC) RL method [10] has achieved shorter steps than Q learning [37] for autonomous driving. However, MC method sometimes falls into a deadlock during exploration and can not get the path. Sarsa method [38] gets more steps than Q learning because of its conservative strategy. Since RLTT is a multi-scale method and the safety buffer constraints of the first level via TPS have been further considered, thus it is better to develop Q learning and integrate it into the RLTT framework for autonomous driving. Next section, the experiments of the RLTT method for autonomous driving will be introduced and discussed.

B. RLTT Method for Motion Planning

1) macro scale motion planning: In this section, macro-scale planning (generally, the planning area can be set as around from 100 m to 2000 m) has been conducted via TPS and can be seen in Fig. 13(a) and Fig. 13(b) which represent the convex obstacle and concave-convex obstacle environments respectively, in which the macro scale planning is denoted as deep pink line and the obstacles are denoted as the red rectangles. The start point and end point are S and S_3 in Fig. 13(a) and S' and S'_3 in Fig. 13(b) respectively. The safety buffer is set as 8 m, as shown in Fig. 13(a) and Fig. 13(a) (denoted as B and yellow arrow).

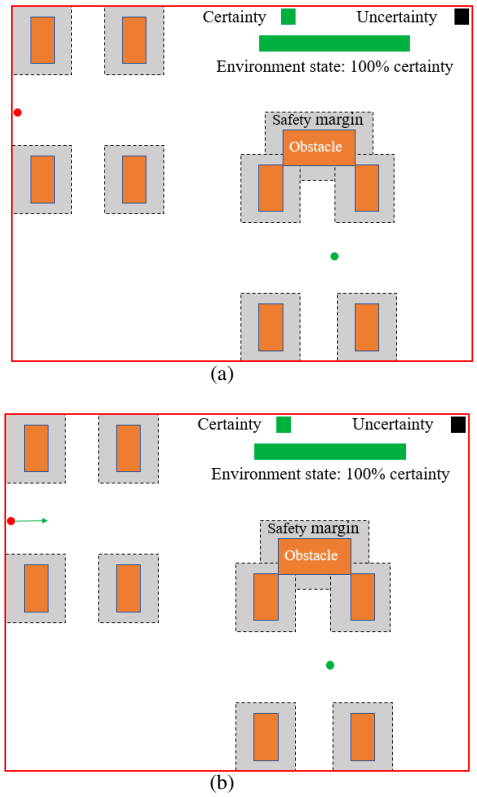

Fig. 10: Certain and perfect information environments for robots. (a) is without considering dynamic constraints. (b) is with considering dynamic constraints.

2) micro scale motion planning: The micro-scale motion planning (generally, the planning area can be set as around 100 m) is achieved shown in Fig. 14 and Fig. 15, where environment information is unknown. The start point is S point and the end point is E point. The obstacle is denoted as O. The RL is constrained by TPS and TLM, and is used to explore the suitable point from S to E point dynamically. The TLM is used to splice the RL points, in which the smooth and dynamic constraints are considered, and the two bottom obstacles are changing stochastically within a certain range. Specifically, the obstacles O_3 and O_4 are changing randomly within O_{31} and O_{41} range (as shown in light blue rectangles), so the motion planning in Fig. 14 and Fig. 15 is different.

C. Comparison experiments in certain and uncertain environments

The related experiments are conducted under unknown environments, which are certain (unknown environment information) and uncertain environments (unknown environment information) respectively, which can be seen in Fig. 16. In the experiments, the environments are the same except for the stochastic obstacles. The average distance of five uncertain experiments is 54.87 m, and the average distance of five certain experiments is 71.98 m. The experiments have shown the RLTT method may be better to adopt the uncertain environments and deal with more complex environments.

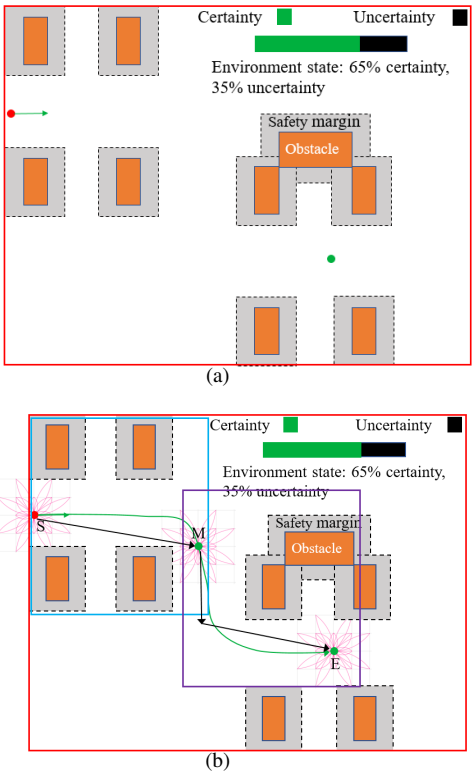

Fig. 11: Uncertain and imperfect information environments for robots. (a) is with considering dynamic constraints. (b) is with considering multi-constraint and multi-scale motion planning.

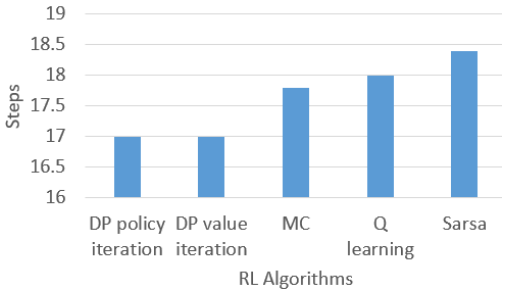

Fig. 12: Different RL methods for path search

D. Comparison experiments between RLTT and Q Learning algorithms

We conduct five experiments using RLTT method and RL algorithm respectively under unknown environments. The RL algorithm that we select is Q learning algorithm. Q learning algorithm is a classic and widely used algorithm for robot navigation and control. The computation time of each experiment and averaged computation time of five experiments using RLTT method and RL are shown at Fig. 17 and Fig. 18, respectively.

Specifically, the averaged computation time of using RLTT method for motion planning is $0.325\ (s)$, the averaged computation time of using Q learning for motion planning is $52.279\ (s)$. Moreover, the trajectory distance via Q learning is usually more than RLTT method. We select randomly one of the five experiment results and compute the trajectory distance,

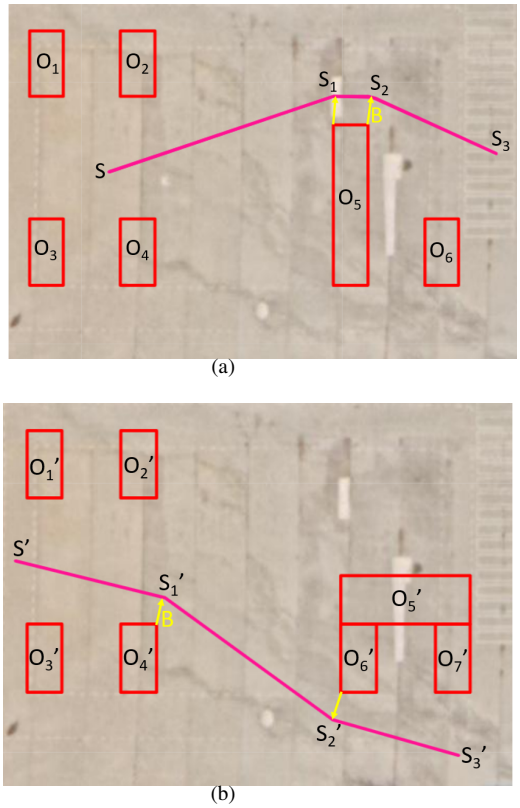

Fig. 13: Macro scale planning. (a) Convex obstacle environments. (b) Concave-convex obstacle environments.

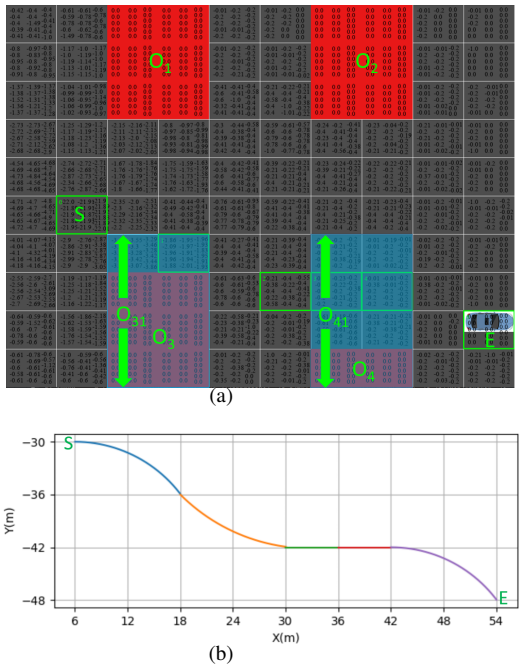

Fig. 14: Micro scale planning and random obstacle (red rectangle) experiment one. (a) grid experiment environments and RLTT method for planning (green rectangle); where the two bottom obstacle shape is changing randomly within a certain range (b) TLM motion planning (color curve) according to RLTT planning.

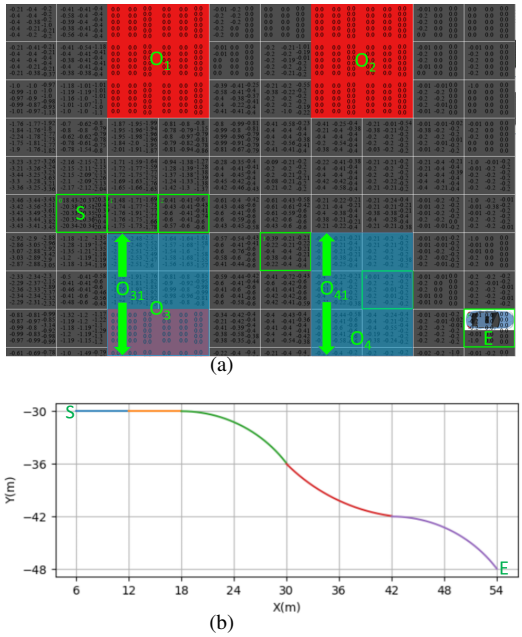

Fig. 15: Micro scale planning and random obstacle experiment two. (a) grid experiment environments and RLTT method for planning; where the two bottom obstacle shape is changing randomly within a certain range (b) TLM motion planning according to RLTT planning.

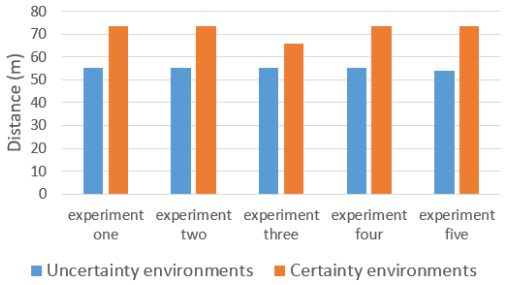

Fig. 16: Motion planning for autonomous driving in uncertain and certain environments

the distance of using RLTT method is $95.834\ (m)$, the distance of using Q learning algorithm is $126.735\ (m)$. The results indicate that the performance of RLTT method is better than the classic RL algorithm, RLTT method achieves shorter trajectory distance and uses less time for searching path compared with the classic RL algorithm. The video of the experiments is available at the link: https://sites.google.com/view/rltt.

E. Comparison experiments between RLTT and traditional algorithms

This section shows the comparison experiments between RLTT and traditional algorithms for autonomous driving motion planning. The adopted traditional algorithm is hybrid A* algorithm which is proposed by Dolgov *et al.* and is developed based on A* algorithm [39], it is very useful and widely application for autonomous driving. In Fig. 19(a) and 19(b), S denotes the start point, E denotes the end point, the green

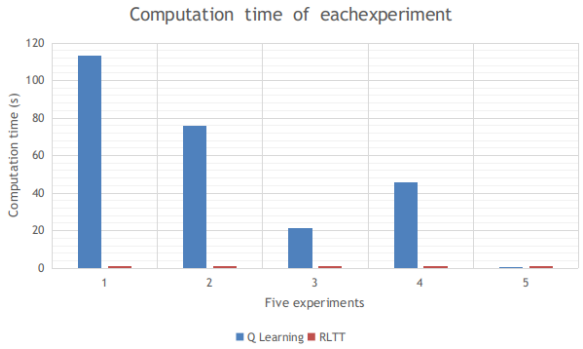

Fig. 17: Computation time of each experiment for motion planning using RLTT and Q Learning algorithms

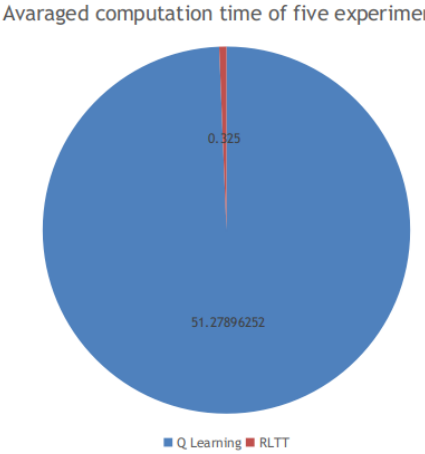

Fig. 18: Averaged computation time of five experiments for motion planning using RLTT and Q Learning algorithms

trajectory indicates the vehicle motion planning, O denotes the obstacles, W denotes the edge track of vehicles.

The distance of motion planning via the RLTT algorithm is $147.4\ (m)$ considering dynamic constraints under the environments of sparse information, and the distance of motion planning via hybrid A* algorithm considering dynamic constraints is $184.0\ (m)$ under the known environment information in the same environments. The experiment results indicate that the performance of our proposed method is better than the hybrid A* algorithm, such as the distance and smoothness, *et al.*

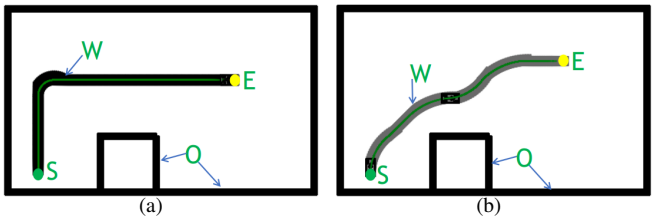

Fig. 19: (a) Hybrid A* algorithm for autonomous driving motion planning. (b) RLTT algorithm for autonomous driving motion planning.

VIII. Conclusion

In this paper, the RL method has been developed and constrained by the TLM and TPS method to achieve multi-constraint and multi-scale safe motion planning for autonomous driving under sparse information environments, in which dynamic constraints of a vehicle, smooth constraints, safety constraints and distance optimization constraints have been taken into account. In addition, the related experiments have been conducted to evaluate the effectiveness of our proposed method; the proposed method is of interpretability, can be extended and applied to other types of vehicle navigation and control, such as ground robots, unmanned surface vehicles, unmanned aerial vehicles, etc. We hope that the ingenious RLTT method may inspire new research for robotics development. In the future, motion planning for multi-vehicle cooperation may need to be considered under uncertain environments and conduct complex tasks.

Acknowledgement

We would like to thank professor Jiancheng Long for his very constructive suggestions; we would also like to thank Xiao Wang for her very useful discussion.

References

[1] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2021.

[2] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1826–1848, 2019.

[3] L. Gao, D. Chu, Y. Cao, L. Lu, and C. Wu, "Multi-lane convoy control for autonomous vehicles based on distributed graph and potential field," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2463–2469.

[4] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," in *Motion and operation planning of robotic systems*. Springer, 2015, pp. 3–27.

[5] Y. Huang, Y. Shen, J. Wang, and X. Zhang, "A platoon-centric multi-channel access scheme for hybrid traffic," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2021.

[6] H. Min, X. Xiong, P. Wang, and Y. Yu, "Autonomous driving path planning algorithm based on improved a* algorithm in unstructured environment," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 235, no. 2-3, pp. 513–526, 2021.

[7] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[8] A. Likmeta, A. M. Metelli, A. Tirinzoni, R. Giol, M. Restelli, and D. Romano, "Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving," *Robotics and Autonomous Systems*, vol. 131, p. 103568, 2020.

[9] P. Hang, C. Lv, C. Huang, J. Cai, Z. Hu, and Y. Xing, "An integrated framework of decision making and motion planning for autonomous vehicles considering social behaviors," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 458–14 469, 2020.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[11] A. Prashanth L and M. Fu, "Risk-sensitive reinforcement learning: A constrained optimization viewpoint," *arXiv*, pp. arXiv–1810, 2018.

[12] A. Tamar, Y. WU, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/c21002f464c5fc5bee3b98ced83963b8-Paper.pdf

[13] L. P. Kaelbling, "The foundation of efficient robot learning," *Science*, vol. 369, no. 6506, pp. 915–916, 2020.

[14] H. Banzhaf, P. Sanzenbacher, U. Baumann, and J. M. Zöllner, "Learning to predict ego-vehicle poses for sampling-based nonholonomic motion planning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1053–1060, 2019.

[15] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[16] T. Ort, K. Murthy, R. Banerjee, S. K. Gottipati, D. Bhatt, I. Gilitschenski, L. Paull, and D. Rus, "Maplite: Autonomous intersection navigation without a detailed prior map," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 556–563, 2019.

[17] Y. Huang, H. Wang, A. Khajepour, H. Ding, K. Yuan, and Y. Qin, "A novel local motion planning framework for autonomous vehicles based on resistance network and model predictive control," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 55–66, 2020.

[18] J. Bernhard and A. Knoll, "Robust stochastic bayesian games for behavior space coverage," *arXiv preprint arXiv:2003.11281*, 2020.

[19] H. Zhang, W. Chen, Z. Huang, M. Li, Y. Yang, W. Zhang, and J. Wang, "Bi-level actor-critic for multi-agent coordination," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7325–7332.

[20] N. Harmening, M. Biloš, and S. Günnemann, "Deep representation learning and clustering of traffic scenarios," *arXiv e-prints*, pp. arXiv–2007, 2020.

[21] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[22] S. Thornton, "Autonomous vehicle speed control for safe navigation of occluded pedestrian crosswalk," *arXiv*, pp. arXiv–1802, 2018.

[23] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[24] Z. Du, Y. Wen, C. Xiao, F. Zhang, L. Huang, and C. Zhou, "Motion planning for unmanned surface vehicle based on trajectory unit," *Ocean Engineering*, vol. 151, pp. 46–56, 2018.

[25] C. Zhou, S. Gu, Y. Wen, Z. Du, C. Xiao, L. Huang, and M. Zhu, "The review unmanned surface vehicle path planning: Based on multi-modality constraint," *Ocean Engineering*, vol. 200, p. 107043, 2020.

[26] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4889–4895.

[27] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[28] J. Gong, Y. Jiang, and W. Xu, "Model predictive control for self-driving vehicles," *Beijing Institute of Technology Press: Beijing, China*, 2014.

[29] C. Zhou, S. Gu, Y. Wen, Z. Du, C. Xiao, L. Huang, and M. Zhu, "Motion planning for an unmanned surface vehicle based on topological position maps," *Ocean Engineering*, vol. 198, p. 106798, 2020.

[30] J. R. Herring, "Opengis implementation specification for geographic information-simple feature access-part 1: Common architecture," *Open Geospatial Consortium*, p. 95, 2006.

[31] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[32] Y.-z. Chen, S.-f. Shen, T. Chen, and R. Yang, "Path optimization study for vehicles evacuation based on dijkstra algorithm," *Procedia Engineering*, vol. 71, pp. 159–165, 2014.

[33] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[34] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning." in *ICLR (Poster)*, 2016.

[35] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[37] A. Greenwald and K. Hall, "Correlated-q learning," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, 2003, pp. 242–249.

[38] D. Zhao, H. Wang, K. Shao, and Y. Zhu, "Deep reinforcement learning with experience replay based on sarsa," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016, pp. 1–6.
[39] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.