Efficient Sampling-Based Motion Planning for On-Road Autonomous Driving

Liang Ma, Jianru Xue, *Member, IEEE*, Kuniaki Kawabata, *Member, IEEE*, Jihua Zhu, Chao Ma, and Nanning Zheng, *Fellow, IEEE*

***Abstract*—This paper introduces an efficient motion planning method for on-road driving of the autonomous vehicles, which is based on the rapidly exploring random tree (RRT) algorithm. RRT is an incremental sampling-based algorithm and is widely used to solve the planning problem of mobile robots. However, due to the meandering path, the inaccurate terminal state, and the slow exploration, it is often inefficient in many applications such as autonomous vehicles. To address these issues and considering the realistic context of on-road autonomous driving, we propose a fast RRT algorithm that introduces a rule-template set based on the traffic scenes and an aggressive extension strategy of search tree. Both improvements lead to a faster and more accurate RRT toward the goal state compared with the basic RRT algorithm. Meanwhile, a model-based prediction postprocess approach is adopted, by which the generated trajectory can be further smoothed and a feasible control sequence for the vehicle would be obtained. Furthermore, in the environments with dynamic obstacles, an integrated approach of the fast RRT algorithm and the configuration-time space can be used to improve the quality of the planned trajectory and the replanning. A large number of experimental results illustrate that our method is fast and efficient in solving planning queries of on-road autonomous driving and demonstrate its superior performances over previous approaches.**

***Index Terms*—Autonomous vehicles, motion planning, on-road driving, rapidly exploring random tree (RRT).**

I. Introduction

As one of six intelligent transportation system major categories [1], autonomous vehicles were originally developed in the 1980s. Google driverless car has been allowed testing and self-driving on Michigan's roads according to the legislation. With the development of the autonomous driving technology, the autonomous vehicle has become one of the key issues for supporting our daily life and economical activities. For example, new transportation service in the city, personal supporting service for elderly people and persons with handicaps, and logistics for delivery service are typical significant applications.

Motion planning is an important and fundamental technology of autonomous vehicles [2], which can solve the problem of computing a sequence of control values or feasible movement states for the vehicle to maneuver among obstacles from an initial state toward a desired terminal state, taking into account the vehicle's kinematic and dynamic model [3]. In addition, the traffic rules and the geometric structural information of roads are also considered in the motion planning for on-road autonomous driving.

Motion planning problem has been widely studied in the field of mobile robotics, and many motion planning approaches have been presented in the robotics literature over the past three decades [4]. Some well-known planning algorithms, such as Dijkstra [5] and A* [6], can find the shortest path in an equally sized grid map representing the environment information. However, the choice of the grid resolution and tracking the path composed of the connected grids are always difficult problems. D* [7] and its variants [8] also cannot address these weaknesses of the grid-based algorithm, although they introduce the environmental dynamic. The potential field approach [9] can plan a path using the repulsive forces from obstacles and the attractive forces from the goal. However, the virtual forces trapping in the local minima will make a path that is not always available. One common drawback of these algorithms is that they cannot take the complex dynamic and differential constraints of the vehicle into consideration easily. Therefore, they are not very suitable for the application of autonomous vehicle, particularly in the context of high-speed or precise motion control for on-road driving.

Recently, sampling-based techniques for motion planning have become quite popular and widespread. Two chief families of sampling-based techniques are single-query algorithms and multiquery algorithms [10]. The single-query algorithms are in general regarded to be better and more suitable for motion planning of the single robot, because it does not normally invest substantial time to preprocess the models and cover the free space in the same manner that multiquery algorithms would [11]. Rapidly exploring random tree (RRT) is a primary single-query algorithm based on incremental sampling, which was first proposed by Steven in a technical report [12]. Its main advantage is that it is easy to take into account the complex system dynamics by directly using the control set of admissible inputs

Manuscript received November 26, 2014; accepted December 22, 2014. Date of publication February 13, 2015; date of current version July 31, 2015. This work was supported in part by the National Natural Science Foundation of China projects under Grants 91320301 and 61273252. The Associate Editor for this paper was L. Li.

L. Ma, J. Xue, J. Zhu, C. Ma, and N. Zheng are with The Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: mlxjtu@gmail.com; jrxue@mail.xjtu.edu.cn; zhujh@mail.xjtu.edu.cn; machao0919@stu.xjtu.edu.cn; nnzheng@mail.xjtu.edu.cn).

K. Kawabata is with The Institute of Physical and Chemical Research (RIKEN), 2-1, Hirosawa, Wako, Saitama, 351-0198, Japan (e-mail: kuniakik@riken.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2015.2389215

[11], [13]. Hence, RRT was widely used for motion planning of mobile robots and manipulations. Nevertheless, there is still a large gap between the basic RRT and those applications. Many RRT variants were proposed for advancing it. RRT-Connect, a bidirectional version, synchronously searches with two trees from start point and goal point for improving the efficiency [14]. This RRT variant was applied to parking of intelligent vehicles [15], but the difficulty of using this algorithm is how to deal with the discontinuity existing at the place where two trees connect. Execution extended RRT biases search toward goals and waypoints [16], and Urmson and Simmons introduced a heuristic function for biasing tree growth [17]. These bias techniques are very efficient and are used in a wide range of applications. Rodriguez *et al*. [18] presented an obstacle-based RRT algorithm that can exploit in nine different methods based on the obstacle geometries. Although the obstacle vectors cannot always be obtained accurately, selecting various growth methods is a novelty. Subsequently, Denny *et al*. introduced the machine learning method into selecting of growth methods [19]. Particle RRT (pRRT) casted the particle filter into the RRT's framework [20], which can select the most promising nodes from the distributions of states instead of single states. Because pRRT is time consuming, it is used to teleoperate a Mars exploration rover rather than a real-time autonomous vehicle. In 2011, Karaman and Frazolli presented an optimal RRT (RRT*) that can get an asymptotic optimal path, which the RRT algorithm lacked [21], and is applied to off-road vehicle maneuvers [22]. The asymptotic optimal solution is obtained at the cost of consuming much time by a rewire process. In recent years, many works have been done on improving the efficiency of RRT* [23]–[25]. However, the efficiency and trajectory quality of RRT algorithm are still worth being studied. This is because the RRT* algorithm is to constantly rewire and refine on the basis of the planning results of the RRT algorithm.

Note that autonomous vehicles competitions promoted the development of the autonomous driving technology, particularly the application of motion planning technology to autonomous vehicles. For example, a series of autonomous vehicles competitions were organized by the Defense Advanced Research Projects Agency in the U.S. The winner team "Boss" from Carnegie Mellon University in the 2007 Urban Challenge applied anytime dynamic A* to off-road driving [26] and the model-predictive trajectory generation method to on-road driving [27]. This trajectory generation method takes into account the vehicles' kinematics and dynamics, resulting in smooth and feasible trajectories, but it does not consider the obstacles in the process of trajectory generation. The obstacle avoidance is accomplished in the manner of lane changing by selecting from candidate trajectories, which is probably not able to deal with the cluttered environments. Furthermore, the error in linearizing the system may lead to the terminal state error of the generated trajectory. The autonomous vehicle from Stanford University also applied different approaches to common road navigation and free-style navigation, including a parallel trajectory planner and a modified version of A* [28]. The parallel trajectory planner with the cumulative cost computed can accomplish the passing on road, but it is unable to handle the cases of blocked roads or intersections. The MIT team proposed a unified motion planning method against the whole competition. An improved RRT as a real-time online motion planning algorithm was used in the MIT's autonomous vehicle [29], where an important extension was the use of closed-loop prediction model [30]. Furthermore, Hanyang University's "A1" won autonomous vehicle competition in Korea. The highlight of their work is the design of a local path planning approach for the off-road autonomous driving [31]. In China, the Intelligent Vehicle Future Challenge competition has been held for several times, hosted by the National Natural Science Foundation of China [32]. Some teams attempt to solve the motion planning problems of autonomous vehicles by the RRT algorithm. For instance, an RRT variant is presented by Du *et al.* [33]. This algorithm is only applied to the narrow passages scenario. In addition, the postprocess operation of the algorithm will possibly make the RRT algorithm lose its advantage of satisfying the system dynamic constraints.

Easily introducing system dynamics and favorable search ability make the RRT algorithm fit for solving the motion planning problems. Some of the aforementioned applications achieved good performance. However, there exist some weaknesses in the motion planning of autonomous vehicle when using the RRT algorithm.

- The path generated by RRT is often jagged and meandering, owing to the growth way of the tree. Although a lot of works were proposed to smooth the path planned by RRT, lessened meanders might still exist in connects of two edges. For instance, lane keeping is the most common maneuver for vehicles on road when without obstacles. Using the RRT makes it difficult to plan a trajectory shaped like a straight line for the lane keeping.
- Whether for RRT or RRT*, the accurate terminal configuration cannot always be reached. Both algorithms extend by the forward kinematics, and the extension result is that the new branch grows toward the sample configuration or the goal configuration, but the new node of this branch does not necessarily reach those configurations. Thus, one of the stop conditions of such algorithms is often that the tree enters a certain goal region. This is risky, if the vehicle cannot reach the desired goal state.
- An important requirement of motion planning for autonomous vehicles is real time, whereas the efficiency of RRT mainly depends on whether sampling domain well adapts to the problem or not. To address this issue, the method in [29] used the Gaussian distributions over sampling domain, whose related parameters were then determined according to the traffic scenes. However, for on-road environments, can the efficiency of RRT algorithm be further improved?

To overcome these shortages presented above, this paper focuses on an efficient sampling-based motion planning method for on-road autonomous driving, including an improved RRT variant, a model-based prediction postprocess, and the integration with configuration-time space. A large number of experimental results illustrate that the proposed method is faster and more efficient and its planned trajectory can accurately

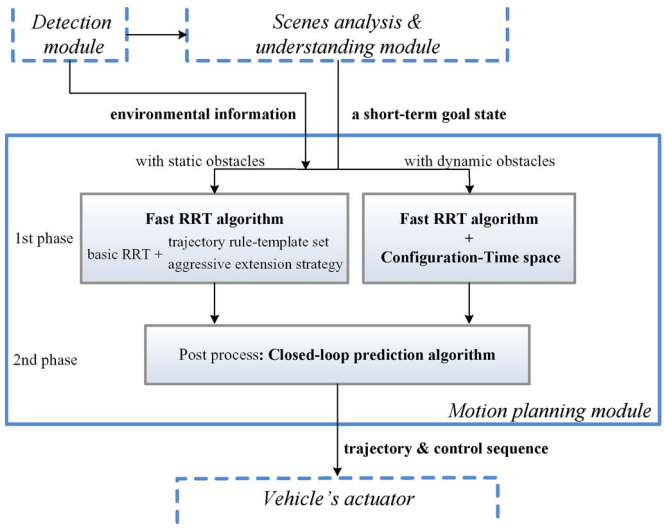


Fig. 1. Overall structure of our method.

reach the desired configuration compared with other related algorithms. The meanders of trajectory can also be eliminated in most cases.

The rest of this paper is organized as follows. Section II presents the framework overview of our method. Section III introduces the problem formulation of motion planning with respect to the autonomous vehicle. Section IV, after reviewing the basic RRT algorithm, describes the fast RRT algorithm in detail. Section V analyzes the existing problems in the replanning process of motion planning algorithms and introduces an integrated approach of the fast RRT and the configuration-time space. Section VI discusses the experiments and results, including a large number of statistical experiments in different scenes and maneuvers. Section VII concludes this paper.

II. Overview of the Framework

The overall structure of our method is illustrated in Fig. 1. Our method is designed for the motion planning module of on-road autonomous driving, which receives a short-term goal state and environmental information from high-level modules, scenes analysis and understanding module, and detection module, respectively. The output of our method is a smooth trajectory and a feasible control sequence, which are sent to the vehicle's actuator, aimed at controlling the vehicle to reach the goal state without collisions. The core of our method is an improved RRT variant, which is named fast RRT algorithm. Based on previous work [29], it introduces a rule-template set (detailed in Section IV-C) in terms of the traffic scenes and an aggressive extension strategy (detailed in Section IV-D) of search tree [34]. A model-based prediction postprocess (detailed in Section IV-E), a closed-loop prediction approach, is adopted for further smoothing and computing the trajectory and control sequence for the vehicle. To address the motion planning problem under the environments with dynamic obstacles, the fast RRT algorithm and the configuration-time space are integrated (detailed in Section V) to improve the quality of the planned trajectory and the replanning.

III. Problem Formulation

A. Motion Planning Definition

The motion planning problem is always formulated as in the following. The state space and the control space of system are represented by compact sets $X \subset \mathbb{R}^n$ and $U \subset \mathbb{R}^m$, respectively. The time-invariant dynamical system, i.e.,

$$\dot{x}(t) = f\left(x(t),\, u(t)\right) \quad x(0) = x_0 \tag{1}$$

where $x(t) \in X$, $u(t) \in U$, and x_0 is the initial state. In addition, the desired goal state is represented as $x_{goal} \in X$. Denote $X_{obs} \subset X$ as the regions occupied by obstacles; an obstacle-free region can be defined as $X_{free} \subset X \backslash X_{obs}$. The motion planning task is to compute

$$\begin{aligned} &x(t) \in X_{free} \qquad \forall t \in [0, t_f] \\ &x(t_f) = x_{goal} \\ &u(t) \in U \qquad \forall t \in [0,\, t_f] \end{aligned} \tag{2}$$

where x is called the trajectory, and $x(t)$ is the reachable state of system; u is called the control input sequence, and $u(t)$ is the feasible control variable for the system. When the control input is $u(t_f)$ at time $t = t_f$, the system can reach the goal state x_{goal}.

B. The Vehicle Model

Simplified from a front-wheel steering Ackerman vehicle, the bicycle model is employed, because it is suitable for our autonomous vehicle prototype KuaFu-1, which is developed for the Intelligent Vehicle Future Challenge competition in China, as shown in Fig. 2. Since the main task is to realize self-drive on road in urban environments, its motion planning problem is considered in the 2-D plane. The kinematic model is described by

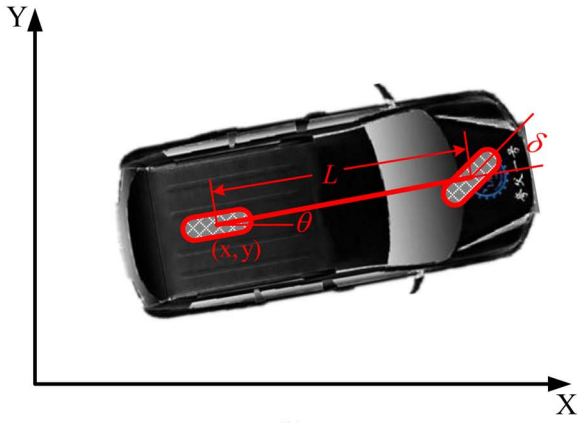
$$\begin{bmatrix} \dot{\mathrm{x}} \\ \dot{\mathrm{y}} \\ \dot{\theta} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ \frac{\tan(\theta)}{L} \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \dot{\delta} \tag{3}$$

where x and y denote the coordinates of the center point of the rear axle, θ is the vehicle heading angle with respect to the x-axis, and L is the vehicle wheelbase. As control input parameters to vehicle, v and $\dot{\delta}$ are the longitudinal velocity and the angle velocity of the steering wheel, respectively. Moreover, some important boundary parameters derived from the prototype are used to constrain the vehicle model as

$$\begin{aligned} &L = 2.790\ (\mathrm{m}) \\ &v_{\min} = 0\ (\mathrm{m/s}) \qquad v_{\max} = 12\ (\mathrm{m/s}) \\ &\delta_{\min} = -0.5236\ (\mathrm{rad}) \qquad \delta_{\max} = 0.5236\ (\mathrm{rad}) \\ &a_{\min} = -5.0\ (\mathrm{m/s^2}) \qquad a_{\max} = 0.9\ (\mathrm{m/s^2}) \\ &\dot{\delta}_{\min} = -0.2183\ (\mathrm{rad/s}) \qquad \dot{\delta}_{\max} = 0.2183\ (\mathrm{rad/s}) \end{aligned} \tag{4}$$

where a is the forward acceleration, and δ represents the steering wheel angle.

(a)



(b)

Fig. 2. Our autonomous vehicle prototype and its geometric model. (a) KuaFu-1: autonomous vehicle of Xi'an Jiaotong University. (b) Autonomous vehicle's bicycle model.

In this paper, the vehicle model is used in four different aspects: 1) the generation of the rule templates; 2) the extension of the search tree; 3) the trajectory generation using the aggressive extension strategy; 4) the closed-loop simulation in the model-based prediction stage.

IV. Algorithm Description

Here, we propose a fast RRT algorithm for on-road driving of the autonomous vehicles, and the algorithm flow is described in detail. We first present a review on the operation of the basic RRT algorithm and then the proposed fast RRT algorithm.

A. Basic RRT

Given an initial state as a root, the basic RRT incrementally grows a tree to explore the state space outward from the root until the goal state is reached, by iterations. Each iteration consists of three main steps:

1) Draw a random sample, i.e., x_{rand}, from the state space over a specified sampling distribution, i.e., $p_{rand}(x)$;
2) Find the nearest node in the tree to the sample using a designated distance metric, i.e., $\rho(x)$;
3) Expand the selected nearest node toward the random sample, when a control input is determined for forward simulation. The resulting state, x_{new}, is then obtained and added to the search tree.

Algorithm 1 Fast RRT algorithm for Autonomous Vehicles

fRRTmain()
1: $template_i \leftarrow$ LoadTemplate$(x_{goal}, TemplateSet)$;
2: $\mathcal{T}_{temp} \leftarrow$ DecomposeCurve$(template_i)$;
3: $\mathcal{T}_{trimtemp} \leftarrow$ TrimTree$(\mathcal{T}_{temp})$;
4: $\mathcal{T}$.AddSubTree$(\mathcal{T}_{trimtemp})$;
5: **if** $success ==$RushTowardsGoal$(\mathcal{T}, x_{goal})$ **then**
6: **return** $\mathcal{T}$;
7: **end if**
8: **for** $k = 1$ **to** K **do**
9: **if** Random$(0, 1) < \lambda_{ss}$ **then**
10: **if** $success ==$RushTowardsGoal$(\mathcal{T}, x_{goal})$ **then**
11: **return** $\mathcal{T}$;
12: **end if**
13: **else**
14: $x_{rand} \leftarrow$ GetRandomSample(x_{goal});
15: $x_{near} \leftarrow$ GetNearestNeighbor$(\mathcal{T}, x_{goal})$;
16: $x_{new} \leftarrow$ Extend$(\mathcal{T}, x_{near}, x_{rand})$;
17: **if** CollisionFreeDetection(x_{near}, x_{new}) **then**
18: $\mathcal{T}$.AddVertex(x_{new});
19: $\mathcal{T}$.AddEdge(x_{near}, x_{new});
20: **end if**
21: **end if**
22: **end for**

RushTowardsGoal$(\mathcal{T}, x_{goal})$
1: $x_{near} \leftarrow$ GetNearestNeighbor$(\mathcal{T}, x_{goal})$;
2: $\mathcal{P}_{near2goal} \leftarrow$ GeneratePath(x_{near}, x_{goal});
3: $\mathcal{T}_{2goal} \leftarrow$ DecomposeCurve$(\mathcal{P}_{near2goal})$;
4: **if** CollisionFreeDetection$(\mathcal{T}_{2goal})$ **then**
5: $\mathcal{T}$.AddSubTree$(\mathcal{T}_{2goal})$;
6: **return** $success$;
7: **else**
8: $\mathcal{T}_{trim2goal} \leftarrow$ TrimTree$(\mathcal{T}_{2goal})$;
9: $\mathcal{T}$.AddSubTree$(\mathcal{T}_{trim2goal})$;
10: **return** $failure$;
11: **end if**

B. Overview of the Fast RRT Algorithm

An overview of the proposed approach is given in Algorithm 1. Compared with the basic RRT, two improvements, namely, the rule templates and an aggressive extension strategy, are added. By introducing the rule templates of the trajectories, more nodes and edges will derive from the rule templates rather than online generation in the initial phase if the obstacles distribute sparsely on road. Furthermore, using the aggressive extension strategy with a certain probability can not only accelerate the growth speed of the search tree at the end of the period but also increase the accuracy of the terminal state of the planned trajectory.

A rule-template set contains several rule templates, which is generated offline according to the context of traffic scenes.

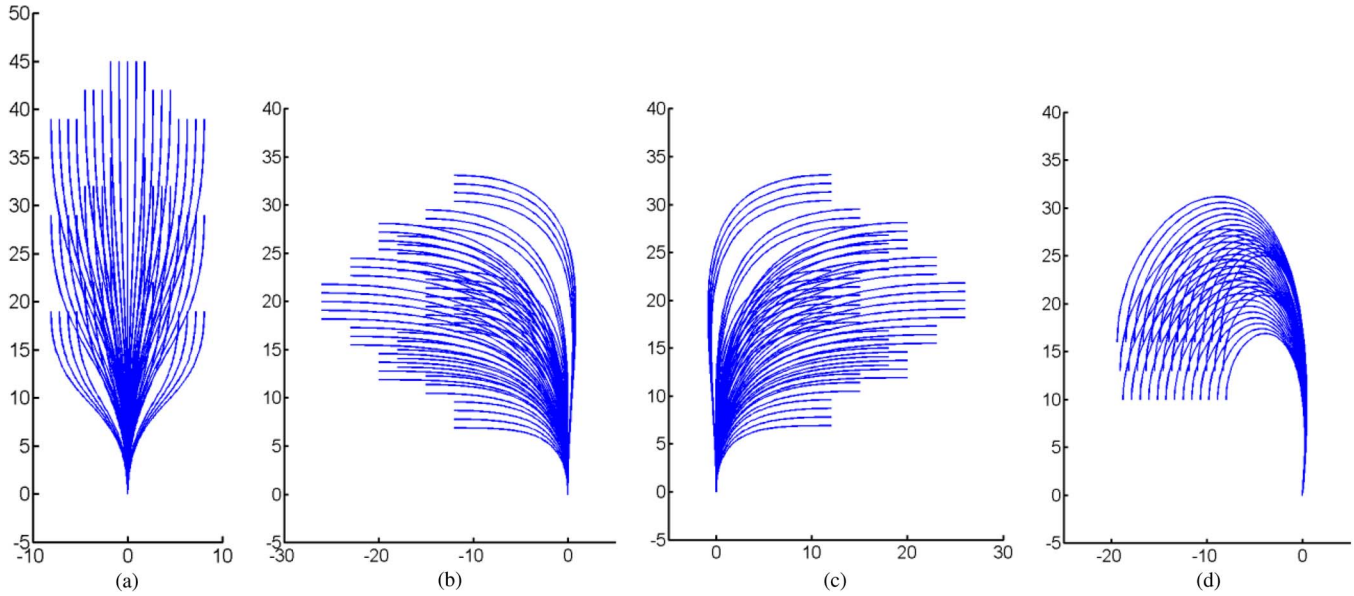


Fig. 3. Various rule templates based on different maneuvers. (a) Go straight. (b) Turn left. (c) Turn right. (d) U-turn.

At the beginning of the proposed algorithm, the rule-template set is loaded, and an appropriate rule template can be selected automatically according to a short-term goal state (*line 1*). This template is then decomposed and saved as the treelike structure (*line 2*). If some nodes and edges of a trajectory in the rule template are not collision free with the obstacles or the boundaries of the road, these portions will be discarded, and the remainders in the rule template will be retained (*line 3*) and added to the root (*line 4*). Thus, at this moment, the tree probably has already possessed a part of branches and leaves before randomly searching. The aggressive extension strategy is first used after the rule template is loaded and trimmed (*lines 5–7*), by calling the function RUSHTOWARDSGOAL(), in which a trajectory generator is used for planning from the current state to the goal state. This aggressive strategy can directly plan a trajectory toward the goal state when without obstacles, to avoid the meaningless search. If the function is executed successfully, the algorithm ends and returns the tree. If the function fails, the algorithm starts the iteration step. In each iteration step (*lines 8–22*), two extension strategies are chosen stochastically, namely, the aggressive extension strategy and the ordinary extension strategy. We denote the probability of the aggressive extension strategy being selected as λ_{ss}. The aggressive extension strategy is used if a random number (0 to 1) is less than λ_{ss} (*line 9*). In this case (*lines 10–12*), the function RUSHTOWARDSGOAL() is still called; then, the result is that a trajectory reaching the goal will be found (if return success) or a partial collision-free trajectory is added to the tree and the algorithm continues (if return failure). Otherwise, an ordinary extension strategy similar to that of the basic RRT algorithm is chosen (*lines 13–21*). During this process, we adopt some small modifications that refer to [29], such as the sampling distribution and the Dubins distance metric. The stop condition of the proposed algorithm is that the goal state is reached by the aggressive extension strategy or the maximum number of iterations K is exceeded.

C. Generation and Trim of Trajectory Rule Templates

Here, more details about the design and using of the rule templates are described.

Compared with a mobile robot running in the maze, the autonomous vehicle traveling on road in urban environments has the characteristic of simple and relative monotonous maneuvers. Some former researchers mainly focused on the path search for mobile robots in the maze. The environment in the maze is complicated, and the motion of robots is diverse and hardly predicted. Likewise, the cases of some complex system planning in the high-dimensional configuration space are similar. Therefore, the rule template cannot be used in these cases. However, there is a lot of geometric structural information in the urban environments. A vehicle is always going straight and changing the lane in most of the time when it runs on the road. A small portion of time is for turning, and a smaller portion of time is for U-turn. Hence, the maneuvers can be categorized in terms of the traffic scenes, by which the rule templates can be generated and applied.

Currently, the maneuvers are roughly divided into four types, namely, go straight, turn right, turn left, and U-turn. In each category, considering the driving experience and prior knowledge, various terminal states are casted. These terminal states with different longitudinal and horizontal distances to the start state are set, in order that generated rule template can cover five lanes centered at the current lane. For each given terminal state, a trajectory generator based on inverse kinematics can be used to solve this two-point boundary value problem for generating trajectories in the rule template [3]. In addition, a path generator considering the vehicle constraints can also be employed to rapidly generate paths [35], as these paths will be transferred to the trajectories and control input in the model-based prediction stage. Fig. 3 presents four offline-generated rule templates based on different maneuvers. For example, in the "Go straight" rule template shown in Fig. 3(a), the horizontal distance ranges

from −8.1 to 8.1 m, which is able to cover five lanes. There are three ranges for the longitudinal distance of terminal states of trajectories, namely, from 19 to 25 m, from 29 to 35 m, and from 39 to 45 m. Note that the design of rule-template size needs to consider the real perception range of the autonomous vehicles. In addition, the "U-turn" rule template shown in Fig. 3(d) is only for turning left because this is to obey Chinese traffic rules.

Various generated rule templates constitute a template set that is loaded before the algorithm is executed and from which an appropriate rule template will be selected as the basis for the later growth of the tree. Generally, a higher level module, i.e., the scenes analysis and understanding module, offers a short-term planning goal according to the local environmental information perceived by the sensors on the vehicle. The desired position and heading of this goal state are to separately match with that of each rule template. The rule template with successful match both in position and heading will be easily picked out from the rule-template set loaded. For example, the position of a given goal state is [20, 20] and the heading angle is 90°, then the rule template "Turn right" is selected.

The selected rule template first needs to be decomposed and saved as the treelike structure. Then, the nodes and edges in the rule template will be checked for collisions with the boundaries of the road and obstacles detected. The collision-free trajectories will be reserved, while the trajectories with possible collision are not discarded entirely. Instead, the collision portion is abandoned on the basis of the nodes, whereas the collision-free portion is reserved. This process is shown in later experimental results.

As the application of the rule template, the tree quite probably has already possessed a part of branches and leaves rather than only a root before randomly searching in the beginning. Usually, it is unlikely to be trimmed for all branches and leaves, because the vehicle on road is hardly blocked by obstacles at a close distance in most cases. These remaining branches and leaves can save the time spent on the initial search of the tree.

Furthermore, some difficult situations in using the rule template are also discussed. Usually, the automatic selection of the appropriate rule template has high accuracy by using the position and heading of the goal state. However, the selection is possibly ambiguous when a road is not straight such as the roads in the ramp of the interchanges or built along the rivers or mountains. We can still use the "Go straight" rule template against the road with small curvature. If the road has great curvature, both the go straight template and the corresponding tuning template possibly match with the obtained short-term goal state. We can randomly select one from these possible rule templates. Most of the branches and leaves are trimmed because of collisions with road boundaries. Only those branches and leaves that are close to start state are retained, probably less than 10 m. Hence, the retained parts have no big differences, no matter if the "Go straight" or the "Turn right/left" rule template is selected. In the extreme case, a root can be returned without selecting any rule template. The search and the aggressive extension strategy will still work afterward. Moreover, for special or unexpected roads, their rule templates can be customized or obtained by scaling the existing rule templates before the driving.

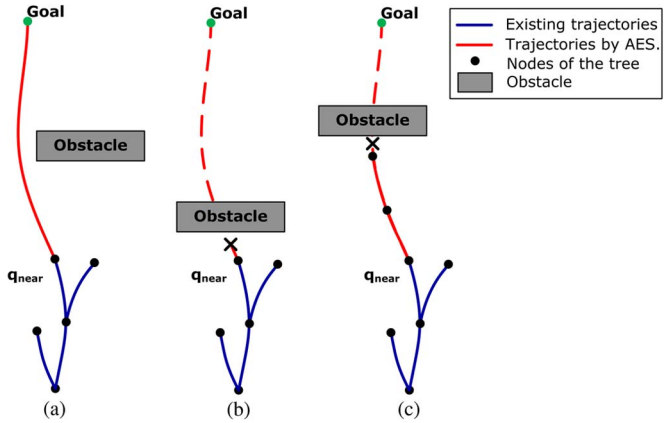


Fig. 4. Extension results by the aggressive extension strategy. (a) Success. (b) No extension. (c) Partial extension with multiple edges.

D. Aggressive Extension Strategy

Here, the principle and the implementation process of the aggressive extension strategy are illustrated in detail. Meanwhile, the difference between the aggressive extension strategy and the goal bias [13], [17] used in the traditional RRT is introduced specifically.

It is well known that traditional RRT grows step by step, and each step length is fixed or has few changes. However, under the environment with a few obstacles, "facing" the goal without hindrance, the step-by-step way of growth, seems to be slow and clumsy, leading to meandering paths. To solve these problems, the aggressive extension strategy is introduced. Under open environments, the aggressive extension strategy can make a certain branch grow as long as possible from q_{near}, even to the goal. The implementation of the aggressive extension strategy is by means of the function RUSHTOWARDSGOAL() in Algorithm 1. Its principle is to take the goal state as sampling state and plan a trajectory directly from the current state to the goal state using the same trajectory generator as in Section IV-C (*line 2*). If it is successful, the trajectory reaching the goal will be found (*line 4–6*), but if not, the collision-free part of the generated trajectory will be reserved in the search tree (*line 8–10*). Even so, the search tree will increase a relatively large branch. As shown in Fig. 4, there are three results for the aggressive extension strategy. Fig. 4(a) presents a successful result of this strategy, where the trajectory can rush directly toward the goal from q_{near} without any collisions. This also means that the planning task is accomplished successfully. Sometimes, the trajectory generated by this strategy cannot reach the goal directly and would collide with obstacles. In this case, if the length of the collision-free part is smaller than that of an edge in the tree, the whole trajectory generated will be discarded, as shown in Fig. 4(b). On the contrary, if the length of the collision-free is larger, its trajectory will be saved as a part of the treelike structure. In Fig. 4(c), two nodes and two edges are added into the tree, which needs two iterations in the traditional RRT. If under open environments, more nodes and edges are probably added into the tree in one-time calling aggressive extension strategy, greatly increasing the growth speed of the tree.

The aggressive extension strategy seems similar to the goal bias strategy used in traditional RRT. In fact, they are quite

different. Although the goal bias strategy takes the goal as sampling state as well, there is no change and improvement in its way of extension, which extends toward the goal by using the forward simulation in constant step length or time interval. In other words, it can only grow one step toward the goal at a time. Compared with the goal bias, our extension strategy is more aggressive, which can plan a trajectory from current state to goal state based on the inverse kinematic.

This extension strategy is applied in two different places of the proposed algorithm (*line 5* and *line 10*). First, it is definitely used before the search, which has the advantage of direct trajectory planning under the collision-free environment. Second, it is used with a certain probability in the iterative search, which can not only promote rapid growth of the search tree but also make the tree rush directly toward the goal when passing the obstacles in the later period of search. In this case, one of the most important problems is how to determine probability λ_{ss} of the aggressive extension strategy being chosen. We design it as an adaptive adjusting parameter. Initially, $\lambda_{ss} = 0.2$; after the calling aggressive extension strategy, the probability λ_{ss} will be updated based on whether the search tree is extended or not, which is shown as

$$\lambda_{ss}(k+1) = \begin{cases} \lambda_{ss}(k) + (1-\lambda_{ss}(k)) \\ \quad \times \left(\gamma e^{-\frac{1}{2n_{po}}} + (1-\gamma)\frac{l_e}{L_e}\right) & \text{if extendable} \\ \lambda_{ss}(k) e^{-\frac{1}{2n_{ne}}} & \text{if unextendable} \end{cases} \tag{5}$$

where $\gamma \in [0, 1]$ is a constant describing a weighting for two factors that influence the change in λ_{ss}. These two factors include the number of consecutive successful extensions n_{po} and the trajectory length of the extension l_e, when the search tree can be extended by the aggressive extension strategy. $L_e = \max\{l_e, l_{\max}\}$ denotes the larger one between the current extension trajectory length l_e and the length of the longest trajectory in the tree $l_{\max}$. Contrarily, when the search tree is not extended by the strategy, the probability λ_{ss} will decrease according to the number of consecutive unsuccessful extensions n_{ne}. n_{po} and n_{ne} are a pair of mutually exclusive parameters. In the extendable and unextendable cases, one is accumulated and the other one is reset as zero. Furthermore, once a node in the tree is chosen as the q_{near} and the aggressive extension strategy is executed, the result will be certain. It is unnecessary to reuse this strategy to the same node. Hence, after executing the aggressive extension strategy, the node then will be put into a closed list and will not be used again. The probability of the aggressive extension strategy being chosen is adaptive to the heterogeneous environment. When the environment is cluttered, the aggressive extension strategy will be chosen with a small probability, and under open and sparse environments, this strategy will be adopted with a high probability.

Using the aggressive extension strategy can not only speed up the growth of the search tree and avoid some meaningless searches but also improve the terminal state accuracy of the planned trajectory. The reason is that, as a stop condition of the proposed algorithm, this strategy makes the planned trajectory accurately reach the desired terminal state, not just a region.

E. Model-Based Prediction Stage

When the search tree reaches the goal state, the task of Algorithm 1 is completed, and a tree T is returned. Here, a type of forward simulation approach based on the vehicle's model, named the closed-loop prediction, is adopted [30], which can further smooth the portion of the rule template and the portion of tree generated online, while a new trajectory and a feasible control sequence for the vehicle can be calculated and obtained.

Algorithm 2 Closed-Loop Prediction Trajectory Generation

1: $\mathcal{P}_{root2goal} \leftarrow$ EXTRACTBRANCH$(\mathcal{T}, x_{goal})$;
2: $\mathcal{X}_0 \leftarrow \mathcal{T}_{root}$
3: **while** x_{goal} not reached **do**
4: $\quad \mathbf{u}_i \leftarrow$ PUREPURSUITPI$(\mathcal{X}_i, \mathcal{P}_{\text{root2goal}}, \mathbf{M}_{vehicle})$;
5: $\quad \mathcal{X}_{i+1} \leftarrow$FORWARDSIMULATION$(\mathcal{X}_i, \mathbf{u}_i, \mathbf{M}_{vehicle}, \Delta t)$;
6: $\quad i = i + 1$;
7: **end while**
8: **return** $\mathcal{X}, \mathbf{u}$;

Algorithm 2 shows the overall flow of the closed-loop prediction algorithm. In a backward manner, the function EXTRACTBRANCH() can extract the trajectory/path $\mathcal{P}_{root2goal}$ in the tree starting from the goal state x_{goal} (*line 1*). Before the forward simulation, the root of the tree is taken as the initial state of the trajectory (*line 2*). As a reference input, $\mathcal{P}_{root2goal}$ is sent to a stable closed-loop system, which consists of a steering controller by the Stanley method, which is an improved Pure-Pursuit [36], a proportional–integral speed controller [29], and the vehicle's model (*line 4*). The steering and speed controller is presented as

$$\delta_{\mathbf{u}} = \theta_e + \tan^{-1}\left(\frac{k_s d_e}{v_{\text{act}}}\right) \tag{6}$$

$$v_{\mathbf{u}} = k_p(v_{\text{cmd}} - v_{\text{act}}) + k_i \int_0^t (v_{\text{cmd}} - v_{\text{act}})\, d\tau \tag{7}$$

where θ_e is the angle error between the current heading of the vehicle and the heading of the path at the nearest point. d_e denotes the distance error to the path. v_{act} and v_{cmd} represent the current actual speed and commanded speed, respectively. k_s is a gain parameter of the Stanley method. k_p and k_i are the proportional and integral gain parameters. When the control signal $\mathbf{u}_i$ from the closed-loop system is taken as the input to the vehicle's model, the new state in the trajectory will be generated by run ning forward simulation for the time interval Δt (*line 5*). Repeat steps in lines 4 and 5 until the resulting trajectory reaches the goal state. Finally, an easy-tracking trajectory $\mathcal{X}$ and a feasible control sequence $\mathbf{u}$ for the vehicle will be obtained.

V. PLANNING IN THE CONFIGURATION-TIME SPACE

The proposed algorithm, like many other motion planning algorithms, only applies to the environments with static obstacles. However, in real world, there are a lot of dynamic obstacles, particularly in on-road environments. Generally, under dynamic and uncertain environments, replanning technology is often used [37], [38], which repeatedly calls the planning algorithm

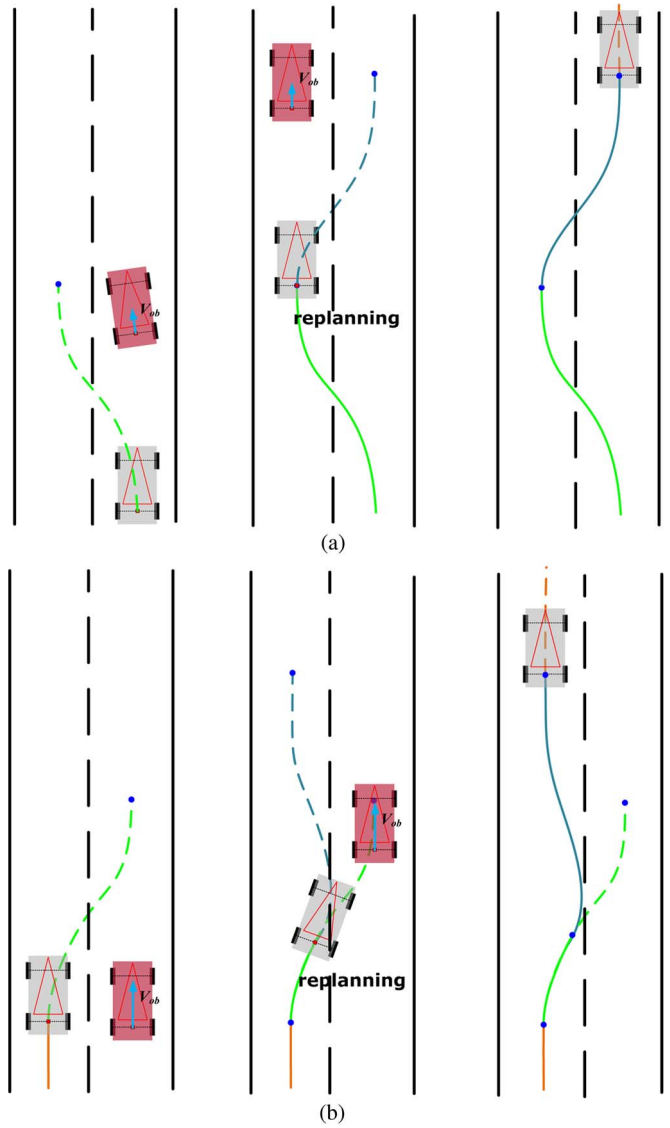


Fig. 5. Unideal replanning cases due to not considering dynamic obstacles. (a) Avoiding an obstacle that has left, due to taking the dynamic obstacle as a static one initially. After replanning, unnecessary lane changing is executed twice. Light green lines represent the trajectories before replanning. Cadet blue lines denote the planned trajectories after replanning. (b) Dangerous example of lane changing, due to not taking the speed of the dynamic obstacle initially. The coming collision triggers the replanning mechanism, resulting in a meandering curve.

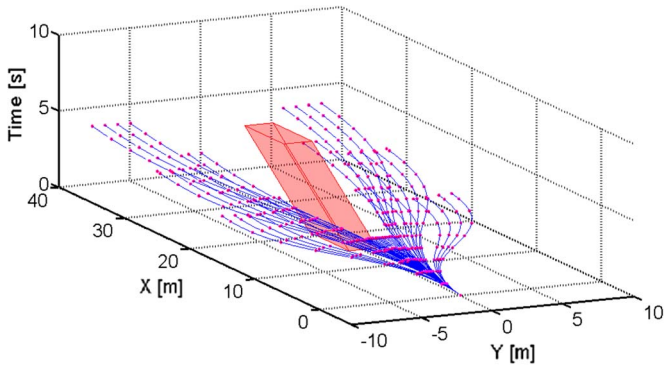


Fig. 6. Tree and dynamic obstacle in the configuration-time space.

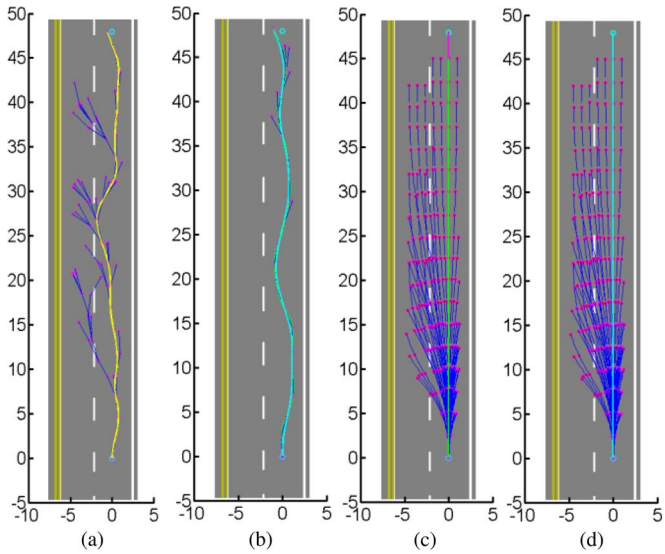


Fig. 7. Comparison among different methods in lane keeping maneuver. (a) Result of RRT. (b) Result of CL-RRT. (c) First phase result of our method. (d) Second phase result of our method.

with respect to static obstacles. There are two ways of calling. One is to repeatedly call the planning algorithm with a fixed duration. The other one is to call the planning algorithm in cases where the environmental changes affect the original planning, for example, the planned trajectory ahead has been occupied by obstacles and the avoidance becomes meaningless due to the departure of the obstacles, as shown in Fig. 5. In a real-time planning system, using the first way of calling is always avoided because of its large computational overhead. If the second way is adopted, dynamic obstacles' velocity information should be considered in the planning algorithm to plan a more feasible trajectory.

We combine the proposed fast RRT algorithm with the configuration-time space to improve the performance of the replanning under environments with dynamic obstacles. The configuration-time space is to incorporate the time as an additional dimension in the configuration space, denoted by **CT**. In the configuration-time space, the vehicle and obstacles all can be presented as pairs $\langle c,\, t\rangle$, where c is the configuration of the vehicle or obstacle at time $t \in [0,\, \infty)$. According to the current velocities of obstacles, both static and dynamic obstacles can be extruded and transformed to static obstacles in this 3-D space. The planning task in the configuration-time space is formulated as finding a 3-D trajectory $\mathcal{T}_{\mathbf{CT}}$ such that $\mathcal{T}_{\mathbf{CT}}(0) = x_0$, $\mathcal{T}_{\mathbf{CT}}(t_f) = x_{goal}$, and $\forall\, (t \in [0,\, t_f] :: \langle \mathcal{T}_{\mathbf{CT}}(t),\, t\rangle \in \mathbf{CT}_{\text{free}})$, where t_f denotes the arrival time of the trajectory. In the extension of the search tree, the arrival time t of each state in new nodes and new edges can be computed [39], shown as

$$t = \frac{l_e}{v} \tag{8}$$

where l_e is the length of extended trajectory segment. v is the longitudinal velocity of the vehicle. Resulting time t as a new dimension is added into each state of the trajectory. Likewise, the arrival time t will also be added into the generated rule

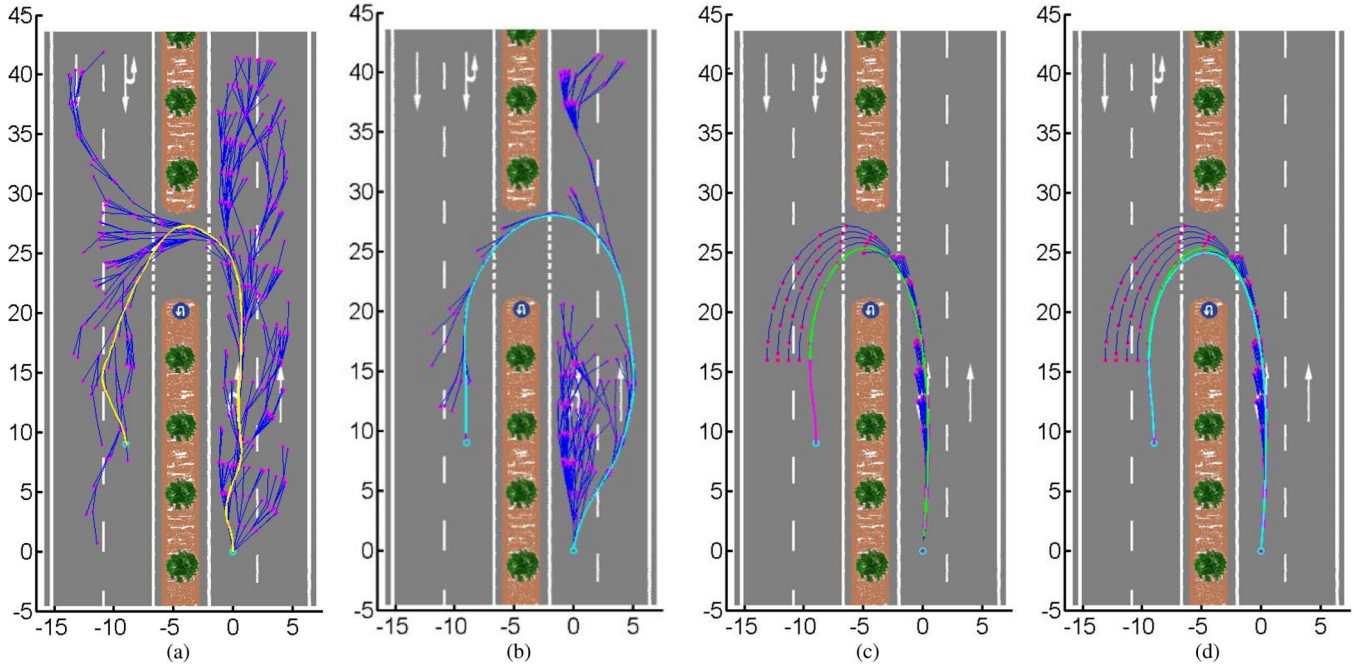


Fig. 8. Comparison among different methods in U-turn maneuver. (a) Result of RRT. (b) Result of CL-RRT (c) First phase result of our method. (d) Second phase result of our method.

template, as shown in Fig. 6. In this figure, red volume denotes the dynamic obstacle moving in duration [0, 5s]. Blue curves represent a 3-D tree in configuration-time space, in which the z-axis denotes the arrival time. In the plane of the same arrival time, collision between configuration of the vehicle and the section of the obstacle is detected. This is similar to the collision detection of motion planning in 2-D environments.

When the search tree reaches a vertical line that is perpendicular to the XY plane and through the goal, the planning in configuration-time space will be accomplished. So far, a 3-D trajectory, including the arrival time, can be generated. Meanwhile, a 2-D trajectory can also be obtained from its projection onto the XY plane. The proposed algorithm can be simply applied to this configuration-time space, and the current speeds of obstacles are used to predict their future states. In this way, the planned trajectory is more reasonable and feasible, which could reduce the computational overhead of replanning and make the whole trajectory smoother. Therefore, the integration of the proposed algorithm into the configuration-time space is the guarantee of good replanning.

VI. Experimental Results and Discussion

To verify the performance of the proposed approach, experiments are conducted under two different types of environments, namely, urban environments with and without obstacles. Under urban environments without obstacles, various maneuvers are taken into account. In urban environments with obstacles, the scenes with a single obstacle and multiple obstacles are set in the test, respectively. The proposed approach is compared with two other related approaches, namely, standard RRT algorithm [13], or RRT algorithm with 5% goal bias, which is abbreviated as RRT-gb, and closed-loop RRT (CL-RRT) algorithm [30] without the optimization heuristics for updating the resulting trajectory [29], because we hope that the performance of the algorithm in a successful planning cycle can be tested. In these tests, the stop conditions of the RRT, RRT-gb, and CL-RRT are that the maximum number of the iterations is reached or the search tree reaches a region close to the goal. The maximum number of the iteration is set as 3000. The region is a circle with a radius equal to 1 m, centered at the goal. Note that the heading error is not considered in the stop conditions. The stop condition of the proposed algorithm is that the tree accurately reaches the goal state by the aggressive extension strategy. Furthermore, to address the problem of dynamic environments, some results with respect to our algorithm planning in the configuration-time space are also presented. All utilized approaches were implemented in MATLAB on a laptop computer with 2.67-GHz CPU and 3.8-GB RAM. In comparison illustrations (see Figs. 7–11), the first phase and second phase results are all presented. The first phase result is generated by the fast RRT algorithm, and the second phase result is produced by its postprocess approach. The purpose of presentation in two separated phases is to synchronously show the components of the generated trajectory and final results. However, the results of two phases are always highly overlapping. This is because the fast RRT algorithm takes into account the model of vehicle in trajectory generation. In all experiments shown in the following, the cadet blue lines and the magenta points denote the edges and nodes of the search tree, respectively. Yellow lines denote the trajectories generated by searching. Magenta lines indicate the paths generated by the aggressive extension strategy. Cyan lines are the resulting trajectories by forward simulation, which could be the result of CL-RRT or the result of the proposed algorithm in the second phase.

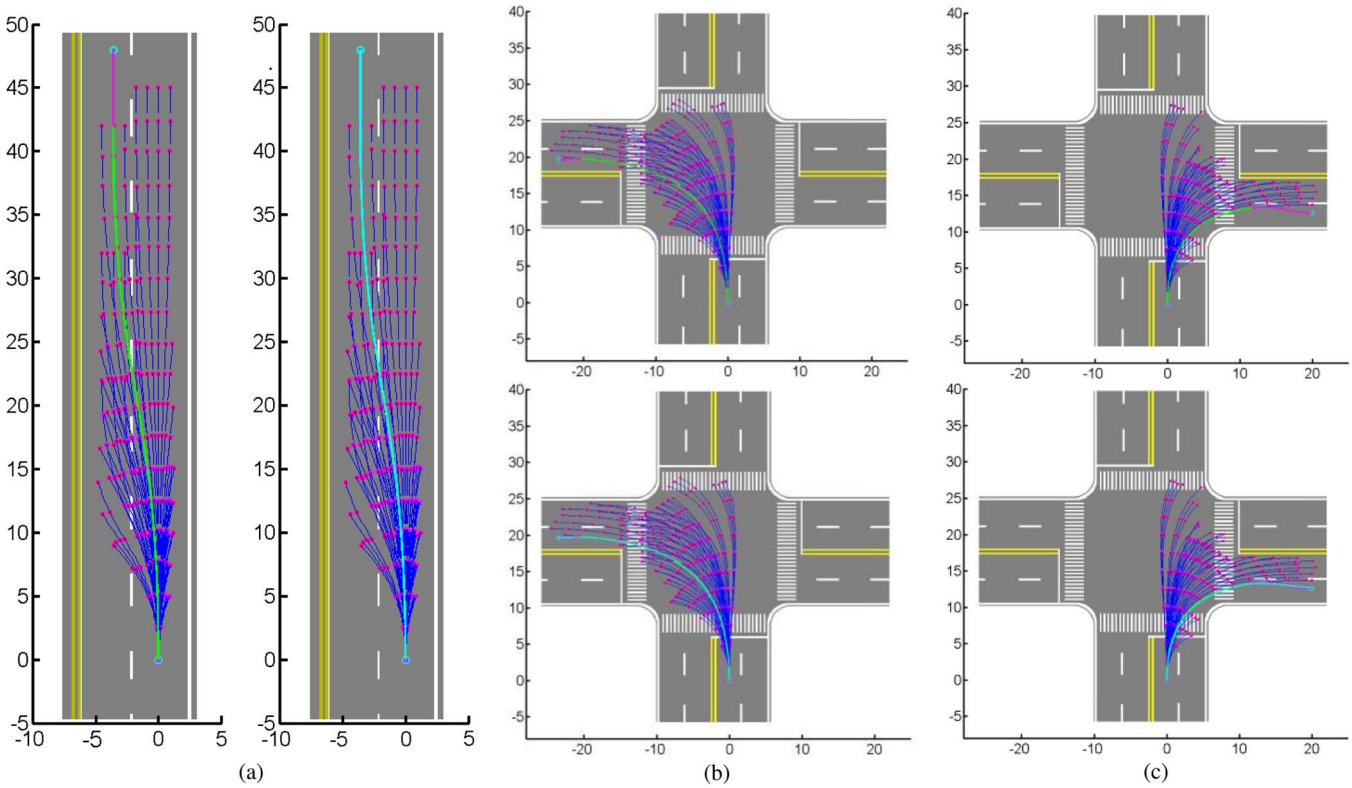


Fig. 9. Planning results of the proposed method in other maneuvers (the first phase and second phase results). (a) Results in lane change maneuver. (b) Results in turn left maneuver. (c) Results in turn right maneuver.

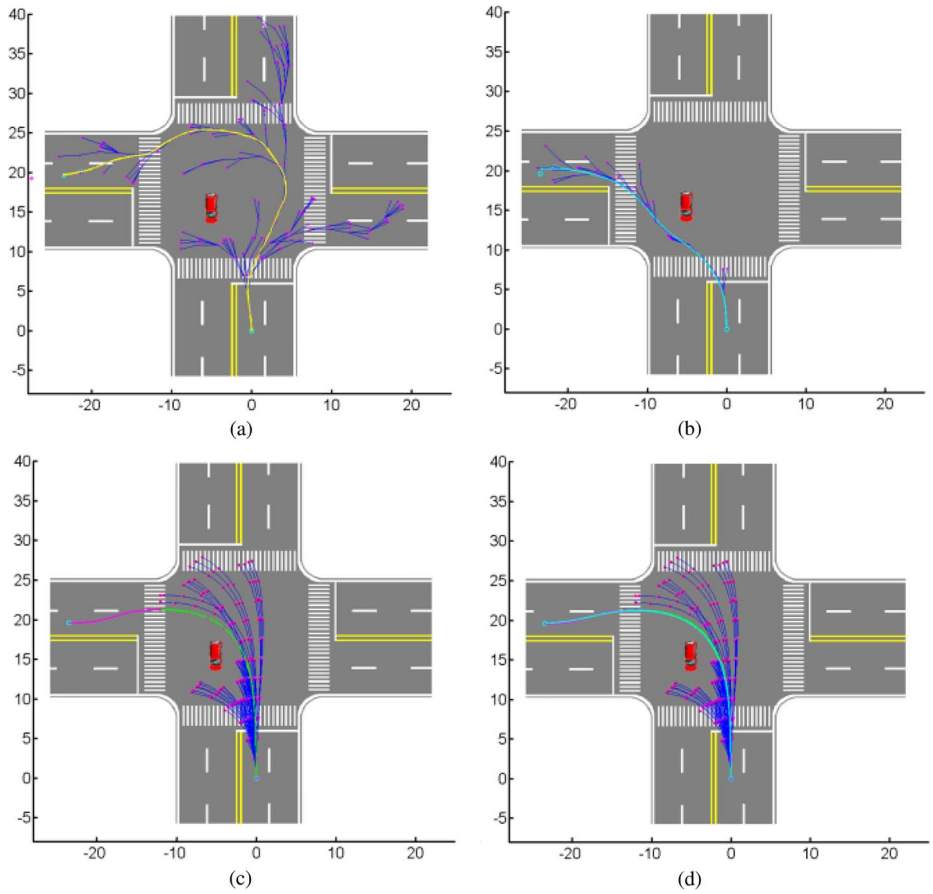


Fig. 10. Comparison among different methods in turning left maneuver at the intersection with one obstacle. (a) Result of RRT. (b) Result of CL-RRT. (c) First phase result of our method. (d) Second phase result of our method.

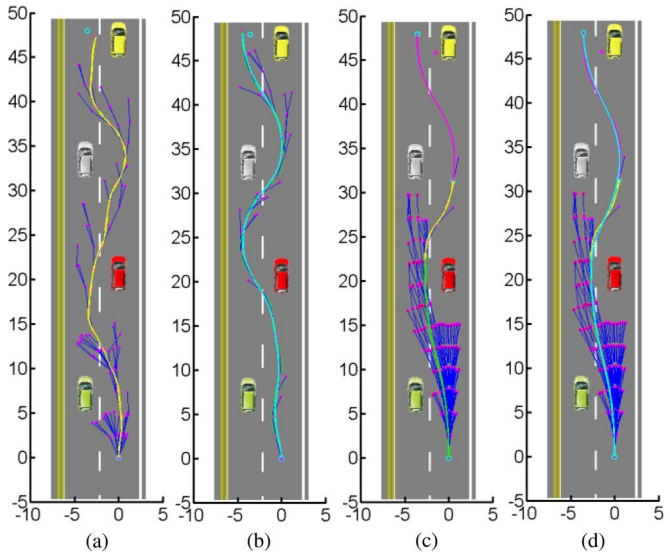


Fig. 11. Comparison among different methods in overtaking maneuver under the cluttered traffic environment with multiple obstacles. (a) Result of RRT. (b) Result of CL-RRT. (c) First phase result of our method. (d) Second phase result of our method.

A. Environments Without Obstacles

In the urban environments without obstacles, five common maneuvers, namely, on-road lane keeping, on-road lane change, turn left at the intersection, turn right at the intersection, and U-turn, are used for test. The comparison results of the proposed algorithm and the related algorithms for on-road lane keeping and U-turn are shown in Figs. 7 and 8, respectively. The planning results of the proposed algorithm for the other three maneuvers are also given in Fig. 9.

Lane keeping on current lane without obstacles is the most common driving behavior on road. Fig. 7(a) presents the result of the RRT algorithm, which obviously shows great meander and terminal state error. Fig. 7(b) shows the result of CL-RRT. Although the meander is lessened, there still exists the terminal state error. Fig. 7(c) and (d) show the results of the proposed algorithm in two phases, respectively. It shows that, for lane keeping, the proposed approach can plan a trajectory that approximates a straight line and reaches the terminal state accurately.

U-turn is a challenging maneuver for motion planning algorithms. Standard RRT algorithms often traverse whole configuration space to accomplish the turning, as shown in Fig. 8(a). It is difficult for selecting appropriate Gaussian sampling clouds to bias the tree to the terminal state. On the contrary, it causes many collisions before turning, as shown in Fig. 8(b). The proposed algorithm can easily achieve the U-turn using the rule template generated offline and the trajectory segment generated by the aggressive extension strategy, as shown in Fig. 8(c). Model-based forward simulation approach can make the trajectory at the place of U-turn smoother, as shown in Fig. 8(d).

The test results of different algorithms for five maneuvers are given in Table I. To eliminate randomness, each algorithm was tested for 3000 Monte Carlo (MC) trials. Table I presents the averages of the number of samples, the number of nodes in the tree, the success rate, the trajectory length, and the run time. The number of samples and available nodes can reflect the efficiency and relative speed of these algorithms. Note that the nodes in the rule template are not included in the results, because they are generated offline. It shows that each algorithm, particularly the CL-RRT algorithm, has good success rate under the obstacle-free environments, except for the U-turn maneuver. Compared with the proposed algorithm, the related algorithms spend more runtime and need more sampling numbers, because they find the trajectory by means of searching alone. For these five basic maneuvers, the proposed algorithm can achieve a smooth trajectory only by combining the rule templates and the trajectory generated by aggressive extension strategy, without any searching. Therefore, the runtime is very short, which is less than 3.5 ms. Furthermore, with respect to the trajectory length for the five maneuvers, the proposed algorithm generates the shortest trajectory with less meander and winding, compared with other related algorithms.

B. Environments With Obstacles

Under the environments with obstacles, two types of cases are considered, namely, with a single obstacle and with multiple obstacles (four obstacles are set). For the comprehensive evaluation, two classical traffic scenes, namely, "straight road" and "intersection," are taken into account, respectively. Similarly, two typical illustrations are given. One is for only one obstacle at the intersection, as shown in Fig 10. The other one is for avoiding multiple obstacles on a straight road, as shown in Fig 11. Additionally, the results of the proposed algorithm under the environment with traffic cones, a kind of special obstacle, are also presented.

Fig. 10 shows that the vehicle turns left and bypasses a single obstacle at the intersection. Fig. 10(a) shows the result of the standard RRT algorithm under this environment. The uniform distribution of sampling makes the search tree need a large number of attempts to accomplish turning left and bypassing the obstacle. CL-RRT obtains a better result due to its sampling distribution. However, there still exist minor meanders and terminal state error, as shown in Fig. 10(b). The proposed algorithm can easily bypass the obstacle and accomplish the turn left, as shown in Fig. 10(c) and (d). By comparing two results of CL-RRT and our algorithm, we can see that the trajectory planned by CL-RRT is too close to the edge of the intersection and the available lanes, whereas the trajectory planned by our algorithm, due to the constraint of the rule template, is around the center of the intersection, which is more like a real path by a human driver. A great number of experiments under environments with a single obstacle show that the proposed algorithm can efficiently accomplish the planning only by the rule template and the aggressive extension strategy without using any search.

Fig. 11 shows the overtaking on a two-lane road with multiple obstacles, which is more complicated than the single-obstacle environment. Fig. 11(a) and (b) are the results of the RRT and CL-RRT algorithms, respectively. Meander trajectory and terminal state error are still their weaknesses, although CL-RRT is relatively smooth and avoids some meaningless

TABLE I
PERFORMANCE COMPARISON AMONG DIFFERENT APPROACHES IN VARIOUS MANEUVERS

		RRT	RRT-gb	CL-RRT	Ours
Lane keeping	***Avg. of samples numbers***	349.5	206.9	138.3	0
	Avg. of nodes numbers	155.5	104.6	67.8	2
	Avg. of Suc. rate (%)	100	100	100	100
	Avg. of path length (m)	50.99	50.58	50.09	49.00
	Avg. of run time (ms)	207.5	113.7	72.6	1.7
Change lane	***Avg. of samples numbers***	548.8	256.4	97.8	0
	Avg. of nodes numbers	217.4	122.2	59.6	3
	Avg. of Suc. rate (%)	99	100	100	100
	Avg. of path length (m)	50.87	50.79	50.62	49.22
	Avg. of run time (ms)	330.8	128.1	67.6	2.1
Turn left	***Avg. of samples numbers***	898.8	235.8	432.5	0
	Avg. of nodes numbers	305.8	91.9	114.7	4
	Avg. of Suc. rate (%)	93	85.20	99.67	100
	Avg. of path length (m)	43.03	41.65	37.68	37.12
	Avg. of run time (ms)	676.3	643.2	293.1	3.1
Turn right	***Avg. of samples numbers***	901.7	819.4	235.8	0
	Avg. of nodes numbers	306.4	265.6	91.9	4
	Avg. of Suc. rate (%)	94.33	96	100	100
	Avg. of path length (m)	32.61	32.99	31.51	29.70
	Avg. of run time (ms)	678.8	583.2	150.9	3.3
U-turn	***Avg. of samples numbers***	790.6	859.8	623.8	0
	Avg. of nodes numbers	275.5	283.5	182.6	3
	Avg. of Suc. rate (%)	93.67	98.67	65	100
	Avg. of path length (m)	52.93	52.08	54.52	51.20
	Avg. of run time (ms)	500.9	534.4	382.3	2.0

extensions. Fig. 11(c) shows the first phase result of the proposed algorithm. It shows that, in this complex case, the algorithm cannot directly plan the path only by the rule template (green line) and the aggressive extension strategy (magenta line). Some search branches (yellow line) can supplement the gap between them, which is also one of the important advantages of the proposed approach. Fig. 11(d) shows the second phase result of the proposed algorithm, in which a smooth trajectory (cyan line) is generated. It is obviously seen that some portions of the path generated in the first phase are further smoothed.

Table II presents the test results under the environment with obstacles in various cases. They are still the averages of 3000 MC trials results like in Table I. It is shown that each algorithm has a good result on the straight road, no matter with a single obstacle or multiple obstacles. Due to the large configuration space of the intersection environment, all of the related algorithms have lower success rates, particularly with multiple obstacles. On the contrary, the proposed algorithm still keeps very high success rates. Because of a large number of search and collision detection, the related algorithms all spend long runtimes and need many samples. The results of our algorithm are completely different. Under an environment with a single obstacle, our algorithm can accomplish the planning without searching. Under a multiple-obstacle environment, it only needs a small number of searches to accomplish the planning. Under a single-obstacle environment, the runtime of the proposed algorithm is less than 2.5 ms. Even in the multiple-obstacle environment, the runtime is also less than 80 ms. Moreover, the length of the trajectory planned by the proposed algorithm is shorter than those planned by the related algorithms in most cases, which indicates that a relatively optimal solution can be achieved by our algorithm. Only under the environment with one obstacle at the intersection is the average trajectory length of the proposed algorithm larger than that of CL-RRT, which has been presented and analyzed above.

Furthermore, traffic cone is a special kind of obstacle because it has both properties of hindrance as a general obstacle and guidance as a traffic sign. Because of road maintenance or other reasons, traffic cones are often used on road or at intersection to guide the vehicles redirection, which is also an important testing part in the Chinese intelligent vehicle competition. The proposed algorithm can accomplish the planning task easily in the traffic scenes, as shown in Fig. 12. Obviously, the rule template and the aggressive extension strategy play important roles in improving the planning efficiency, when the vehicle comes into and departs the narrow passage formed by traffic cones.

The aforementioned experimental results show that the proposed algorithm is a faster and more efficient RRT variant, compared with other related algorithms. Rather than generating the trajectory only by searching, like what the conventional RRT algorithm does, the proposed algorithm generates the trajectory with three mechanisms, namely, the rule template, the rushing from the aggressive extension strategy, and the searching. The first two mechanisms greatly reduce the time consumption. Therefore, the proposed algorithm can generate a trajectory rapidly. By analyzing the breakdown of planned

TABLE II
PERFORMANCE COMPARISON AMONG DIFFERENT APPROACHES IN THE ENVIRONMENTS WITH OBSTACLES

		RRT	RRT-gb	CL-RRT	Ours
With a single Obs. on the straight road	***Avg. of samples numbers***	353.7	215.2	156.2	0
	Avg. of nodes numbers	141.1	99.3	55.9	7
	Avg. of Suc. rate (%)	99.67	100	99	100
	Avg. of path length (m)	50.38	50.11	50.11	48.85
	Avg. of run time (ms)	211.9	127.2	87.5	2.3
With a single Obs. at the intersection	***Avg. of samples numbers***	908.3	1008.1	496.4	0
	Avg. of nodes numbers	322.4	305.1	106.3	5
	Avg. of Suc. rate (%)	86.33	76	98.33	100
	Avg. of path length (m)	42.02	42.79	36.64	40.17
	Avg. of run time (ms)	796.1	823.1	378.8	2.5
With multi-Obs. on the straight road	***Avg. of samples numbers***	242.3	201.5	126.4	55.8
	Avg. of nodes numbers	88.5	76.2	43.0	12.9
	Avg. of Suc. rate (%)	100	100	100	100
	Avg. of path length (m)	50.27	50.16	50.16	49.66
	Avg. of run time (ms)	171.2	135.7	89.8	46.5
With multi-Obs. at the intersection	***Avg. of samples numbers***	1444.2	1353.1	1119.9	82.5
	Avg. of nodes numbers	441.5	314.8	115.8	13.9
	Avg. of Suc. rate (%)	38.33	56	40.33	100
	Avg. of path length (m)	42.78	42.99	44.26	39.72
	Avg. of run time (ms)	1410.5	1192.3	750.4	79.8

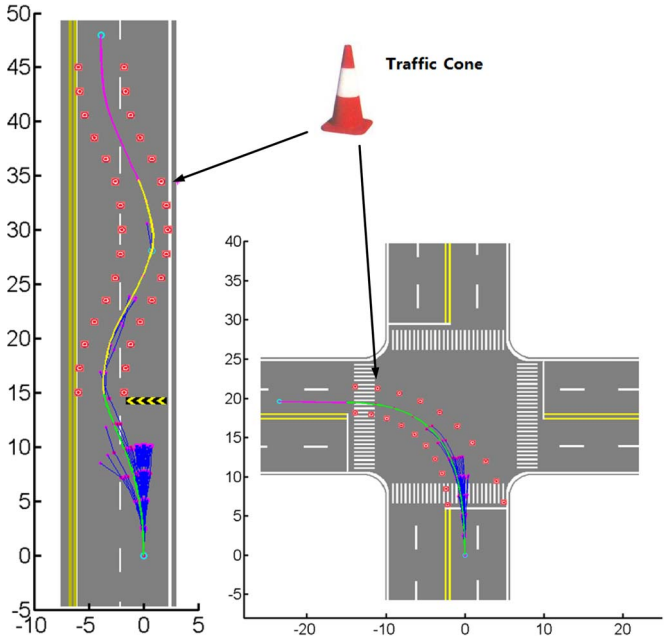


Fig. 12. Planning results of the proposed method for traffic redirection by cones used on road and at intersection.

trajectory length, we find that, under environments with a few or without obstacles, the proposed algorithm can accomplish the trajectory generation only by the rule template and the aggressive extension strategy, without any searching. Under multiple-obstacle environment, the proposed algorithm does not only take the advantage of these two ways but also bridge the gap between them by searching, as shown in Fig. 13. In summary, the proposed algorithm does not only keep the characteristic of searching but also becomes more efficient than the conventional RRT. The use of the rule template can make the generated trajectory more in accordance with the geometric structural information of the roads.

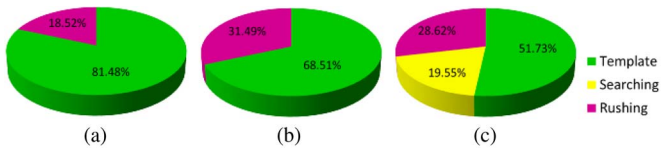


Fig. 13. Breakdown of planned trajectory length with different numbers of obstacles. (a) Without obstacles. (b) With an obstacle. (c) With multiple obstacles.

C. Test in the Configuration-Time Space

We also test the integrated method of the proposed algorithm and the configuration-time space, which is to deal with the dynamic obstacles and to improve the quality of the replanning. A typical experimental result is given in Fig. 14. In this experiment, on a two-lane road, two vehicles are driving with low speed. We hope that a feasible trajectory for the autonomous vehicle is planned to overtake these two vehicles from the space between them. In Fig. 14(a), it is shown that two obstacles are extruded according to their velocities, which is marked as red volume. A 3-D trajectory is calculated in this 3-D space, which has no collision with the volume. Fig. 14(b) shows the resulting trajectory projected on the 2-D plane. For comparison, the planning result without considering the dynamic obstacles is also given (black line). It shows great differences in the shapes of the two trajectories. The result without considering the dynamic obstacles will inevitably be replanned, triggered by the coming collision with the vehicles ahead. However, the planning result in the configuration-time space is more reasonable and does not need to trigger the replanning for a short time, if there is no sudden change in the velocities of those vehicles.

VII. CONCLUSION

This paper has presented an efficient sampling-based motion planning method designed for on-road driving of autonomous

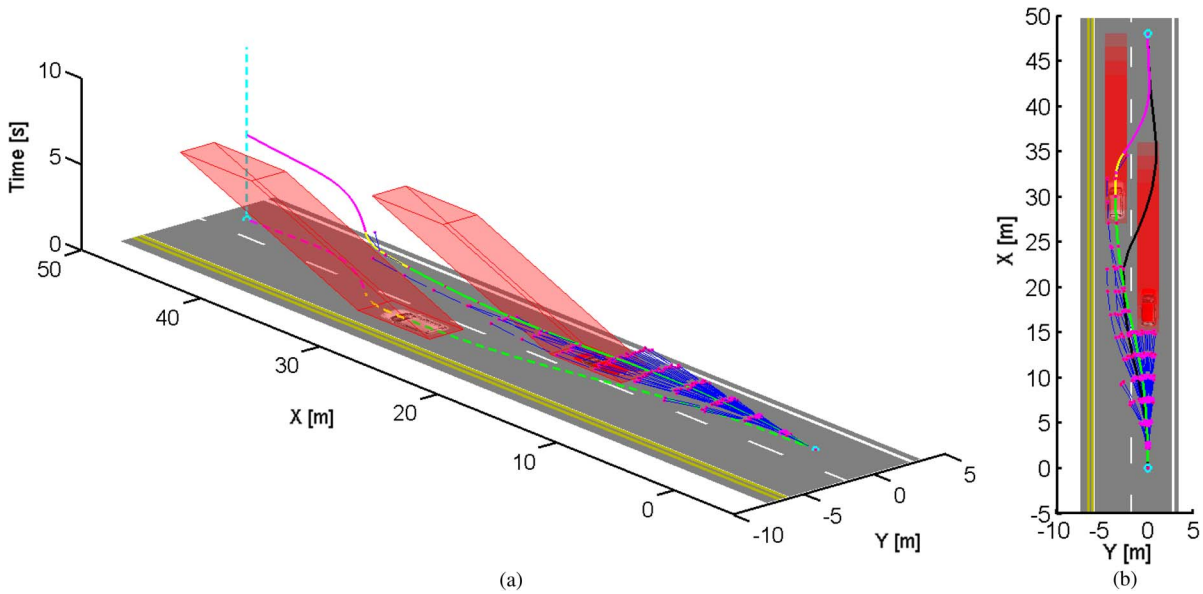


Fig. 14. Result of our algorithm in the configuration-time space (the black line in 2-D view indicates the planning result when without considering the dynamic obstacles). (a) 3-D view. (b) 2-D view.

vehicles. The core of this method is a fast RRT variant, which introduces a rule-template set based on the traffic scenes and an aggressive extension strategy of search tree, and combines a prior closed-loop prediction approach. For improving the re-planning in the dynamic environments, an integrated approach of the fast RRT and the configuration-time space is proposed and used.

A large number of statistical experiments were conducted for comparing the performance of our approach against other related approaches in various scenes. Experimental results demonstrate that the proposed approach can plan a trajectory as follows: 1) faster with a lower density of obstacles; 2) more accurate toward goal state; and 3) shorter and with less meanders. By the analysis, the introduction of the rule template can improve the efficiency of the RRT algorithm in the early stage and can make the generated trajectory more consistent with the geometric structural information on the roads, while the use of the aggressive extension strategy can raise the efficiency of the algorithm in the final phase and guarantee the accuracy of the terminal state. In some simple environments, the proposed algorithm can accomplish the trajectory planning only by these two improvements without any searching. Under the cluttered environments, the algorithm can still keep the search ability of the traditional RRT and accomplish the planning efficiently by the integration of three options, namely, the rule template, the rushing by the extension strategy, and the searching. The rule-template technique perhaps is only suitable for the application of autonomous vehicles, whereas the aggressive extension strategy can be generalized to various application fields. The forward simulation postprocess can make the resulting trajectory smoother. The proposed approach integrated into the configuration-time space can plan more reasonable and reliable trajectory, addressing the dynamic obstacles.

In future work, we will attempt to find an asymptotic optimal path based on the fast RRT variant. As the proposed algorithm is more efficient than the traditional RRT algorithms, it is quite possible to refine the trajectories, like what RRT* does.

REFERENCES

[1] L. Figueiredo, I. Jesus, J. T. Machado, J. Ferreira, and J. M. Carvalho, "Towards the development of intelligent transportation systems," in *Proc. 4th IEEE Int. Conf. Intell. Transp. Syst.*, Oakland, CA, USA, 2001, vol. 88, pp. 1206–1211.

[2] H. Cheng, *Autonomous Intelligent Vehicles: Theory, Algorithms, and Implementation*. New York, NY, USA: Springer-Verlag, 2011, pp. 3–5.

[3] T. M. Howard, M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Model-predictive motion planning," *IEEE Robot. Autom. Mag.*, vol. 21, no. 1, pp. 64–73, Mar. 2014.

[4] J. C. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts," *Int. J. Robot. Res.*, vol. 18, no. 11, pp. 1119–1128, Nov. 1999.

[5] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.

[6] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[7] A. Stentz, "Optimal and efficient path planning for unknown and dynamic environments," *Int. J. Robot. Autom.*, vol. 10, no. 3, pp. 89–100, Aug. 1995.

[8] S. Koenig and M. Likhachev, "D* lite," in *Proc. 8th Nat. Conf. AAAI*, 2002, pp. 476–483.

[9] J. Barraquand, B. Langlois, and J. C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Trans. Syst., Man Cybern.*, vol. 22, no. 2, pp. 224–241, Mar./Apr. 1992.

[10] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[11] S. M. LaValle, *Planning Algorithms*. Cambridge, U. K.: Cambridge Univ. Presss, 2006.

[12] S. M. LaValle, "Rapidly-Exploring random trees: A new tool for path planning," Comput. Sci. Dept. , Iowa State Univ., Ames, IA, USA, Tech. Rep. 98-11, 1998.

[13] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[14] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, San Francisco, CA, USA, 2000, vol. 2, pp. 995–1001.

[15] H. Long, Q. H. Do, and S. Mita, "Unified path planner for parking an autonomous vehicle based on RRT," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, 2011, pp. 5622–5627.
[16] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Zurich, Switzerland, 2002, vol. 3, pp. 2383–2388.
[17] C. Urmson and R. G. Simmons, "Approaches for heuristically biasing RRT growth," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Las Vegas, NV, USA, 2003, pp. 1178–1183.
[18] S. Rodriguez, X. Tang, J. M. Lien, and N. M. Amato, "An obstacle-based rapidly-exploring random tree," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, USA, 2006, pp. 895–900.
[19] J. Denny, M. Morales, S. Rodriguez, and N. M. Amato, "Adapting RRT growth for heterogeneous environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Tokyo, Japan, 2013, pp. 1772–1778.
[20] N. A. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *Proc. IEEE Int. Conf. Robot. Autom.*, Rome, Italy, 2007, pp. 1617–1624.
[21] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.
[22] J. H. Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *Proc. 50th IEEE Conf. Decision Control Eur. Control*, Orlando, FL, USA, 2011, pp. 3276–3282.
[23] G. Goretkin, A. Perez, R. J. Platt, and G. Konidaris, "Optimal sampling-based planning for linear-quadratic kinodynamic systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, 2013, pp. 2429–2436.
[24] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, 2013, pp. 5041–5047.
[25] D. Kim, J. Lee, and S. Yoon, "Cloud RRT*: Sampling cloud based RRT*," in *Proc. IEEE Int. Conf. Robot. Autom.*, Hong Kong, 2014, pp. 2519–2526.
[26] M. Likhachev, D. Ferguson, G. Gordon, A. T. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm," in *Proc. ICAPS*, 2005, pp. 1–10.
[27] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *J. Field Robot.*, vol. 25, no. 11, pp. 939–960, 2008.
[28] M. Montemerlo *et al.*, "Junior: The Stanford entry in the urban challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, 2008.
[29] Y. Kuwata *et al.*, "Real-time motion planning with applications to autonomous urban driving," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009.
[30] Y. Kuwata *et al.*, "Motion planning in complex environments using closed-loop prediction," in *Proc. AIAA Guid., Navigat., Control Conf.*, Honolulu, Hawaii, 2008, pp. 1–22.
[31] K. Chu, M. Lee, and M. Sunwoo, "Local path planning for off-road autonomous driving with avoidance of static obstacles," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1599–1616, Dec. 2012.
[32] J. Xin, C. Wang, Z. Zhang, and N. Zheng, "China future challenge: Beyond the intelligent vehicle," *IEEE Intell. Transp. Syst. Soc. Newslett.*, vol. 16, no. 2, pp. 8–10, Apr. 2014.
[33] M. Du *et al.*, "An improved RRT-based motion planner for autonomous vehicle in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, Hong Kong, 2014, pp. 4674–4679.
[34] L. Ma *et al.*, "A fast RRT algorithm for motion planning of autonomous road vehicles," in *Proc. 17th IEEE Int. Conf. Intell. Transp. Syst.*, Qingdao, China, 2014, pp. 1033–1038.
[35] K. Kawabata, L. Ma, J. R. Xue, and N. N. Zheng, "A path generation method for automated vehicles based on Bezier curve," in *Proc. IEEE/ASME Int. Conf. Adva. Intell. Mechatron.*, Wollongong, NSW, Australia, 2013, pp. 991–996.
[36] S. Thrun *et al.*, "Stanley: The robot that won the DARPA grand challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, Sep. 2006.
[37] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, Fl, USA, 2006, pp. 1243–1248.
[38] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite RRTs for rapid replanning in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, Rome, Italy, 2007, pp. 1603–1609.
[39] I. Ardiyanto and J. Miura, "3D time–space path planning algorithm in dynamic environment utilizing arrival time field and heuristically randomized tree," in *Proc. IEEE Int. Conf. Robot. Autom.*, Rome, Italy, 2012, pp. 187–192.

Liang Ma received the B.S. degree in electronic and information engineering and the M.S. degree in communication and information system from China University of Geosciences, Wuhan, China, in 2005 and 2008, respectively. He is currently working toward the Ph.D. degree with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, China.

From 2008 to 2009, he was a Research Staff with the Xi'an Institute of Optic and Precision Mechanics, Chinese Academy of Sciences, Xi'an. His research interests include motion planning, autonomous navigation, and computer vision.

Jianru Xue (M'06) received the B.S. degree from Xi'an University of Technology, Xi'an, China, in 1994 and the M.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, in 1999 and 2003, respectively.

From 2002 to 2003, he was with Fuji Xerox, Tokyo, Japan. In 2008 to 2009, he visited the University of California, Los Angeles, CA, USA. He is currently a Professor with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University. His research field includes computer vision, visual navigation, and video coding based on analysis.

Prof. Xue served as a Coorganization Chair for the Asian Conference on Computer Vision 2009 and the International Conference on Virtual System and Multimedia 2006. He also served as a Program Committee Member for the International Conference on Pattern Recognition 2012, the Asian Conference on Computer Vision 2010 and 2012, and the International Conference on Multimedia and Expo 2014.

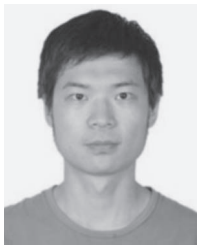
Kuniaki Kawabata (M'93) received the B.E., M.E., and Ph.D. degrees from Hosei University, Tokyo, Japan, in 1992, 1994, and 1997, respectively, all in electrical engineering.

From 1997 to 2000, he was a Special Postdoctoral Researcher with the Biochemical Systems Laboratory, The Institute of Physical and Chemical Research (RIKEN), Wako, Japan, where he is currently the Unit Leader of the RIKEN-XJTU Joint Research Unit. He is also currently a Visiting Researcher with the Research into Artifacts, Center for Engineering, The University of Tokyo, Tokyo; a Visiting Professor with Kagawa University, Takamatsu, Japan; and a Project Associate Professor with Keio University, Tokyo. His research interests include mobile robotics, distributed autonomous systems, networked system, and understanding social adaptability of the insect.

Dr. Kawabata is a member of the Robotics Society of Japan, the Society of Instrument and Control Engineers, and the Japanese Society of Mechanical Engineers.

Jihua Zhu received the B.E. degree in automation from Central South University, Changsha, China, in 2004 and the Ph.D. degree in control science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2011.

He is currently an Associate Professor with the School of Software Engineering, Xi'an Jiaotong University. His research interests include computer vision, autonomous robots, and image processing.



Chao Ma received the B.S. degree in automatic control engineering from Xi'an Jiaotong University, Xi'an, China, in 2010. He is currently working toward the Ph.D. degree in the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University.

His research interests include motion planning and control.

Nanning Zheng (SM'94–F'06) received the Bachelor's degree and M.E. degree in information and control engineering from Xi'an Jiaotong University, Xi'an, China, in 1975 and 1981, respectively, and the Ph.D. degree in electrical engineering from Keio University, Tokyo, Japan, in 1985.

He is currently a Professor and the Director of the Institute of Artificial Intelligence and Robotics with Xi'an Jiaotong University. His research interests include computer vision, pattern recognition, computational intelligence, image processing, and hardware implementation of intelligent systems.

Prof. Zheng became a member of the Chinese Academy of Engineering in 1999. Since 2000, he has been the Chinese Representative on the Governing Board of the International Association for Pattern Recognition. He currently serves as an Executive Editor of the Chinese Science Bulletin.