2014 IEEE 17th International Conference on
Intelligent Transportation Systems (ITSC)
October 8-11, 2014. Qingdao, China

A Fast RRT Algorithm for Motion Planning of Autonomous Road Vehicles*

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

***Abstract*—The Rapidly-exploring Random Tree (RRT) is a classical algorithm of motion planning based on incremental sampling, which is widely used to solve the planning problem of mobile robots. But it, due to the meandering path, the inaccurate terminal state and the slow exploration, is often inefficient in many applications such as autonomous road vehicles. To address these issues and considering the realistic context of autonomous road vehicles, this paper proposes a fast RRT algorithm that introduces an off-line template set based on the traffic scenes and an aggressive extension strategy of search tree. Both improvements can lead to a faster and more accurate RRT towards the goal. Meanwhile, our approach combines the closed-loop prediction approach using the model of vehicle, which can smooth the portion of off-line template and the portion of on-line tree generated, while a trajectory and control sequence for the vehicle would be obtained. Experimental results illustrate that our method is fast and efficient in solving planning queries of autonomous road vehicle in urban environments.**

I. Introduction

As one of six Intelligent Transportation System (ITS) major categories [1], autonomous road vehicles were originally developed in 1980s. Nowadays Google driverless car has been allowed testing and self-driving on Michigan's roads according to the legislation. With the development of the technology with respect to autonomous road vehicles, it 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 the persons with handicaps, logistics for delivery service are typical significant applications.

Motion planning of autonomous road vehicles is an important and fundamental technology [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 towards a desired terminal state, taking into account vehicle's kinematic and dynamic model [3], as shown in Fig. 1.

Motion planning problem has been widely studied in the field of mobile robot, and many motion planning approaches were presented in the robotics literature over the past three decades [4]. Some well-known planning algorithms such as Dijkstra, A*, D* and Potential Field method can plan paths and avoid the obstacles, but it is difficult to consider the complex dynamic and differential constraints of the vehicle. Recently, sampling-based techniques for robot motion planning have become more popular and widespread. Rapidly-exploring Random Tree (RRT) is a main algorithm based on incremental sampling, which was first proposed by Steven in a technical report [5]. 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. Hence, RRT was widely used for motion planning of mobile robots and manipulations. Nevertheless, there is still a large gap between basic RRT and those applications. Many RRT variants were proposed for advancing it. RRT-Connect, a bi-directional version, synchronously search with two trees from start point and goal point for improving efficiency [6]. This RRT variant was applied to parking of intelligent vehicles [7], 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 (ERRT) biases search towards goals and waypoints [8], and [9] introduced a heuristic function for biasing tree growth. These bias techniques are very efficient and used in a wide range of applications. [10] presented an obstacle-based RRT (OBRRT) 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, [11] introduced the machine leaning method into selecting of growth methods. Particle RRT [12] casted the particle filter into the RRT's framework, 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 and not a real-time autonomous road vehicle.

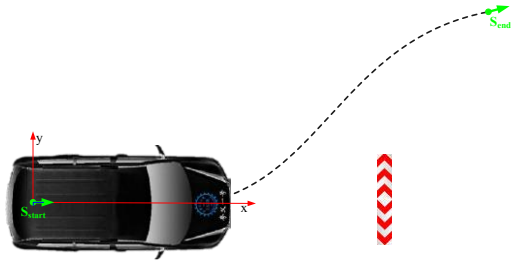


Fig. 1. The motion planning of the autonomous road vehicle

* Research is partially supported by NSFC project under No. 91320301 and No. 61203326, and by the Fundamental Research Funds for the Central Universities under No. xjj2012089.

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, Shaanxi, 710049, China (phone: +86 (0)158 7744 3777; e-mail: mlxjtu@gmail.com).

K. Kawabata is with RIKEN (The Physical and Chemical Research), Hirosawa, Wako, Saitama, 3510198, Japan (e-mail: kuniakik@riken.jp).

Recently, Karaman presented an optimal RRT (RRT*) that can get an asymptotic optimal path which the RRT algorithm lacked [13] and is applied to off-road vehicle maneuvers [14]. The asymptotic optimal solution is obtained at the cost of consuming much time by a rewire process. RRT improved as a real-time, on-line motion planning algorithm was used in the MIT's autonomous road vehicle [15], where an important extension was the use of closed-loop prediction model [16].

However, there exist some weaknesses in the motion planning of autonomous road vehicle using RRT algorithm.

- The path planned by RRT is often jagged and meandering due to the growth way of the tree. Although lots of work were presented for smoothing the path planned by RRT, lessened meanders might still exist in connects of two edges. For instance, lane keeping is the most common maneuvers for vehicles on road when without obstacles. Using the RRT is 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 towards the sample configuration or 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. There will be a risk, if the vehicle cannot reach the desired goal state.
- An important feature of autonomous road vehicles is real-time while the efficiency of RRT depends mainly on whether sampling domain well adapts to the problem or not. To address this issue, [15] used the Gaussian distributions over sampling domain, whose related parameters were then determined according to the traffic scenes. But, for on-road environments, the efficiency of RRT algorithm can be further improved?

To overcome these shortages presented above, this paper focuses on improving the RRT algorithm to solve the motion planning problem of autonomous road vehicle. Based on previous work [15], a fast RRT algorithm is proposed, which introduces an off-line template set in terms of the traffic scenes and an aggressive extension strategy of search tree. Meanwhile, the closed-loop prediction approach via the model of vehicle is adopted for smoothing and computing the trajectory and control sequence for the vehicle. The proposed approach is faster, and its planned trajectory can accurately reach the desired configuration. The meanders of trajectory can also be eliminated in most cases.

II. Problem Formulation

A. Motion Planning Definition

The motion planning problem is always formulated as following. The state space and 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:

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

where, $x(t) \in X$, $u(t) \in U$, and x_0 is the initial state. And 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 $X_{free} \subset X \setminus X_{Obs}$. The motion planning task is to compute:

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

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. Autonomous Road Vehicles Model

Fig. 2. KuaFu-1: XJTU's autonomous vehicle prototype

A front-wheel steering Ackerman vehicle model is considered, because it is suitable for our autonomous vehicle prototype KuaFu-1, which is developed for the competition of Future Challenge of Intelligent Vehicles in China [17], as shown in Fig. 2. Since the main task is to realize self-drive on road in urban environments, we call it as autonomous road vehicle and the motion planning problem is considered on a 2D plat. The kinematic model is described by

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

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

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

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

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

III. Algorithm Description

In this section, we propose a fast RRT algorithm for autonomous road vehicle, and the algorithm flow is described in detail. We first give a review on the operation of the basic RRT algorithm, and then present our 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) A random sample, x_{rand}, is drawn from the state space over a specified sampling distribution, $p_{rand}(x)$. 2) Find the nearest node in the tree to the sample using a designated distance metric, $\rho(x)$. 3) The selected nearest node expands towards the random sample, when a control input is determined for forward simulation. The resulting state, x_{new}, is then obtained and added to the tree.

B. Overview of Fast RRT Algorithm

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

A template set contains several templates, which is off-line generated according to various traffic scenes. At the beginning of the proposed algorithm, the template set is loaded, and the most suitable template is selected according to a short-term goal state (*line 1*). This template is then decomposed and saved as the tree-like structure (*line 2*). If some nodes and edges of a trajectory in the template are not collision-free with the obstacles or the boundaries of the road, these portions will be discarded, and the remainders in the template will be retained (*line 3*) and added to the root (*line 4*). Thus the tree probably has already possessed a part of branches and leaves before randomly searching. The aggressive extension strategy is first used after the template is

Algorithm 1 Fast RRTs algorithm for Autonomous Vehicles

RRTmain()

1: $template_i \leftarrow$ LOADTEMPLET(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) < λ_{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_{rand});
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_{rand});
2: $\mathcal{P}_{near2goal} \leftarrow$ GETNERATEPATH(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**

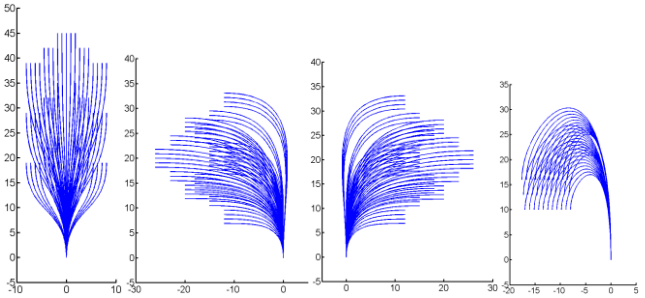
loaded and trimmed, by calling the function RUSHTOWARDSGOAL(), in which a trajectory generator is used for planning from the current state to the goal state (*line5-7*). This aggressive strategy is to directly plan a trajectory towards the goal when without obstacles and keep away from the meaningless search. If it is successful, the algorithm will end and return the tree. If it is unsuccessful, the iteration steps will start. In each iteration step (*line 8-22*), there are two alternative extension strategies, where λ_{ss} denotes the probability of selecting the aggressive extension strategy, and this strategy will be used, if a random number (0 to 1) is less than λ_{ss} (*line 9*). In this case (*line 10-12*), we still call the function RUSHTOWARDSGOAL(), and the result is that a trajectory reaching the goal will be found (if success) or a partial collision-free trajectory is added to the tree and the algorithm continues (if failure). If the random number is not

less than λ_{ss}, we select an ordinary extension strategy similar to the basic RRT algorithm (*line 10-12*). During this process, there are some small modifications that refer to [15], such as the sampling distribution and Dubins distance metric. The stop condition of the proposed algorithm is that the goal state is reached using the aggressive extension strategy, or the maximum number of iterations *K* is exceeded.

C. Generation and Trim of Trajectory Templates

In this sub-section, more details about the generation and using of the off-line templates are described.

The autonomous road vehicle traveling in urban environments has the characteristic of simple and monotonous maneuvers compared with mobile robots in the maze. Therefore, the maneuvers can be categorized in terms of the traffic scenes, by which templates can be generated off-line. Currently, the maneuvers are approximately divided into four types including go straight, turn right, turn left, and U-turn. In each category, considering the driving experience and prior knowledge, various terminal states are drawn. These terminal states with different distances cover five lanes taking current lane as center, which can applies to most of our applications. For each given terminal state, a trajectory generator can be used to generate the trajectories in the template [3]. In addition, a path generator considering the vehicle constraints can also be employed to rapidly generate paths [18], as these paths will be transferred to the trajectories and control input in the model-prediction stage.



(a) Go straight (b) Turn left (c) Turn right (d) U-turn

Fig. 3. Various off-line templates based on different maneuvers

Fig 3 presents four off-line trajectory templates based on different maneuvers which compose a template set. Generally, the higher model, the driving behavior planning model, will offer a short-term planning goal according to the local environmental information perceived by sensors on the vehicle. Based on the desired position and heading of this goal state, a proper trajectory template will be easily picked out from the template set loaded.

The selected template first needs to be decomposed and saved as the tree-like structure. Then the nodes and edges in the 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 while the collision-free portion is reserved. This process is shown in Fig 4.

As the application of off-line template, the tree quite probably has already possessed a part of branches and leaves rather than only a root before randomly searching, because it is unlikely to be trimmed for all branches and leaves. In other words, the vehicle on road is hardly blocked by obstacles at a close distance in most cases.

D. Aggressive Extension Strategy

In this sub-section, the aggressive extension strategy and its difference from the goal bias [9] will be introduced specifically.

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 current state to the goal state using the same trajectory generator as in subsection III-C (*line 2*). If it's successful, the trajectory reaching the goal will be found (*line 4-6*), but if not, the collision-free part of generated trajectory will be reserved in the search tree (*line 8-10*). Even so, the search tree will increase a relatively large branch.

This extension strategy is quite different from the goal bias. Goal bias strategy is to take goal state as sampling point and extend towards the goal by forward simulation in constant step length or time interval. In other words, it can only grow one step towards the goal. 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 strategy is applied in two different places of the proposed algorithm (*line 5* and *line 10*). First, it is used before the search, which has the advantage of direct trajectory planning under the collision-free environment. Second, it is used with certain probability in the iterative search, which can not only promote rapid grow of the search tree, but also make the tree rush directly towards the goal when passing the obstacles in the later period of search. Using this strategy can avoid some meaningless search. Moreover, as a stop condition of the proposed algorithm, this strategy makes the planed trajectory accurately reach the desired terminal state, not just in a region.

E. Model-Prediction stage

When the search tree reaches the goal, the task of algorithm 1 completes and a tree $\mathcal{T}$ is returned. Here, a type of forward simulation approach based on the vehicle's model, named closed-loop prediction is adopted [16], which can further smooth the portion of off-line template and the portion of tree generated on-line, while a trajectory and control sequence for the vehicle would be obtained.

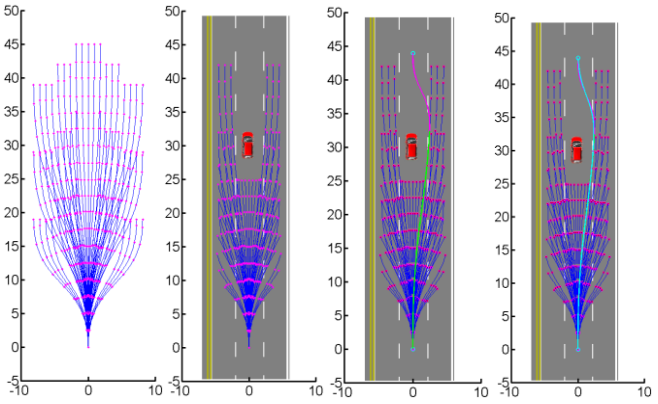
Algorithm 2 Closed-loop prediction trajectory generation

1: $\mathcal{P}_{root2goal} \leftarrow$ EXACTBRANCH($\mathcal{T}, x_{goal}$);
2: $\mathcal{X}_0 \leftarrow \mathcal{T}_{root}$;
3: **while** x_{goal} not reached **do**
4: $\mathbf{u}_i \leftarrow$ PUREPURSUITPI($\mathcal{X}_i$, $\mathcal{P}_{root2goal}$, $M_{vehicle}$);
5: $\mathcal{X}_{i+1} \leftarrow$ FORWARDSIMULATION($\mathcal{X}_i$, u_i, $M_{vehicle}$, Δt);
6: $i \leftarrow i + 1$;
7: **end while**
8: **return** $\mathcal{X}$, $\mathbf{u}$;

Algorithm 2 shows the overall flow of the closed-loop prediction algorithm. By the backwards way, the Function EXACTBRANCH() can extract the trajectory/path $\mathcal{P}_{root2goal}$ in the tree starting from the goal 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 pure-pursuit steering controller, a Proportional-Integral speed controller, and the vehicle's model (*line 4*). 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 running forward simulation for the time interval Δt (*line 5*). Repeat steps in line 4 and line 5 until the resulting trajectory reaches the goal state. Finally, an easy-tracking trajectory $\mathcal{X}$ and control sequence $\mathbf{u}$ for the vehicle will be obtained.

IV. RESULTS

In this section, we experimented adopting the go straight template to verify the performance of the proposed approach. As to this template, three traffic scenes with various complexities are selected including overtaking under a simple traffic environment, lane keeping, and passing under a cluttered traffic environment. In the first case, the implementation process of our approach is mainly presented. The second and the third cases show the result comparison of the proposed approach with RRT and CL-RRT algorithms, respectively. Finally, the statistical experimental results of the proposed algorithm and related algorithms under these three cases are given.

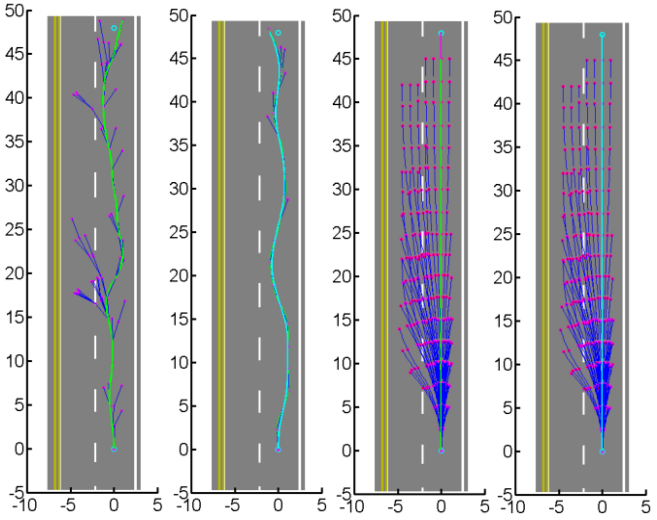


(a) Load the template (b) Trim the template (c) 1st phase result of our method (d) 2nd phase result of our method

Fig. 4. The process and results of our method in overtaking maneuver

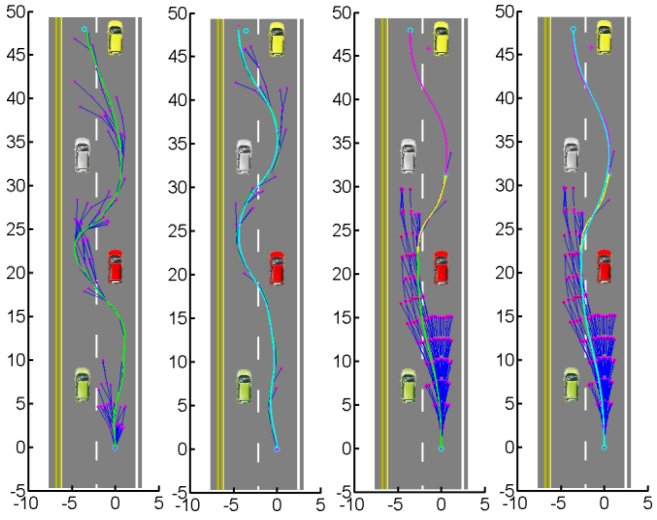
The task of the first case is to plan a trajectory overtaking the vehicle straight-ahead on a three-lanes road. First, the off-line template is loaded, and then decomposed as the tree-like structure, as shown in Fig. 4 (a). In terms of the constraints of road boundaries and the obstacles, the template is trimmed as shown in Fig. 4 (b). The first phase result of the proposed algorithm is given in Fig. 4 (c) using off-line templates and an aggressive extension strategy. Green line indicates the proper path selected from the template. Magenta line indicates the path generated by the aggressive extension strategy. It shows that, from the beginning, the proposed algorithm directly generates the path towards the goal state by the aggressive extension strategy without using the search tree. Fig. 4 (d) shows the second phase result of our method using the model-prediction approach. Cyan line indicates the resulting trajectory. Fig. 4 shows the entire implementation process of the proposed approach.

The second case is the lane keeping on current lane without obstacles, which is the most common driving behavior encountered on road. Fig. 5 (a) presents the result of the RRT algorithm which obviously shows great meander and terminal state error. Fig. 5 (b) shows the result of CL-RRT. Although the meander is lessened, there still exists the terminal state error. Fig. 5 (c) and Fig. 5 (d) respectively show the results of the proposed algorithm in two phases, where the generated paths are marked in the same colors as in Fig. 4 (c) and Fig. 4 (d). It shows that the proposed approach can plan a trajectory which approximates a straight line and reach the terminal state accurately.



(a) the result of RRT (b) the result of CL-RRT (c) 1st phase result of our method (d) 2nd phase result of our method

Fig. 5. Comparison among different methods in lane keeping maneuver



(a) the result of RRT (b) the result of CL-RRT (c) 1st phase result of our method (d) 2nd phase result of our method

Fig. 6. Comparison among different methods in passing maneuver under the cluttered traffic environment

The third case is passing on two lanes road with multi-vehicels. Fig. 6 (a) and Fig. 6 (b) are the results of RRT and CL-RRT, respectively. Their weaknesses are the same as the second case although the CL-RRT is relatively smooth and keeps away from some meaningless extension. Fig. 6 (c) is the first phase result of the proposed algorithm. It shows that, in this complex case, the algorithm cannot directly plan the path only by off-line 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. 6 (d) is the second phase result of the proposed algorithm and 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 I. PERFORMANCE COMPARISON AMONG DIFFERENT APPROACHES IN THREE CASES

	Case 1		Case 2		Case 3	
	Avg. of samples	*Avg. of nodes*	*Avg. of samples*	*Avg. of nodes*	*Avg. of samples*	*Avg. of nodes*
RRT	324.1	172.1	546.1	293	490.6	180.9
RRT-5%gb	169.7	103.8	158.6	105.4	363	138.6
CL-RRT	47.6	35.1	103.2	40.9	608.8	42.3
Ours	0	2	0	5	194	19.1

RRT-5%gb is the basic RRT algorithm with 5% goal bias.

Table I presents experimental comparison of RRT, RRT with 5% goal bias, CL-RRT, and the proposed approach under three cases. To eliminate randomness, each algorithm was tested for 20 MC trials. The table presents the averages of the number of samples and nodes in the tree, which can reflect the efficiency and relative velocity of the algorithms. Note that the nodes in the template are not included in the results, because they are generated off-line. Compared with other related algorithms, the proposed algorithm is more efficient and needs less samples and meaningless extensions. Especially, in some cases with less or no obstacles, the proposed approach can directly generate trajectory by both improvements and avoid the meanders caused by search as far as possible.

V. CONCLUSION

This paper presents a fast RRT algorithm designed for autonomous road vehicles, which introduces an off-line template set based on the traffic scenes and an aggressive extension strategy of search tree, and combines a prior closed-loop prediction approach. Experiments were conducted for comparing the performance of our approach against other related approaches in various scenes. Experimental results demonstrate that proposed approach can plan a trajectory 1) faster with a lower density of obstacles, 2) more accurate towards goal state, 3) and with less meanders. The off-line template technique perhaps is only suitable for the application of autonomous road vehicles, while the aggressive extension strategy seems to be generalized.

As future work we would like to estimate the quality of nodes for selecting the best extension node, because the end nodes of the trimmed trajectories are not often promising in the template or after rushing towards the goal. A choice is the CVF (Constraint Violation Frequency) method [19], which can penalize the nodes that once leaded to the collision. Furthermore, more trails in more traffic scenes should be conducted.

REFERENCES

[1] L. Figueiredo, I. Jesus, J. T. Machado, J. Ferreira, and J. M. Carvalho, "Towards the development of intelligent transportation systems," in *Proc. 14th IEEE Int. Conf. Intelligent Transportation Systems*, Washington D. C., 2001, vol. 88, pp. 1206–1211.

[2] H. Cheng, *Autonomous Intelligent Vehicles: Theory, Algorithms, and Implementation*. Springer, 2011, pp. 3–5.

[3] T. M. Howard, M. Pivtoraiko, R. A. Knepper and A. Kelly, "Model-Predictive Motion Planning," *IEEE Robotics & Automation Magazine*. 2014, pp. 64–73.

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

[5] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," tech. report, 1998.

[6] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. 2000 IEEE Int. Conf. on Robotics and Automatio*n, San Francisco, 2000, vol. 2, pp. 995–1001.

[7] H. Long, Q. H. Do and S. Mita, "Unified path planner for parking an autonomous vehicle based on RRT," in *Proc. 2011 IEEE Int. Conf. on Robotics and Automatio*n, Shanghai, 2011, pp. 5622–5627.

[8] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *Proc.2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Zurich, 2002, vol. 3, pp. 2383–2388.

[9] C. Urmson and R. G. Simmons, "Approaches for heuristically biasing RRT growth," in *Proc.2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Las Vegas, 2003, pp. 1178–1183.

[10] S. Rodriguez, X. Tang, J. M. Lien and N. M. Amato, "An obstacle-based rapidly-exploring random tree," in *Proc.2006 IEEE Int. Conf. on Robotics and Automatio*n, Orlando, 2006, pp. 895–900.

[11] J. Denny, M. Morales, S. Rodriguez and N. M. Amato, "Adapting RRT growth for heterogeneous environments," in *Proc.2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Tokyo, 2013, pp. 1772–1778.

[12] N. A. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *Proc. 2007 IEEE Int. Conf. on Robotics and Automatio*n, Roma, 2007, pp. 1617–1624.

[13] S. Karaman and E.Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, 2011, vol. 30, no. 7, 2011, pp. 846–894.

[14] 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. on Decision and Control and European Control*, Orlando, 2011, pp. 3276–3282.

[15] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *IEEE Trans. Control Systems Technology*, 2009, vol. 17, no. 5, pp. 1105–1118.

[16] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli and J. P. How, "Motion planning in complex environments using closed-loop prediction," in *Proc. AIAA Guidance, Navigation, and Cont*rol *Conf.*, Honolulu, 2008.

[17] J. Xin, C. Wang, Z. Zhang, and N. Zheng, "China future challenge: Beyond the intelligent vehicle," *IEEE ITS Society Newsletter*, 2014, vol. 16, no. 2 pp. 8-10.

[18] K. Kawabata, L. Ma, J. R. Xue, and N. N. Zheng, "A path generation method for automated vehicles based on Bezier curve," in *Proc.2013 IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, Wollongong, 2013, pp. 991–996.

[19] L. Jaillet, J. Hoffman, J. Van den Berg, P. Abbeel, J. M. Porta, and K. Goldberg, "Eg-rrt: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles," in *Proc.2011 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, San Francisco, 2011, pp. 2646–2352.