A Model based Path Planning Algorithm for Self-driving Cars in Dynamic Environment

Chaocheng Li
Department of Control
Science and Engineering
Tongji University
Shanghai, P. R. China
Email: 13lichaocheng@tongji.edu.cn

Jun Wang
Department of Control
Science and Engineering
Tongji University
Shanghai, P. R. China
Email: junwang@tongji.edu.cn

Xiaonian Wang
and Yihuan Zhang
Department of Control
Science and Engineering
Tongji University
Shanghai, P. R. China

***Abstract*—Self-driving cars require robust and fast path planning algorithms to operate in dynamic environment. In the last years, model based path planning has emerged as an effective solution to real-time path planning taking into account the kinematics of the vehicle. However, this approach only validates in simple environment such as static or single moving obstacle. In this paper, we propose a model based path planning algorithm for dynamic environment. The approach generates candidate trajectories online, then evaluates and selects the most appropriate one according to real-time environment information. The selected trajectory can be directly tracked until emergency occurs and replanning starts. Thus, the planning algorithm is anytime and dynamic, and also effective in complex multi-obstacle environment. Simulation results of different scenarios show the validity of the proposed algorithm.**

Keywords*—Self-driving cars, vehicle model, path planning, dynamic environment.

I. INTRODUCTION

In the past two decades, self-driving cars have been drawing a considerable attention from both academia and industry, with promising applications in military, transportation and industrial production. Self-driving cars are expected to perform various missions to replace humans in different fields. A significant function required for self-driving cars performing missions is to plan reasonable trajectories offline or to generate trajectories dynamically based on real-time environment information.

Many different algorithms such as heuristic searching, artificial intelligence and model based algorithms have been developed to cope with path planning problems of self-driving cars. The heuristic searching algorithms such as potential field approach [1], [2], A* algorithm [3] and D* algorithm [4] guarantee to obtain the optimal path if the path exists, however, the environment is gridized and the vehicle model is not concerned. Artificial intelligence based algorithms including fuzzy logic algorithm [5], [6], genetic algorithm [7], [8] and neural networks [9] have been successfully implemented in path planning, however, the optimal results or algorithm performance are not necessarily guaranteed. Model based algorithms [10], [11], [12], [13], [14] take vehicle kinematic constraints into consideration to guarantee smoothness and feasibility, but it mostly deals with static obstacles or single moving obstacle, which makes it ineffective in dynamic multi-obstacle environment.

In this paper, a model based path planning algorithm for dynamic environment is proposed. The algorithm consists of two parts: online trajectory generation and selection. The trajectory generator produces candidate trajectories based on current vehicle state, terminal vehicle state and vehicle kinematic constraints. Trajectory selector focuses on evaluating candidate trajectories and selecting the most appropriate one based on threat assessment of environment. This algorithm not only guarantees a safe, short and feasible trajectory, but also performs well in dynamic environment.

II. ONLINE TRAJECTORY GENERATION

A. Model based trajectory generation

The trajectory generation method is improved based on [12] which considers a kinematic model of a car-like vehicle. In this paper, the terminal state setup is improved according to current vehicle state and road information. In addition, performance index in [12] took connecting trajectory between start and goal as a straight line, which is unreasonable when road is a curve, so a Hermite interpolation based connecting trajectory generation method is proposed considering initial and goal heading. Thus, performance index is redefined to minimize offset to connecting trajectory, with the best performance index a model based trajectory can be obtained.

1) Vehicle model: To navigate real environment, position and heading constraints are typically required to properly orient a vehicle along the road, so the vehicle kinematic model is given by

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = v\dfrac{\tan\delta}{l} \end{cases} \tag{1}$$

where (x,y) are the coordinates of the center point of the rear axle, θ is the heading angle of the car body with respect to the x axis, and δ is the steering angle of the front wheel with respect to the vehicle's longitudinal line, which can be seen as a control input. l is the distance between the front axle and the rear axle, which is shown in Fig. 1.

2) Initial and terminal states setup: To plan a trajectory, the initial and terminal states should be given first. Suppose that the initial state is $\mathbf{x}_0 = [x_0, y_0, \theta_0, \delta_0, v_0, \dot{v}_0]^T$ at time t_0 and terminal state is $\mathbf{x}_f = [x_f, y_f, \theta_f, \delta_f, v_f, \dot{v}_f]^T$ at time t_f $(t_f > t_0)$. t_f can be estimated with preview distance and current vehicle

978-1-4673-7189-6/15/$31.00©2015 IEEE

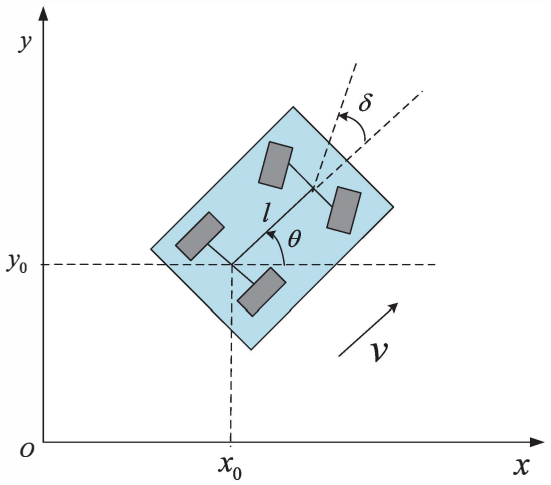


Fig. 1. The vehicle model.

velocity. $(x_f, y_f, \theta_f, \delta_f, \dot{v}_f)$ can be previewed with mission profile and road information. The terminal velocity v_f is computed based on the maximum speed limit v_{max} of the current road segment and the curvature of the road.

To avoid rollover or sideslip, the maximum terminal velocity v_{cur} should be constrained with maximum lateral acceleration denoted by a_c,

$$v_{cur} \leq \sqrt{a_c R} \tag{2}$$

where $R = \frac{1}{c_f}$, c_f is the road curvature of goal.

To achieve safety and efficiency, the terminal velocity v_f is given by

$$v_f = \min(v_{max}, v_{cur}) \tag{3}$$

Given initial and terminal states, the connecting trajectory $(\mathbf{x}_n, \mathbf{y}_n)$ between $\mathbf{x}_0$ and $\mathbf{x}_f$ is constructed with Hermite interpolation, for every point (x, y) of the connecting trajectory

$$y = y_0(1-2\frac{x-x_0}{x_0-x_f})(\frac{x-x_f}{x_0-x_f})^2 + y_f(1-2\frac{x-x_f}{x_f-x_0})(\frac{x-x_0}{x_f-x_0})^2$$
$$+\tan\theta_0(x-x_0)(\frac{x-x_f}{x_0-x_f})^2 + \tan\theta_f(x-x_f)(\frac{x-x_0}{x_f-x_0})^2 \tag{4}$$

3) Path planner: Given the initial state $\mathbf{x}_0$ and terminal state $\mathbf{x}_f$, the desired trajectory is presented by $(\mathbf{x}_d, \mathbf{y}_d)$, according to the vehicle model in Equation 1, the constraints of the initial and terminal states are

$$\begin{cases} \mathbf{x}_d(t_0) = x_0 \\ \dot{\mathbf{x}}_d(t_0) = v_0 \cdot \cos\theta_0 \\ \ddot{\mathbf{x}}_d(t_0) = \dot{v}_0 \cdot \cos\theta_0 - \dfrac{v_0^2 \cdot \tan\delta_0 \cdot \sin\theta_0}{l} \end{cases} \tag{5}$$

$$\begin{cases} \mathbf{x}_d(t_f) = x_f \\ \dot{\mathbf{x}}_d(t_f) = v_f \cdot \cos\theta_f \\ \ddot{\mathbf{x}}_d(t_f) = \dot{v}_f \cdot \cos\theta_f - \dfrac{v_f^2 \cdot \tan\delta_f \cdot \sin\theta_f}{l} \end{cases} \tag{6}$$

$$\begin{cases} \mathbf{y}_d(t_0) = y_0 \\ \dot{\mathbf{y}}_d(t_0) = v_0 \cdot \sin\theta_0 \\ \ddot{\mathbf{y}}_d(t_0) = \dot{v}_0 \cdot \sin\theta_0 - \dfrac{v_0^2 \cdot \tan\delta_0 \cdot \cos\theta_0}{l} \end{cases} \tag{7}$$

$$\begin{cases} \mathbf{y}_d(t_f) = y_f \\ \dot{\mathbf{y}}_d(t_f) = v_f \cdot \sin\theta_f \\ \ddot{\mathbf{y}}_d(t_f) = \dot{v}_f \cdot \sin\theta_f - \dfrac{v_f^2 \cdot \tan\delta_f \cdot \cos\theta_f}{l} \end{cases} \tag{8}$$

The above equations show that both $\mathbf{x}_d$ and $\mathbf{y}_d$ have six constraints, so at least six coefficients are needed to satisfy these constraints. In order to obtain the flexibility of choosing a collision-free trajectory, one more parameter is added to increase degree of freedom to trajectory generation, so the desired trajectory is formed as a sixth-order polynomials with seven coefficients:

$$\begin{cases} \mathbf{x}_d(t) = \sum_{k=0}^{6} a_k t^k \\ \mathbf{y}_d(t) = \sum_{k=0}^{6} b_k t^k \end{cases} \tag{9}$$

Consider constraints of initial and terminal states, the following equations must be satisfied for $\mathbf{x}_d$ and $\mathbf{y}_d$:

$$\begin{cases} \mathbf{x}_d(t_0) = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 t_0^4 + a_5 t_0^5 + a_6 t_0^6 \\ \dot{\mathbf{x}}_d(t_0) = a_1 + 2a_2 t_0 + 3a_3 t_0^2 + 4a_4 t_0^3 + 5a_5 t_0^4 + 6a_6 t_0^5 \\ \ddot{\mathbf{x}}_d(t_0) = 2a_2 + 6a_3 t_0 + 12a_4 t_0^2 + 20a_5 t_0^3 + 30a_6 t_0^4 \\ \mathbf{y}_d(t_0) = b_0 + b_1 t_0 + b_2 t_0^2 + b_3 t_0^3 + b_4 t_0^4 + b_5 t_0^5 + b_6 t_0^6 \\ \dot{\mathbf{y}}_d(t_0) = b_1 + 2b_2 t_0 + 3b_3 t_0^2 + 4b_4 t_0^3 + 5b_5 t_0^4 + 6b_6 t_0^5 \\ \ddot{\mathbf{y}}_d(t_0) = 2b_2 + 6b_3 t_0 + 12b_4 t_0^2 + 20b_5 t_0^3 + 30b_6 t_0^4 \end{cases} \tag{10}$$

The equations can also be obtained for t_f similarly.

The equations for t_0 and t_f can be formed as a matrix:

$$\begin{cases} \mathbf{g}_1 = \mathbf{P}\mathbf{a} + \mathbf{n}a_6 \\ \mathbf{g}_2 = \mathbf{P}\mathbf{b} + \mathbf{n}b_6 \end{cases} \tag{11}$$

where

$$\mathbf{P} = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \tag{12}$$

and

$$\begin{cases} \mathbf{a} = [a_0, a_1, a_2, a_3, a_4, a_5, a_6]^T \\ \mathbf{b} = [b_0, b_1, b_2, b_3, b_4, b_5, b_6]^T \\ \mathbf{g}_1 = [\mathbf{x}_d(t_0), \dot{\mathbf{x}}_d(t_0), \ddot{\mathbf{x}}_d(t_0), \mathbf{x}_d(t_f), \dot{\mathbf{x}}_d(t_f), \ddot{\mathbf{x}}_d(t_f)]^T \\ \mathbf{g}_2 = [\mathbf{y}_d(t_0), \dot{\mathbf{y}}_d(t_0), \ddot{\mathbf{y}}_d(t_0), \mathbf{y}_d(t_f), \dot{\mathbf{y}}_d(t_f), \ddot{\mathbf{y}}_d(t_f)]^T \\ \mathbf{n} = [t_0^6, 6t_0^5, 30t_0^4, t_f^6, 6t_f^5, 30t_f^4]^T \end{cases} \tag{13}$$

According to (9) and (11), the polynomials can be expressed as:

$$\begin{aligned}\mathbf{x}_d(t) &= f(t)\mathbf{P}^{-1}(\mathbf{g}_1 - \mathbf{n}a_6) + a_6 t^6 \\ \mathbf{y}_d(t) &= f(t)\mathbf{P}^{-1}(\mathbf{g}_2 - \mathbf{n}b_6) + b_6 t^6\end{aligned} \tag{14}$$

where $f(t) = [1, t, t^2, t^3, t^4, t^5]$.

In (9), a_6 and b_6 are added parameters which are determined by minimizing the trajectory length. The performance index is defined as

$$J(\mathbf{x}_d, \mathbf{y}_d) = \int_{t_0}^{t_f} \left[(\mathbf{x}_d(t) - \mathbf{x}_n(t))^2 + (\mathbf{y}_d(t) - \mathbf{y}_n(t))^2\right] dt \tag{15}$$

where $(\mathbf{x}_n, \mathbf{y}_n)$ is the connecting trajectory between $\mathbf{x}_0$ and $\mathbf{x}_f$. The performance index integrates the difference between connecting trajectory and planned trajectory, the a_6 and b_6 that minimizes the perform index are chosen. For simplicity, the solution of performance index can be obtained by building linear relationship of a_6 and b_6, then the performance index becomes quadratic, which means that there must be a_6 and b_6 for minimum J, thus the trajectory $(\mathbf{x}_d, \mathbf{y}_d)$ can always be obtained. According to (1), the velocity of trajectory can be resolved subsequently with $\mathbf{v}(t) = \sqrt{(\dot{\mathbf{x}}_d(t))^2 + (\dot{\mathbf{y}}_d(t))^2}$ if the trajectory $(\mathbf{x}_d, \mathbf{y}_d)$ is obtained.

B. Candidate trajectories generation

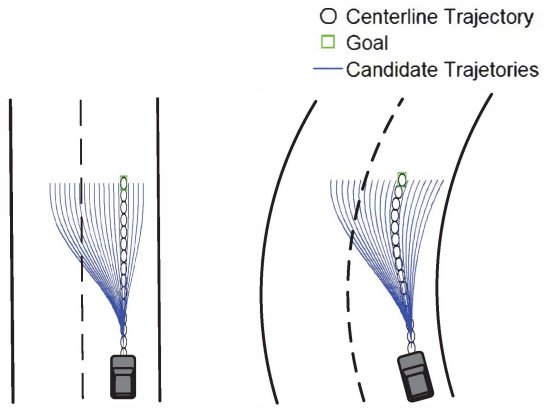
During on-road navigation, the goal is set with respect to mission profile and road information, the trajectory generator attempts to generate a trajectory that moves the self-driving car towards this goal. To accomplish this, the trajectory generator first constructs a curve connecting the initial point and goal point, the curve denotes the centerline trajectory the self-driving car should follow when no obstacles exist. To robustly track the centerline trajectory and avoid static or moving obstacles, the trajectory generator constructs a set of candidate trajectories with respect to corresponding candidate goals. All candidate goals are set with same longitudinal distance y_f from the self-driving car and same heading θ_f, while the lateral distance of candidate goals are set with a certain lateral offset from the centerline trajectory to offer more options. The trajectory generator produces feasible trajectories to these candidate goals dynamically. Assume the goal of centerline trajectory is (x_f, y_f, θ_f), the lateral offset between adjacent candidate goals is l_{off}, the left and right lateral boundary at y_f are x_{left} and x_{right} respectively. The lateral distance of the left-most goal and the right-most goal are

$$\begin{cases} x_{lmost} = x_f - \left\lfloor \dfrac{x_f - x_{left}}{l_{off}} \right\rfloor l_{off} \\ x_{rmost} = x_f + \left\lfloor \dfrac{x_{right} - x_f}{l_{off}} \right\rfloor l_{off} \end{cases} \tag{16}$$

Thus the lateral distance of candidate goals are obtained

$$\mathbf{x}_{goal} = [x_{lmost}, x_{lmost} + l_{off}, \ldots, x_f, x_f + l_{off}, \ldots, x_{rmost}]^T \tag{17}$$

The candidate goals and trajectories in straight road and curve scenarios are shown in Fig. 2.



(a) Straight road scenario (b) Curve scenario

Fig. 2. Candidate goals and trajectories.

III. Online Trajectory Selection

The trajectory generator constructs a set of candidate trajectories with constraints of current vehicle state, terminal vehicle state and vehicle kinematics. The trajectory generator offers more options in case of obstacles in real environment. When on-road navigation, candidate trajectories are evaluated against their proximity to static and moving obstacles in the environment, distance from the centerline trajectory and other metrics as well, thus the one with the the best performance is selected and directly tracked by the self-driving car.

A. Threat probability area and vehicle state prediction

The "threat probability area" is developed to evaluate threat of obstacle to candidate trajectories generated above. Assume that the obstacle velocity stays constant in a short predictive period T, thus we can predict the obstacle's position in several steps, combined with obstacle's size, the "threat probability area" shown in Fig. 3 is obtained, the length of rectangle depends on obstacle current velocity, the larger the velocity, the longer the rectangle. The width of rectangle denotes the size of the obstacle.

Suppose the current state of obstacle i is $(\mathbf{x}_{o,i}(t_0), \mathbf{y}_{o,i}(t_0), \mathbf{v}_{o,i}(t_0), \theta_{o,i}(t_0))$, where $(\mathbf{x}_{o,i}(t_0), \mathbf{y}_{o,i}(t_0))$ is the location in cartesian coordinates, and $\mathbf{v}_{o,i}(t_0)$ and $\theta_{o,i}(t_0)$ are velocity and heading respectively. The predictive period T is decomposed as a time sequence $[0, \Delta t, 2\Delta t, \ldots, T]^T$, thus we can predict the position of obstacle i at time t, that is

$$\begin{cases} \mathbf{x}_{o,i}(t) = \mathbf{x}_{o,i}(t_0) + \mathbf{v}_{o,i}(t_0) \cdot \cos(\theta_{o,i}(t_0)) \cdot t \\ \mathbf{y}_{o,i}(t) = \mathbf{y}_{o,i}(t_0) + \mathbf{v}_{o,i}(t_0) \cdot \sin(\theta_{o,i}(t_0)) \cdot t \\ \theta_{o,i}(t) = \theta_{o,i}(t_0) \end{cases} \tag{18}$$

So the predictive trajectory of obstacle i is obtained as $(\mathbf{x}_{o,i}, \mathbf{y}_{o,i}, \mathbf{v}_{o,i}, \theta_{o,i})$, then the "threat probability area" represented by $(\hat{\mathbf{x}}_{o,i}, \hat{\mathbf{y}}_{o,i}, \hat{\mathbf{v}}_{o,i}, \hat{\theta}_{o,i})$ is formed when the predictive trajectory is expanded with respect to the obstacle's width.

On the other hand, the predictive trajectory of self-driving car can be every one of candidate trajectories, and all candidate trajectories are of the same length which are defined as $[\mathbf{X}_{m\times n}, \mathbf{Y}_{m\times n}, \mathbf{V}_{m\times n}, \Theta_{m\times n}]$, where m, n are number of candidate trajectories and length of candidate trajectory respectively.

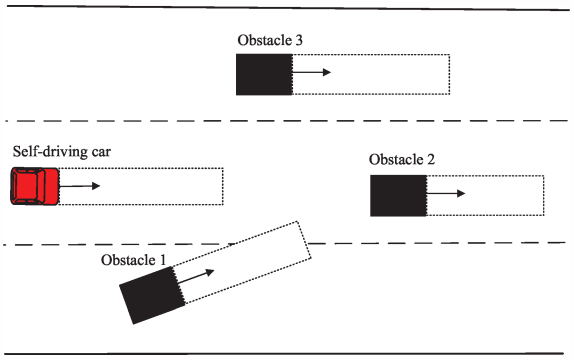


Fig. 3. Threat probability area.

So the predictive trajectory of self-driving car can be represented by $(\mathbf{X}_i, \mathbf{Y}_i, \mathbf{V}_i, \Theta_i)$ where $i = 1, 2, ..., m$.

B. Trajectory evaluation

The candidate trajectories are evaluated against the threat of obstacle "threat probability area" and the distance from the centerline trajectory, the one with the minimum total cost is selected as the desired trajectory.

1) Cost of "threat probability area": The cost of obstacles is evaluated according to "threat probability area" and predictive trajectory of self-driving car. The "threat probability area" and predictive trajectory of self-driving car are time related, because the collision only occurs when obstacle and self-driving car are in the same position at the same time. So the minimum distance of "threat probability area" and predictive trajectory of self-driving car at the same time can be applied to weigh the cost of obstacles.

So the threat of obstacle j to candidate trajectory i is evaluated as

$$J_{o,i,j} = \min_{0 \le t \le T} \left(\sqrt{[\mathbf{X}_i(t) - \hat{\mathbf{x}}_{o,j}(t)]^2 + [\mathbf{Y}_i(t) - \hat{\mathbf{y}}_{o,j}(t)]^2} \right) \quad (19)$$

here we define

$$C(J_{o,i,j}) = \frac{1}{\max[(J_{o,i,j} - R_d), \varepsilon]} \quad (20)$$

and the threat of all obstacles to candidate trajectory i is obtained

$$J_{o,i} = \sum_{j=1}^{p} C(J_{o,i,j}) \quad (21)$$

where R_d is the dangerous distance, p is the number of obstacles and ε is set for nonsingularity.

2) Cost of deviation from centerline trajectory: To reach the goal safely and quickly, the self-driving car should drive near the centerline trajectory while avoiding obstacles. It means that the larger the deviation from centerline trajectory, the larger the cost of the candidate trajectory.

So the deviation cost of centerline trajectory candidate trajectory i is defined as

$$J_{dev,i} = \sum_{j=1}^{n} \sqrt{[\mathbf{X}_i(j) - \mathbf{x}_c(j)]^2 + [\mathbf{Y}_i(j) - \mathbf{y}_c(j)]^2} \quad (22)$$

where $(\mathbf{x}_c, \mathbf{y}_c)$ is the centerline trajectory and $(\mathbf{X}_i(j), \mathbf{Y}_i(j))$ represents the positions of candidate trajectory i at time j.

The total cost of each candidate trajectory i is obtained

$$J_i = w_1 \cdot J_{o,i} + w_2 \cdot J_{dev,i} \quad (23)$$

where w_1, w_2 are weighting parameters, and the one with minimum total cost is selected as the desired trajectory.

IV. Trajectory Replanning

Given that the desired trajectory is selected, it is assumed that the trajectory can be precisely tracked. While tracking the selected trajectory, the self-driving car may encounter environmental changes due to limited sensing ranges and the appearance or loss of moving obstacles. In order to model the environment, the following assumptions are made without loss of generality.

1. The self-driving car is modeled as a rectangle centered at $(\mathbf{x}_0(t), \mathbf{y}_0(t))$ with length l_0 and width w_0, also radius r_0 of rectangle's circumcircle is applied to represent the size of self-driving car for simplicity. The velocity of self-driving car is $\mathbf{v}(t) = (\mathbf{v}_x(t), \mathbf{v}_y(t))$, where subscripts x and y denote the components on x and y axes respectively.

2. The obstacles are also represented by rectangles, centered at $(\mathbf{x}_{o,i}(t), \mathbf{y}_{o,i}(t))$, with length l_i, width w_i and radius r_i of circumcircle, similarly, $\mathbf{v}_{o,i}(t) = (\mathbf{v}_{o,i,x}(t), \mathbf{v}_{o,i,y}(t))$.

3. The dangerous region is defined as a circle centered at $(\mathbf{x}_0(t), \mathbf{y}_0(t))$ with radius R_d, the alert region is a concentric circle with radius R_a similarly [15], R_d, R_a are set according to vehicle current velocity.

4. "An obstacle is coming close" means the lateral relative velocity of obstacle and the self-driving car $\Delta v_x(t) = (\mathbf{v}_{o,i,x}(t) - \mathbf{v}_x(t))$ and lateral relative distance $\Delta s(t) = (\mathbf{x}_{o,i}(t) - \mathbf{x}_0(t))$ hold the condition $\Delta v_x(t) \cdot \Delta s(t) < 0$.

The model of dynamic environment is shown in Fig. 4.

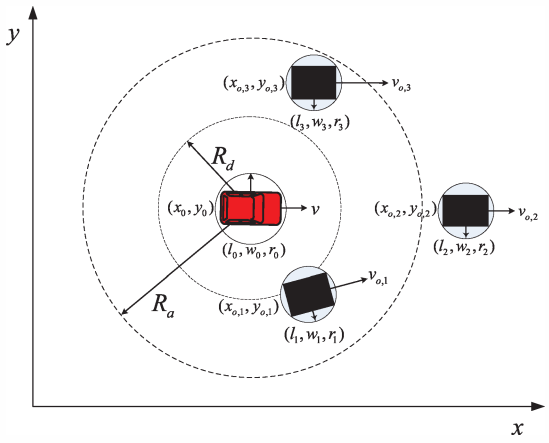


Fig. 4. Model of dynamic environment

To track the trajectory, the self-driving car just needs to follow the preview point of the selected trajectory, but when environment changes, self-driving car may need to replan according to self-driving car's current state and environment information.

The self-driving car starts replanning if one of the following conditions holds:

1. One obstacle is crossing dangerous region.
2. One obstacle is crossing alert region and coming close.
3. The selected trajectory has been completely tracked.
4. The self-driving car passes one obstacle.

V. Simulations and Results

The performance of the proposed model based path planning algorithm is demonstrated in straight road and curve scenarios. In both scenarios, two moving obstacles with velocity less than the self-driving car's velocity are simulated. The self-driving car starts from initial state $\mathbf{x}_0$ at time t_0, and reaches goal state $\mathbf{x}_f$ at time t_f while avoiding static or moving obstacles on the road. When the desired trajectory is generated and selected, the self-driving car directly tracks it until replanning conditions are held, the replanning starts based on the update of initial and terminal states, and the same process repeats until the self-driving car reaches the destination. The planning period is 0.1 second in account of high dynamic environment. The preview distance is set according to self-driving car's velocity which is 50 m in the simulation. Two simulation scenarios are set as follows.

A. Straight road scenario

For the straight road scenario, two obstacles are initialized at (-2,40), (0,25), both with constant velocity $v = 5$ m/s and the same heading $\theta = \frac{\pi}{2}$, the initial state of self-driving car is $\mathbf{x}_0 = [0,0,\frac{\pi}{2},0,10,0]^T$ and the goal state is $\mathbf{x}_f = [0,100,\frac{\pi}{2},0,10,0]^T$. The simulation results are illustrated in Fig. 5 to Fig. 7. Assume that the obstacles are within sensor range, a set of candidate trajectories shown in Fig. 5 are generated, the self-driving car selects the green one because it has the minimum cost with respect to the obstacle "threat probability area" and the goal, the algorithm runs every 0.1 second to update states of self-driving car and obstacles and check whether the replanning needs to start or not. During the trajectory tracking, the self-driving car is always outside dangerous region, so the first selected trajectory is completely tracked and replanning starts at the end of trajectory shown in Fig. 6. The same process repeats until the self-driving car arrives at the destination safely in Fig. 7.

B. Curve scenario

For the curve scenario, two obstacles are initially located at (-2,35), (0,25) with different velocity of 8 m/s and 4 m/s respectively, both obstacles runs parallel with road boundary. The initial state of self-driving car is $\mathbf{x}_0 = [0,0,\frac{\pi}{2},0,10,0]^T$ and the goal state is $\mathbf{x}_f = [-1,100,\frac{4\pi}{9},0,8,0]^T$.

The simulation results are illustrated in Fig. 8 to Fig. 10. Similarly, a set of candidate trajectories shown in Fig. 8 are generated, the self-driving car selects the green one although the obstacle is currently on this trajectory, the self-driving car follows the green one until it passes the right obstacle shown in Fig. 9, the self-driving car updates terminal state and starts replanning to quickly reach the destination. The self-driving car arrives at the destination safely in Fig. 10.

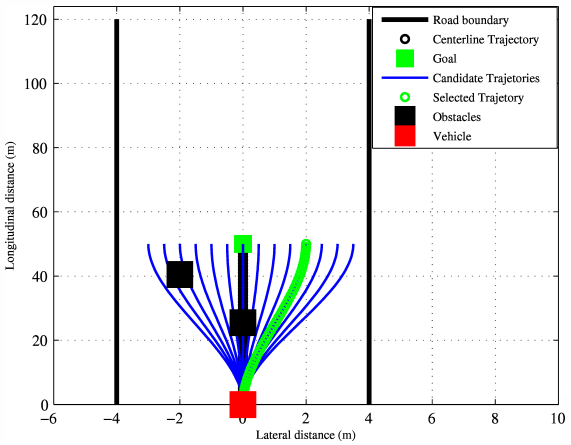


Fig. 5. Initialization of obstacles and self-driving car in straight road scenario.

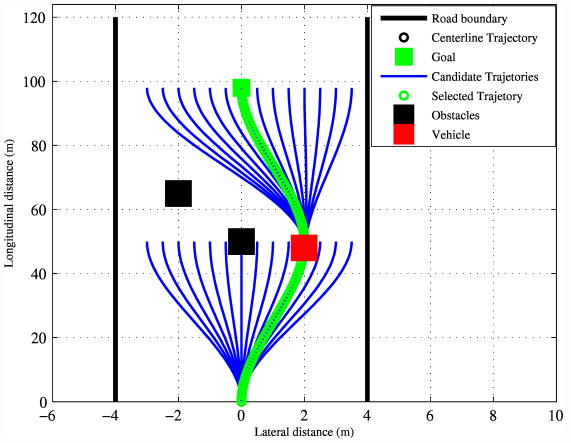


Fig. 6. Replanning when the selected trajectory is completely tracked.

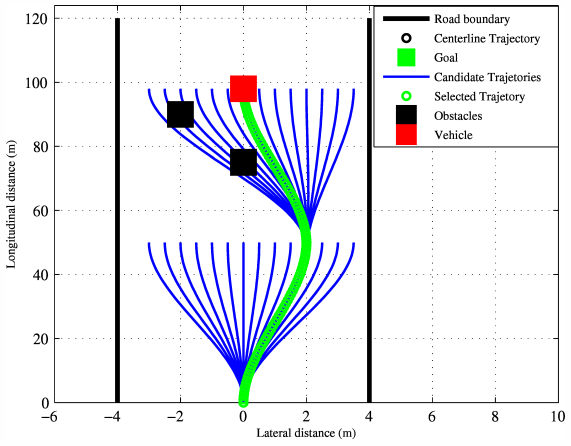


Fig. 7. Self-driving car successfully reaches the destination.

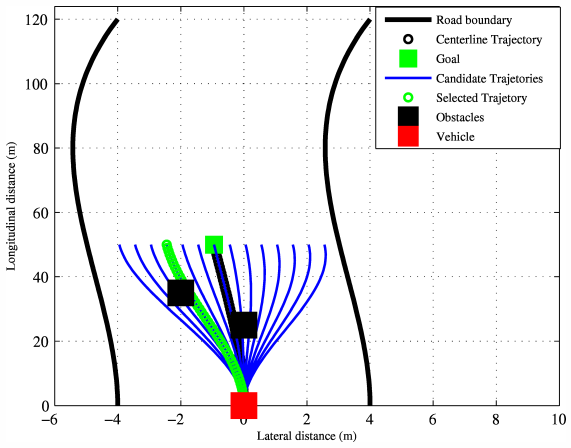


Fig. 8. Initialization of obstacles and vehicle in curve scenario.

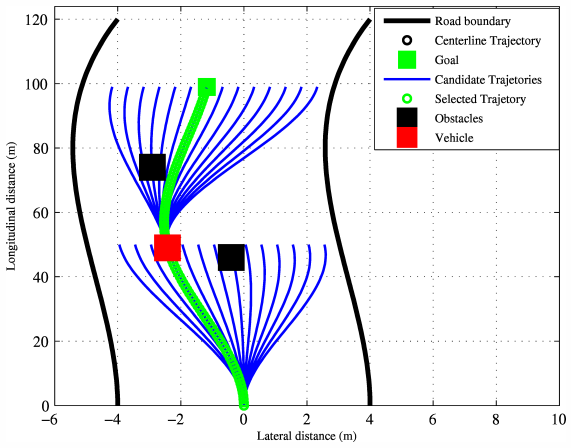


Fig. 9. Replanning when self-driving car passes one obstacle.

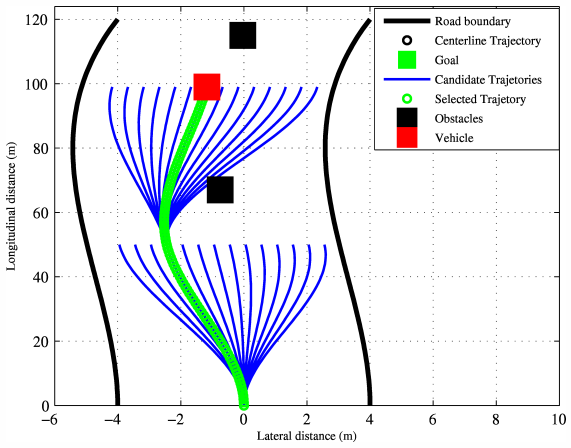


Fig. 10. Self-driving car successfully reaches the destination.

In both scenarios, it is seen that the self-driving car can successfully generate candidate trajectories and select the most appropriate one to follow, during the tracking, the changes of environment are detected real-time based on which the self-driving car arbitrates whether to track the current trajectory or replan. The same process repeats until the self-driving car reaches the destination. Besides, the algorithm is time-efficient to generate a solution at each planning period, which is significant in high dynamic environment.

VI. Conclusions

In this paper, a model based path planning algorithm for dynamic environment is proposed. The proposed algorithm takes vehicle kinematics into account to guarantee smooth and feasible trajectories. Online trajectory generation and evaluation methods are combined to obtain an optimal trajectory considering environment information. During the navigation on road, if any collision is detected, the self-driving car starts replanning to make sure the collision is avoided. The simulation results in both straight road and curve scenarios have shown the feasibility and high computational efficiency of the algorithm. Therefore, the algorithm is both optimal and practical, and improves the performance of path planning in dynamic environment.

References

[1] Khatib, O., "Real-time obstacle avoidance for manipulators and mobile robots", IJRR, vol. 5(1), pp. 90-98, 1986.

[2] Y. Hwang, and N. Ahuja, "A potential field approach to path planning", IEEE Trans. Robotics. Automation, vol. 8, pp. 23-32, 1992.

[3] A. Stentz, "Optimal and efficient path planning for partially-known environments", ICRA, pp. 3310-3317, 1994.

[4] Likhachev, M., et al., "Anytime Dynamic A*: An Anytime, Replanning Algorithm", ICAPS, pp. 262-271, 2005.

[5] A. Saffiotti, "The uses of fuzzy logic in autonomous robot navigation", Soft Computing, vol. 1, pp. 180-197, 1997.

[6] Antonelli, G., S. Chiaverini and G. Fusco, "A fuzzy-logic-based approach for mobile robot path tracking", IEEE Trans. Fuzzy. Systems, vol. 15(2), pp. 211-221, 2007.

[7] K. Sedighi, R. L. Wainwright, and H. Tai, "Autonomous local path planning for a mobile robot using a genetic algorithm", Evolutionary Computation, pp. 1338-1345, 2004.

[8] A. Ismail, A. Sheta and M. Al-Weshah, "A mobile robot path planning using genetic algorithm in static environment", Journal of Computer Science, vol. 4, pp. 341-350, 2008.

[9] M. Phinni, A. Sudheer, and K. Jemshid, "Obstacle Avoidance of a wheeled mobile robot: A Genetic-neurofuzzy approach", Journal of Field Robotics, vol. 25, pp. 939-960, 2008.

[10] R. Pepy, A. Lambert and H. Mounier, "Path planning using a dynamic vehicle model", Information and Communication Technologies. vol. 1, pp. 781-786, 2006.

[11] Cunjia Liu and Wen Hua Chen, "Optimisation based control framework for autonomous vehicles: algorithm and experiment", ICMA, pp. 1030-1035, 2010.

[12] T. Shim, G. Adireddy and H. Yuan, "Autonomous vehicle collision avoidance system using path planning and model-predictive-control-based active front steering and wheel torque control", Journal of automobile engineering, pp. 275-283, 2012.

[13] Pivtoraiko, M. and Kelly, A, "Efficient contstrained path planning via search in state lattices", International Symposium on Artificial Intelligence, Robotics, and Automation in Space, vol. 26, pp. 308-333, 2009.

[14] Qu, Zhihua., et al, "A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles", IEEE Trans. Rob, vol. 4, pp. 978-993, 2004.

[15] Urmson, C., et al,"Autonomous driving in urban environments: Boss and the Urban Challeng", Journal of Field Robotics, vol. 25, pp. 425C466, 2008.