Parking Like a Human: A Direct Trajectory Planning Solution

Wei Liu, Zhiheng Li, *Member, IEEE*, Li Li, *Senior Member, IEEE*, and Fei-Yue Wang, *Fellow, IEEE*

***Abstract*— Parking control problems remain to be fully solved for autonomous vehicles. Existing approaches usually first design a reference parking trajectory that does not exactly match vehicle dynamic constraints and then apply certain online negative feedback control to make the vehicle roughly track this reference trajectory. In this paper, we propose a novel trajectory planning method that directly links the actual parking trajectories and the steering actions to find the best parking trajectory. Tests show that this new approach has high reliability and less computation cost. Moreover, we also discuss how to counter with trajectory planning errors that are caused by model uncertainty in this paper. We show that an appropriate combination of feedforward trajectory planning and online feedback control can solve such problems.**

***Index Terms*— Autonomous vehicles, parking, trajectory planning.**

I. Introduction

AUTONOMOUS vehicles are designed to complete most driving tasks instead of human drivers [1]–[3]. Such a requirement becomes intricate even for a traffic scenario that is met by every driver in each day: the parking scenario.

Generally, the parking control problem for autonomous vehicles can be defined as: *finding a control strategy that guides an autonomous vehicle to move from the given initial position and orientation to the given final position and orientation* [1].

Usually, a parking controller adopts a step-by-step control strategy which compares the current state (position/orientation) with the given final state (position/orientation) to choose the steering action at the current stage [4]–[8]. However, most existing approaches in this direction use heuristic rules to determine the steering actions so that it is hard to prove whether the selected action is valid or optimal. Nevertheless, if obstacles exist around the parking berth, it is sometimes difficult to choose the right steering actions, merely using heuristic rules.

To solve this problem, increasing researchers resort to indirect trajectory planning method [9]–[17] which considers an equivalent formation of the parking control problem as: *finding a valid trajectory following along which an autonomous vehicle can track so as to move from the given initial position and orientation to the given final position and orientation* [1].

Clearly, a special parking control strategy exactly determines a parking trajectory, and vice versa. Therefore, if the valid trajectory is planned, we can then determine a sequence of steering actions along with this trajectory. Noticing there may exist an infinite number of trajectories that link the given initial state to the given final state, we often require to find a trajectory with shortest length or shortest time.

The major problem of these trajectory planning methods is that we must fully consider vehicle dynamics heavily affecting the geometry property of the candidate trajectories. Unlike many mobile robot control problems in which the steering angle can easily vary 360 degrees, the steering angle of an autonomous vehicle is limited. Hence, many movements that can be made for a small mobile robot cannot be reproduced by an autonomous vehicle.

Most existing indirect trajectory planning approaches attack this problem by designing a reference parking trajectory that an autonomous vehicle could approximately follow [9]–[17]. Such a reference trajectory usually belongs to a set of special curves (e.g. polynomial curves [1], β-spline curves [14]) which have distinctive geometric properties to simplify the presentation and planning of desired trajectories. The gap between the reference parking trajectory and the actual trajectory (produced by applying the steering actions that correspond with the reference parking trajectory) will be narrowed by adding a certain negative feedback controller to reshape the steering actions and make the vehicle roughly track this reference trajectory.

However, such trajectory planning is still indirect and sometimes troublesome. The gap between the designed parking trajectories and the actual trajectories of vehicles may be too large in some parts of the trajectory. These deviations may cause the vehicles to collide with the obstacles, especially when reverse parking actions are needed.

To fix this problem, a direct trajectory planning method was first proposed in [18]. The key idea of this novel solution is to

Manuscript received December 6, 2016; accepted March 10, 2017. This work was supported by the National Natural Science Foundation of China under Grant 91520301. The Associate Editor for this paper was P. Kachroo. *(Corresponding author: Li Li.)*

W. Liu is with the Tsinghua National Laboratory for Information Science and Technology, Department of Automation, Tsinghua University, Beijing 100084, China.

Z. Li is with the Tsinghua National Laboratory for Information Science and Technology, Department of Automation, Tsinghua University, Beijing 100084, China, and also with the Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China.

L. Li is with the Tsinghua National Laboratory for Information Science and Technology, Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: li-li@tsinghua.edu.cn).

F.-Y. Wang is with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China.

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.2017.2687047

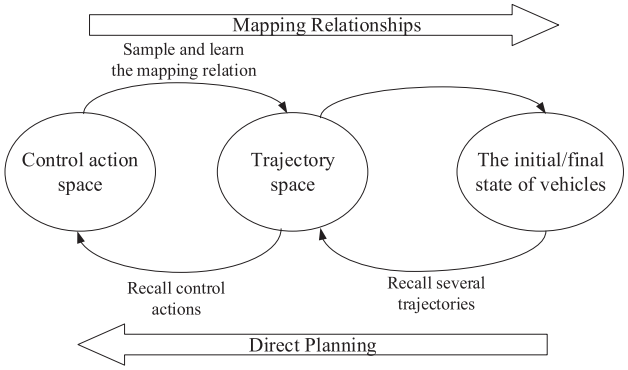


Fig. 1. The solution framework of direct trajectory planning for autonomous parking.

enumerate all the possible parking trajectories that a vehicle can make and learn to set up a direct relationship between any initial/final state pair and the corresponding steering actions as well as parking trajectory. Based on this learning result, the autonomous vehicle will soon recall the desired steering action/parking trajectory, if an initial/final state pair is given; see Fig. 1 for an illustration.

According to our survey, many mature human drivers said that by just witnessing the position/orientation of the final parking berth, they will know the following steering actions to park a vehicle before they really begin to park. This fact indicates that human drivers often recall the right parking trajectory in their mind before parking.

The direct trajectory planning method proposed in [18] is indeed an imitation of a human driver. The core of this approach is to correctly establish the relationship between an initial/final state pair and the parking trajectory. A neural network is proposed in [18] to achieve this goal, similar to what had been used in the famous Alphago go player recently proposed in [19], which uses a deep neural network to learn all the possible scenarios in the the game of go so as to find the appropriate move in a given stage. Fortunately, the solution space of parking is much smaller than the solution space of go, so we do not need to build a supercomputer like Alphago to learn the relationship.

However, the neural network proposed in [18] is still very complicated to be used on board. So, we further analyze the parking problem and find it can be solved in a multi-stage style.

The trajectory planning method in [18] is a single-stage planning style, which recalls the accurate whole trajectory for parking. Clearly, the size of the solution space increases with the time length of the trajectory. If we want to sample the solution space with an acceptable resolution level, the computation cost might be huge.

However, we find there is no need to determine the accurate whole trajectory. Actually, the second half of a trajectory is most important than the first half in moving the vehicle to the desired parking berth. So, we can use a coarse resolution level to sample the first half of a trajectory and then a finer resolution level to sample the second half of a trajectory. This will greatly reduce the computation cost of direct parking trajectory planning.

Another prominent feature of direct planning method is that we adopt a dynamic model rather than a kinematic model to depict vehicle movements. This choice is based on the following considerations:

First, we aim to build a general trajectory planning method not only for the parking problem but also for other vehicle motion planning/control problems (e.g. lane change planning) for autonomous vehicles. Different from kinematic approaches, the proposed solution framework allows large vehicle speeds and can be extended to other vehicle motion planning problems.

Second, the complexity of the direct trajectory planning method is lower than conventional indirect trajectory planning methods that use kinematic models for vehicle movements. In a kinematic approach, we need to first plan the trajectory and then design a controller to make vehicle track the planned trajectory as close as possible. The design cost of an appropriate controller increases the calculation complexity noticeably. While in a direct planning approach, the design cost of a proper controller is saved.

Third, it is hard to make vehicles strictly track the ideal trajectories obtained in indirect trajectory planning methods.

Fourth, using the direct planning method, we can explicitly consider vehicle dynamic constraints rather than consider vehicle kinematic models that require vehicles to move at a very slow speed. So, in some situations, we may choose a certain trajectory with relatively high speed to dock a vehicle in a much higher speed (and meanwhile in a much shorter time) to the parking berth. This function is very useful in some applications.

Besides, we also discuss how to counter with trajectory planning errors that are caused by model uncertainty. We show that an appropriate combination of feedforward trajectory planning and online feedback control can solve such problems.

To give a better explanation of our new finding, the rest of this short paper is arranged as follows. *Section II* explains first how to sample the trajectory space, then how to learn the relationship, and finally how to make a direct parking trajectory planning. *Section III* discusses some important problems, e.g. how to use empirical trajectory databases to avoid errors introduced by un-modeled vehicle dynamics. Finally, *Section IV* briefly concludes the paper.

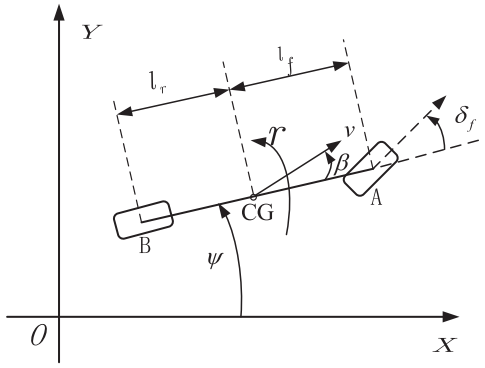
II. The Trajectory Planning Solution

A. Vehicle Dynamics Model for Control-Trajectory Mapping

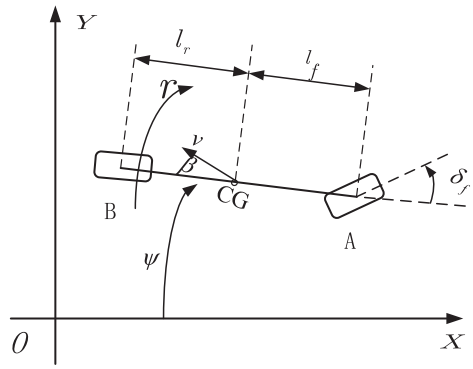
Without losing generality, we consider the parking problems of front steering vehicles and the parking problems of full steering vehicles can be solved in a similar manner [1], [20], [21].

To explain how to sample trajectory solution space, we adopt the well-known bicycle model to describe the dynamics of a vehicle [22]. Suppose a vehicle is operating on a flat surface, the bicycle model is characterized by the following variables shown in Fig. 2 and also Table. I.

Here, the *reference point* ***center of the gravity (CG)*** is chosen at the center of gravity of the vehicle body. Its coordinate vale (x, y) represents the position of the vehicle. ***Vehicle***



(a)



(b)

Fig. 2. The bicycle model of the front steering vehicle: (a) vehicle goes forward; (b) vehicle goes backward.

velocity v is defined at the *reference point* ***CG***. ***Heading Angle*** ψ refers to the angle from the X-axis to the longitudinal axis of the vehicle body ***AB***. ***Slide-slip Angle*** β is the angle from the longitudinal axis of the vehicle body ***AB*** to the direction of the vehicle velocity.

According to [20]–[22], the state space model of the bicycle model can be written as

$$\begin{bmatrix}\dot{\beta}\\ \dot{r}\end{bmatrix}=\begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}\begin{bmatrix}\beta\\ r\end{bmatrix}+\begin{bmatrix}b_1\\ b_2\end{bmatrix}\delta_f \tag{1}$$

with coefficients

$$a_{11}=-\frac{c_f+c_r}{mv}a_{12}=-1-\frac{c_f l_f-c_r l_r}{mv^2} \tag{2}$$

$$a_{21}=-\frac{c_f l_f-c_r l_r}{I}a_{22}=-\frac{c_f {l_f}^2+c_r {l_r}^2}{Iv} \tag{3}$$

$$b_1=\frac{c_f}{mv}b_2=\frac{c_f l_f}{I} \tag{4}$$

The world coordinate is set up as follows: the origin is set at the ***CG*** of the vehicle at the initial time; the direction of the positive X-axis is assumed to point to the head of the vehicle. Let v_x and v_y be the projection of v onto the X and Y axes, respectively. Referring to the Fig. 2, if the vehicle goes forward, we can get

$$\begin{aligned}v_x&=v\cos(\beta+\psi)\\ v_y&=v\sin(\beta+\psi)\end{aligned} \tag{5}$$

Otherwise, if the vehicle goes backward, we have

$$\begin{aligned}v_x&=-v\cos(\beta+\psi)\\ v_y&=v\sin(\beta+\psi)\end{aligned} \tag{6}$$

TABLE I

NOMENCLATURES AND REPRESENTATIVE VALUES

Symbol	Meaning	Value
$X-Y$	The world coordinate system	
β	Vehicle sideslip angle which is between the heading of the vehicle and the velocity of its ***CG***	
ψ	The angle from the X-axis to the longitudinal axis of the vehicle body ***AB***.	
r	Yaw rate of the vehicle, $r=\dot{\psi}$	
δ_f	Front steering angle	
v	The velocity of the vehicle	
δ_{max}	The max of the front steering angle	$0.6rad$
m	The mass of the vehicle	$1500kg$
I	The moment of inertia around the vertical axis through ***CG***	$2500kg\cdot m^2$
l_f	Distance from point ***A*** and point ***CG***	$1.20m$
l_r	Distance from point ***B*** and point ***CG***	$1.50m$
c_f	Stiffness coefficients of front tire	$80000N/rad$
c_r	Stiffness coefficients of rear tire	$80000N/rad$

Based on Eq.(1)-(6), we can easily calculate the position and orientation of the vehicle during parking.

B. Sampling Control-Trajectory Mapping Relation

Based on the dynamic model, we can calculate the resulting trajectory of a vehicle, given the inputted vehicle speed (the value of velocity) and the front steering angle within a time range. This will generate a sample of the mapping relation between the inputted vehicle speed/steering angle to the trajectory. If we obtain a sufficient number of samples, we can then "*learn*" this mapping relation.

Notice that the vehicle speed is quite slow during the parking process, the starting and stopping process of the vehicle can be assumed to be instantaneous. In practice, we can choose other values of the vehicle velocity and establish more complex and richer mapping relations. For presentation simplicity, we only consider the case that the vehicle keeps $v=1$m/s throughout the whole parking process in the rest of this paper.

In this paper, we discretize the value of the control variable δ_f along the time axis. That it, all the allowable parking trajectories could be generated by assigning distinct steering sequences $\delta_f^N=\{\delta_f(1),...\delta_f(N)\}$ for N consequent time segments. During each time segment, the steering angle is chosen from a preselected steering set $S_1=\{\delta_1,\delta_2,...,\delta_K\}$ and remains constant within this time segment. Here, K is the number of steering angles we considered. Usually, we uniformly sample the allowable value range of δ_f, say $[-\delta_{max},\delta_{max}]$. Therefore, the control action space is sampled into a set I_1 consisting of K^N steering angle sequences as its elements. Enumerating allowable control sequences, we can establish a sketch of the mapping relation.

As an example in this section, we consider a vehicle with parameters given in Table. I. The size of this vehicle is assumed to be 3.7m×1.8m. Suppose the parking behavior of the vehicle must be accomplished in 12 seconds which is uniformly divided into 4 segments. So, the time interval in each segment is 3 seconds.

The steering angles are selected from the set $S_1 = \{0.6, 0.4, 0.2, 0, -0.2, -0.4, -0.6\}$. Setting $N = 4$ and $K = 7$, we can generate $7^4 = 2401$ trajectories as shown in Fig. 3a.

Fig. 3b shows a special trajectory with the steering consequence $\delta_f^4 = \{0.6, 0.4, -0.6, -0.4\}$ and Fig. 3c gives the corresponding steering angle sequence. This sequence of steering actions indeed implements a parallel parking.

C. Single-Stage Direct Trajectory Planning

As pointed out in [18], if the sampling resolution level is high enough, we can enumerate all the possible control actions and their corresponding trajectories (as well as the corresponding initial/final state pairs). This actually builds up a complete look-up table for autonomous vehicles. Each time an initial/final state pair is given, we can check this look-up table to find the desired control actions.

We call such a trajectory planning method a single-stage direct trajectory planning, because the whole control action sequence will be generated in the beginning of the parking process. The difference between indirect trajectory planning methods [9]–[17] and direct trajectory planning method [18] is whether the formulated trajectory is generated by approximated curves or by vehicle dynamics.

For numerical simplicity, we store all the found trajectories in discrete form with discretization time interval chosen as 0.1s. For the numerical example shown in Fig. 3a, we will store $2401 \times 120 = 288120$ candidate states to match for the possible final state. In other words, a trajectory will be stored as a consecutive series of states. The finally obtained trajectory set is denoted as R_1. If the distance between a given final state to any state stored in R_1 is less than a given threshold, we say this given final state can be matched by a stored trajectory. In such a situation, the required control actions can be correctly recalled.

It should be pointed out that we do not need to further divide the planning time in direct trajectory planning process. Instead, we can allow different final states correspond to the same control actions.

Since there may exist more than one trajectory that links a given OD pair, we only store the optimal trajectory/actions for each OD pair. To reach this goal, we define the performance index of a trajectory as

$$J = \int_0^\tau (v(t) + \delta_f^2(t) + c(t))\mathrm{d}t + [h(x_\tau, y_\tau, \psi_\tau) - h(x_p, y_p, \psi_p)]^2 \quad (7)$$

where τ is the terminal time, $c(t)$ is the curvature at time t along the trajectory. $(x_\tau, y_\tau, \psi_\tau)$ is the final state of the vehicle and (x_p, y_p, ψ_p) is assumed to be the state where the vehicle exactly stops at the center of the parking berth. h is the integral

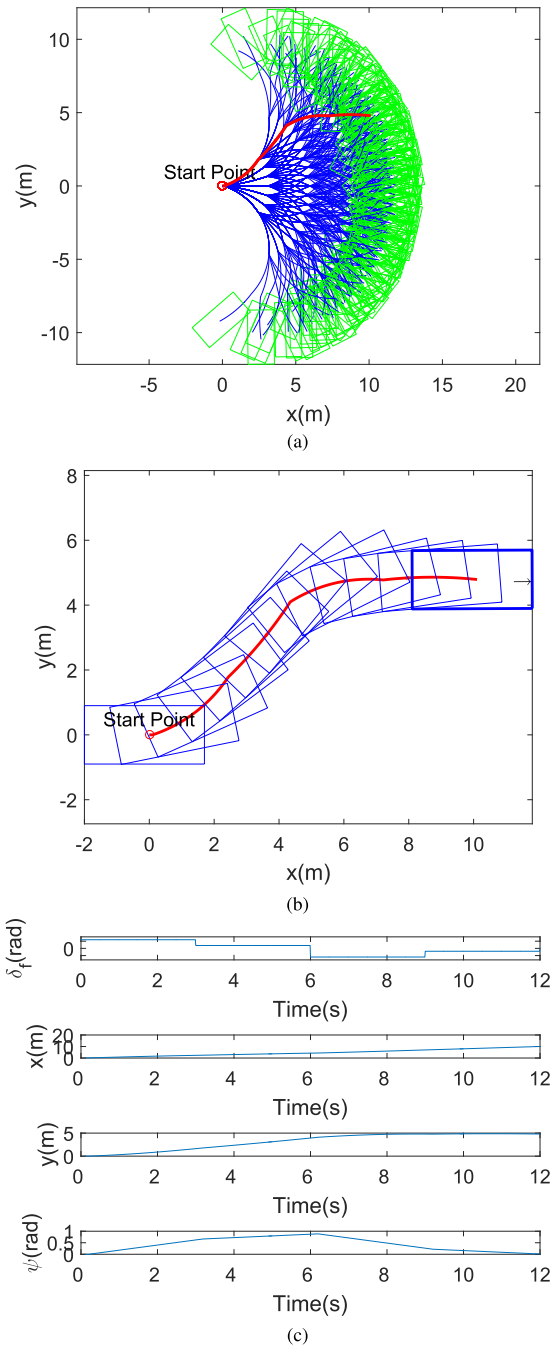


Fig. 3. An illustration of sampling control-trajectory mapping relation. (a) to make the plot clear, only about 1/10 of the generated 2401 trajectories are drawn; (b) the detailed plot of the 145th trajectory for the positions and orientations of the vehicle at each second; (c) the corresponding steering actions and the change of the vehicle state.

of the surface defined by the function $f(x, y)$ with respect to the rectangular region representing the vehicle.

$$h(x_t, y_t, \psi_t) = \iint_{D_t} f(x, y)\mathrm{d}\sigma \quad (8)$$

where D_t is the rectangular region representing the vehicle at state (x_t, y_t, ψ_t). The closer the vehicle parks to the center of the berth, the smaller the integral is.

The function $f(x, y)$ is defined in a new coordinate with its origin at the center of the parking berth and axes parallel

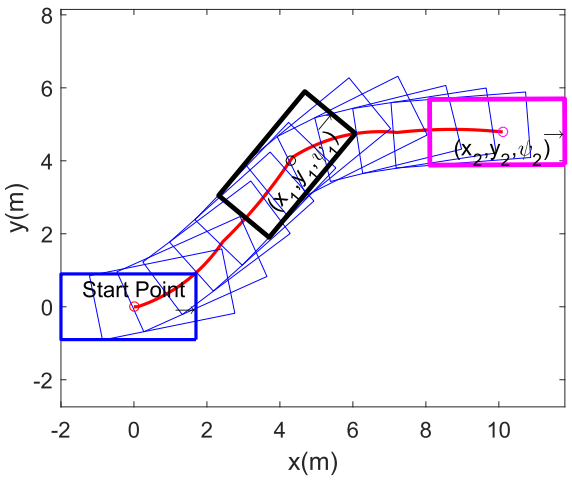


Fig. 4. An illustration of many-to-one mapping.

to the two vertical edges of the parking berth. Its formation is written as

$$f(x,y)=\begin{cases} kx, & -kx<y\le kx;\\ -kx, & kx\le y\le -kx;\\ y, & -y<kx<y;\\ -y, & y<kx\le -y. \end{cases} \tag{9}$$

where k is the ratio of the length and the width of the parking berth.

Fig. 4 gives an example of many-to-one mapping, where the final state $(x_1, y_1, \psi_1) = (4.3, 4.0, 0.9)$ plotted in the color black and the final state $(x_2, y_2, \psi_2) = (10.1, 4.8, 0)$ plotted in magenta correspond to the same steering consequence $\delta^4_{f_1} = \{0.6, 0.4, -0.6, -0.4\}$. To park the vehicle to the final state (x_1, y_1, ψ_1) rather than the final state (x_2, y_2, ψ_2), we need to monitor the parking process and stop appropriately when the vehicle reaches (x_1, y_1, ψ_1).

To implement a parking control that can precisely move a vehicle to a given final state, the size of the control action space might be huge, the storage and inquiry cost of the complete look-up table becomes the major difficulty that prevents us from applying such single-stage trajectory planning on board.

An AI based solution had been proposed in [18] to solve this problem. Like what had been suggested in [23], the look-up table was transferred into a deep neural network, which learns the mapping relation.

As shown in Fig. 5, we set the vehicle speed and final state as the inputs of this special deep neural network (suppose we had shifted the coordinates to make the initial state to zero) and the desired control actions as the output of this special neural network. Because of the compression capability of the deep neural network, we can use a much smaller storage and inquiry cost to reach the same goal.

However, the computation cost for training the deep neural network becomes a new problem. It is still hard to guarantee that the deep neural network had correctly learned the mapping relation.

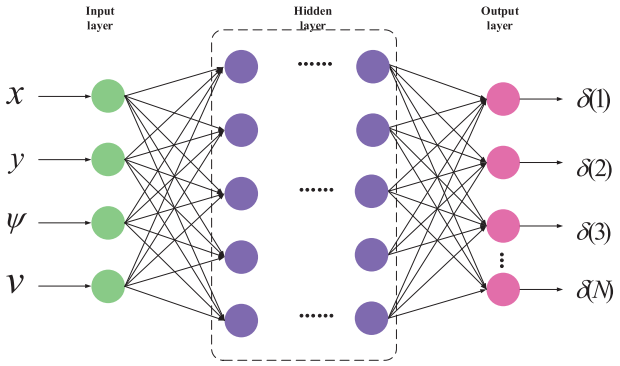


Fig. 5. The structure of the deep neural network proposed in [18].

D. Multi-Stage Direct Trajectory Planning

In this paper, we propose a new solution that is still planning based. The key idea of this solution lies in two facts:

1) *the direct trajectory planning process can be decomposed into several sub-processes and the computational cost for each sub-process can be greatly reduced;*

2) *the latter half of the parking trajectory is usually more important than the first half in precisely moving the vehicle towards the desired parking position/orientation.*

More precisely, we first establish a coarse mapping relation between control actions and their corresponding trajectories for a relatively long time period (e.g. 12s used in *Section II.B*), by using a low sampling resolution level for steering angles. We use the direct trajectory planning method (based on the coarse mapping relation) to pick up a candidate trajectory from the generated trajectory set that moves the vehicle from the initial state to a final state that is closest to the desired final state.

Then, we build a precise mapping relation between control actions and their corresponding trajectories for a relatively short time period, by using a high sampling resolution level for steering angles. Then, we start from a certain midway point of the selected coarse trajectory and execute the direct trajectory planning method (based on the precise mapping relation) to find a trajectory that moves the vehicle from the midway state to the desired final state.

If the required parking trajectory is very long, we can further decompose the trajectory planning process into multiple stages, where the last one or few sub-processes will be executed by using precise mapping relations. We call such direct trajectory planning methods: multi-stage trajectory planning.

Fig. 6 gives an example of two-stage trajectory planning for parking, in which we aim to move the vehicle with the initial state $(x, y, \psi) = (0, 0, 0)$ into a parking berth which is represented as a 2.5m×4.8m rectangular with its center coordinate and orientation $(x_p, y_p, \psi_p) = (10, 6.5, 0.6)$.

We can easily find that no steering consequences selected from the steering set $S_1 = \{0.6, 0.4, 0.2, 0, -0.2, -0.4, -0.6\}$ can perfectly reach this goal. Fig. 6a shows four closest trajectories.

To solve this problem, we generate a precise trajectory set R_2 which covers a smaller parking area; while the candidate

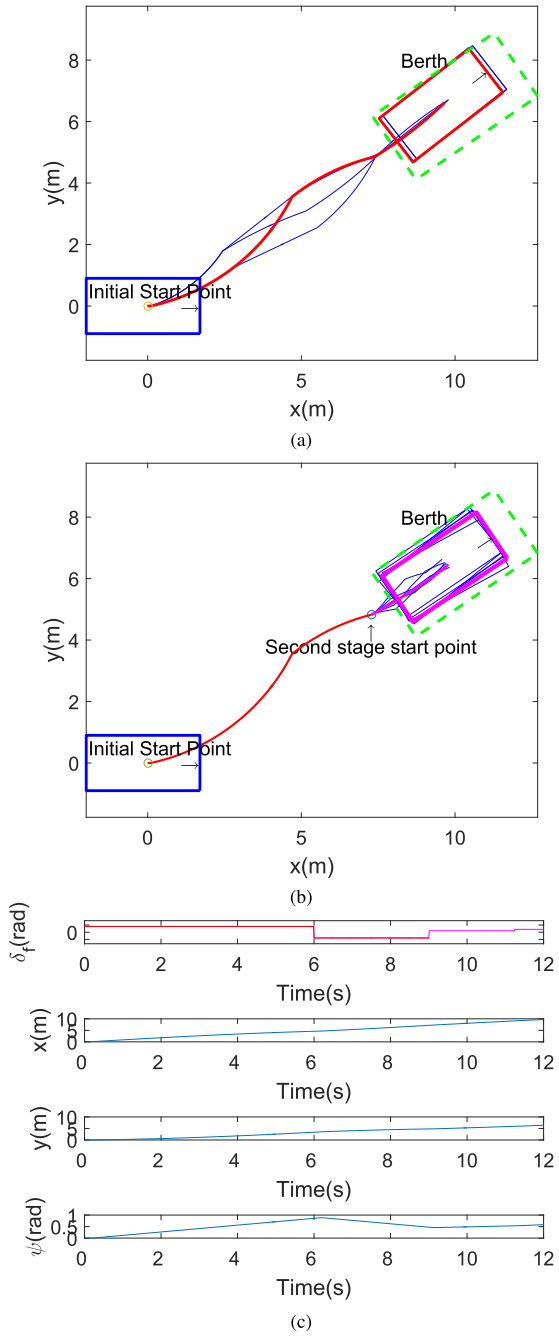


Fig. 6. (a) The closest coarse trajectory (plotted in red color) that moves the vehicle from the initial state to a final state towards the parking berth; (b) the parking trajectories found by two-stage direct trajectory planning; (c) the steering actions and the change of vehicle position/orientation that corresponds with the magenta trajectory found in (b).

trajectories are much denser than those in R_1. Similarly, the trajectory set R_2 is generated in a shorter time range that is also divided into N equal time segments. The steering angles can be selected from the set $S_2 = \{\delta_1', \delta_2', ..., \delta_L'\}$ which has much more elements than S_1. Here, L is the number of steering angles we considered in S_2.

In the following example, we consider a precise trajectory generated in 3 seconds and the steering angle set R_2 is chosen as $S_2 = \{0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0, -0.1, -0.2, -0.3, -0.4, -0.5, -0.6\}$.

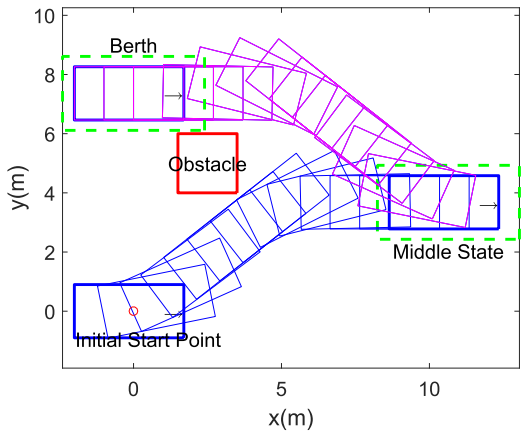


Fig. 7. A two-part parking process where the adjusting process is plotted in blue color and the docking process is plotted in magenta color.

The second-stage direct trajectory planning begins from one point of the coarse trajectory found in the first-stage direct trajectory planning, where the vehicle takes 9 seconds to reach this point in the coarse trajectory. So, the maximum parking time of a vehicle is still 12 seconds.

Fig. 6b shows several precise trajectories that can park the vehicle right into the parking berth. Fig. 6c presents the steering actions and the change of vehicle position/orientation that corresponds with the magenta trajectory found in Fig. 6b. In the second-stage, the steering consequence is chosen as $\delta_{f_2}^4 = \{0.1, 0.1, 0.1, 0.2\}$.

The major benefit of multi-stage trajectory planning is that it reduces the computational complexity for the first part of parking trajectory, since we can decompose a long-term trajectory planning problem into several short-term trajectory planning problems, most of which can be solved using coarse sampling resolution levels.

III. Further Discussions

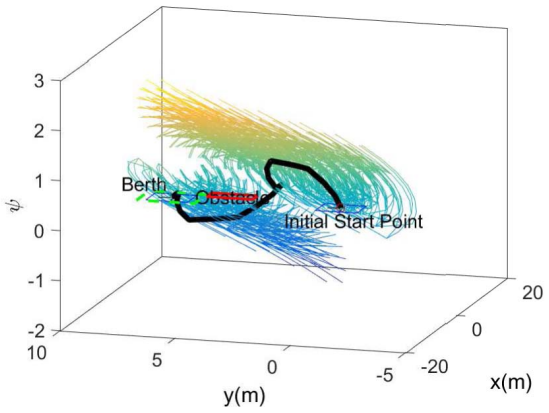
A. Dealing With Obstacles

In the above discussions, we omit the existence of obstacles and focus on the key idea of direct trajectory planning. In this subsection, we explain how to directly plan some complex trajectories to detour the obstacles.

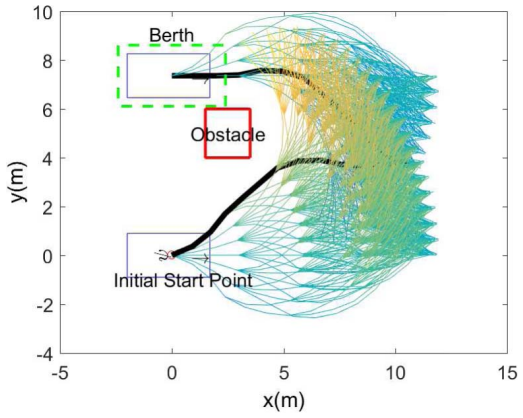
Normally, a complex parking process could be divided into several parts. For presentation simplicity, let us consider a two-part parking process, including: (1) Adjusting Process which moves the vehicle from the initial starting point to the middle point (switching point); (2) Docking Process which moves the vehicle from the middle point to the ending point in the parking berth; see Fig. 7. For even more complex parking processes, there might exist several adjusting processes.

Clearly, for a two-part parking process, we can start from both the initial state point and the ending state point to grow two trees of trajectories in the 3-dimension space of (x, y, ψ); see Fig. 8. Here, the tree of trajectories that stems from the ending state is established by shifting all the trajectories in which the vehicle goes back so as to make the final states of these trajectories overlap on the desired ending point; see Fig. 9 for an illustration.

If either two middle states in two trees overlap, we can find a common state that can be reached after apply a series of



(a)



(b)

Fig. 8. (a) 3D and (b) 2D Illustrations from different angles for the trajectory found in Fig.7.

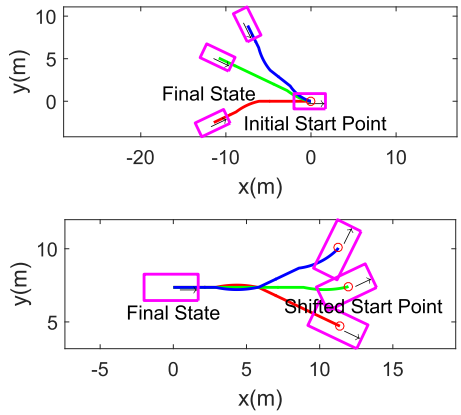


Fig. 9. An illustration of building the tree of trajectories for the ending point.

control actions. Then, we check whether we can start from the initial state, then traverse this middle point, and finally park into the parking berth, without colliding with the obstacle. If no collision is detected, we find a valid trajectory to park; see Fig. 8.

B. Replacing Analytical Models With Empirical Trajectory Records

In *Section II.A*, we use the simplified bicycle model to generate control-trajectory sample pairs. However, as pointed out in [21], [24]–[26], there exist nonlinear dynamics that cannot

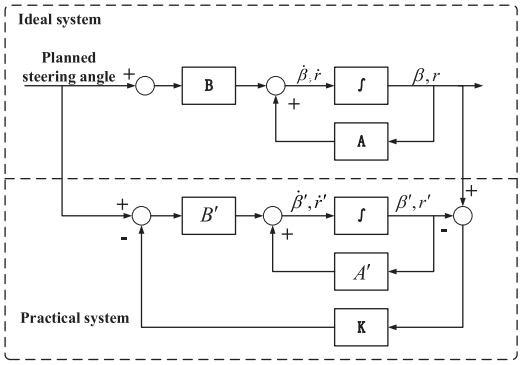


Fig. 10. A simplified control system with negative feedback control.

be accurately modeled by bicycle model. Moreover, the change of weather/road conditions, the number of passengers and other factors can all affect the values of parameters and thus influence the accuracy of the bicycle model Eq.(1)-(4). This will cause the proposed direct trajectory planning method to fail, since the sampled and stored mapping relation does not hold in practice.

To solve this problem, we may use another important merit of direct trajectory planning method. Noticing that we only need control-trajectory pairs but not control models in direct trajectory planning, we can carry out a number of experiments and store empirical control-trajectory pairs for planning. This enables us to eliminate the effect of modeling errors in practice.

In short, the direct planning method works as an end-to-end solution which aims to set up a direct link between the steering control command sequences and the trajectories. No matter the type of vehicle, the condition of the road surface, or how the vehicle performs, if there exists a certain relationship between the steering actions and the trajectories, direct planning method can learn it from practical experiments.

C. Combining Feedback Control

The model of the front steering vehicle considered in *Section II* is very simple and not robust. If the mass of the vehicle or the road condition changed, it may not function well any more. In order to precisely move the vehicle in the process of parking, we consider to add a negative feedback controller to this system.

Take the difference between the desired state and the practical state in every moment as the negative feedback signal and the planned input steering sequence as the feedforward signal, we can obtain an adjusted steering angle sequence to make the practical trajectory overlap the planned trajectory as much as possible. The simplified structure diagram of a control system combined with feedback control is given in Fig. 10.

The state space of this feedback system can be written as

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ \dot{\boldsymbol{x}}' &= A'\boldsymbol{x}' + B'[u - K(\boldsymbol{x}' - \boldsymbol{x})] \end{aligned} \tag{10}$$

where $\boldsymbol{x} = [\beta, r]^{\mathrm{T}}$ denotes the state of the ideal system, and $\boldsymbol{x}' = [\beta', r']^{\mathrm{T}}$ denotes the state of the practical system. $u = \delta_f$

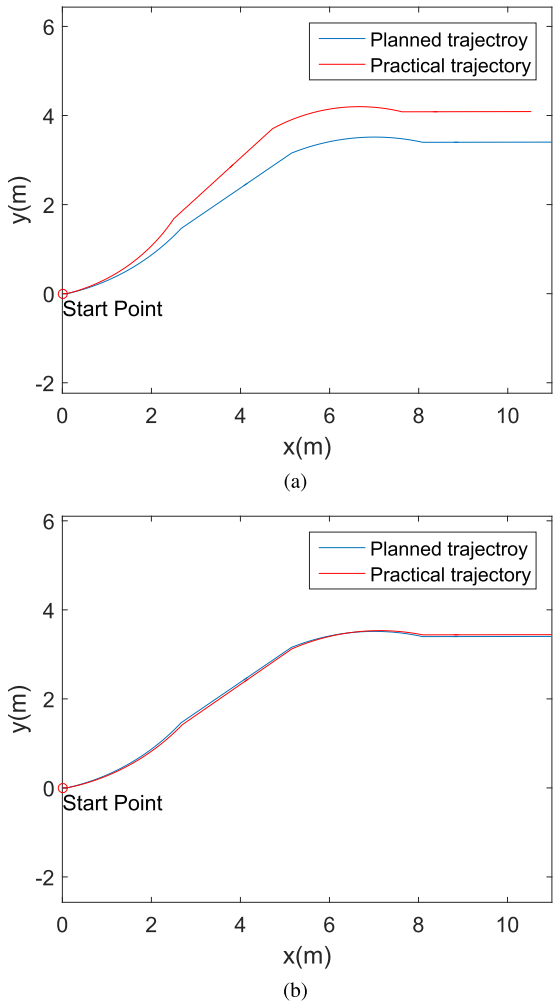


Fig. 11. The planned trajectory and the practical trajectory: (a) with just feedforward trajectory planning but no online feedback; (b) with combined feedforward trajectory planning and online feedback control.

and K is the feedback matrix. Matrices A and B are system and control matrices that can be formulated from Eq.(1)-(4).

Clearly, Eq.(10) indeed belongs to the classical robust control problems with parameter uncertainty [27]. By assigning proper feedback matrix K, we can guarantee the practical stability (convergence to a bounded error) of the tracking error $\boldsymbol{e} = \boldsymbol{x}' - \boldsymbol{x}$. Since how to design such a feedback matrix K does not fit with the main theme of this paper, we do not further discuss it here and just give a numerical example.

Suppose the stiffness coefficients of front tire used in trajectory planning is $c_f = 80000N/rad$. However, in practice the stiffness coefficients of front tire is $c'_f = 60000N/rad$. As shown in Fig. 11a, after applying the steering angle sequence $\delta_f^4 = \{0.4, 0, -0.4, 0\}$, there will be a notable gap between the planned ending state and the practical ending state. While, if we add the feedback control showed in Fig. 10 with the feedback matrix set as $K = [1, -25]$, the gap becomes so small that it can be ignored; see Fig. 11b.

In addition, it is highly suggested to monitor the tire-road friction online so as to build correct system dynamic models and thus we can generate appropriate candidate trajectories for planning. Some useful algorithms to monitor tire-road friction can be found in [1], [21], [28], [29].

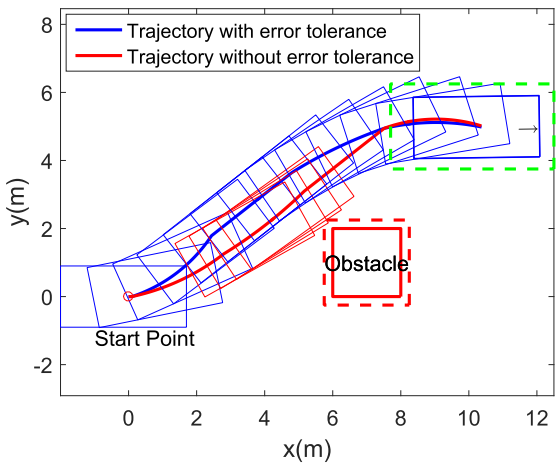


Fig. 12. An illustration on how to deal with location errors of obstacles.

D. Further Discussions

First, the direct planning method is a trading-space-for-time strategy. The time consuming task is to enumerate all the possible trajectories and pre-store them in the computer. The search then becomes quite quick.

The simulation test shown in this paper is carried out on a PC computer (installed Windows 7 and Matlab 2012b) in our laboratory. It has an Intel i7 processor, 16GB RAM, an NVIDIA GTX 650 graphics card. It took less than 0.25 seconds to find the feasible steering control sequences for all the testing cases. So, it would be acceptable, when this method is implemented by the onboard computer in the future.

Second, the main purpose of this paper is trajectory planning rather than sensing. So, we do not further discuss how to gather the information of driving environment in this paper.

As other trajectory planning methods for parking, the direct planning method assumes that the information of driving environment can be correctly obtained, e.g. e.g. using vision based methods proposed in [30]–[34] or V2X assistance proposed in [35]. If there does exist uncertainty in the positions of vehicles/berth/obstacles, we can choose a conservative trajectory to detour, which ensures the vehicle avoids the obstacles without collision, considering the possible location errors.

For example, as shown in Fig. 12, we can expand the outline of the obstacle for 50 centimeters (see the square of red dashed line), if the maximum location error is 50 centimeters. Then, we can carry out trajectory planning by considering the inflated obstacle instead. All the trajectories that may collide with the inflated obstacle will not be adopted.

Third, we do not limit the length of the parking trajectory in the range of about 10 meters. We take 12 seconds as the parking time limit in this paper, because it is usually enough for successfully parking a car.

For very long-distance parking planning, we need to first construct a series of midway turning points and then decompose the whole planning problem into a series of trajectory

planning within smaller range. We will finally combine the obtained trajectories within every two adjacent turning points to form a whole long trajectory and a complete steering control sequence.

For example, in the example shown in *Section III.A*, the whole parking process is decomposed into two parts in which two short trajectories are constructed respectively. The total length of the overall trajectory is about 24 meters not just about 10 meters.

IV. Conclusions

Parking bothers many novice drivers, mainly because these novice drivers have not learned the precise relationship between their steering actions during a time period and the vehicles' final position/orientation. Many existing methods solve this problem in an indirect and separate way. First, we design a trajectory that links the beginning and ending states of vehicles. Second, we design a controller to guarantee the vehicle moves along the designed trajectory to its final position. So, the whole planning process is often time-consuming.

In this paper, we address the issues related to mimicking human driving behaviors to solve the parking problem. Based on the study of vehicle dynamics, the (inverse) mapping relationships between control actions, and the corresponding trajectories, the initial/final states are established. This enables us to directly consider implicit dynamic constraints of vehicles.

From the mapping relation, the required parking trajectory can be easily recalled using a multi-stage planning method. Tests prove the high reliability and efficiency of the proposed method. In addition, we show that an appropriate combination of feedforward trajectory planning and online feedback control can reduce trajectory planning errors that are caused by model uncertainty.

Outstripping some parking algorithms that generate abstract control strategies to guide a vehicle, the new method provides huaman drivers concrete control guiding rules like "turn θ degree to left at time T", which is preferred by most human drivers. This indicates that we could also use this method to build advanced driver assistance systems for human drivers.

It should also be pointed out that the proposed indirect trajectory planning algorithm can be viewed as a special case of the so called "parallel learning" strategy that was proposed in [36], [37], [38], [39], [40]. One key idea of "parallel learning" strategy is to generate a number of data samples (special trajectory-action pairs in this paper) under a semi-supervised manner and then "learn" from these data samples via recently developing machine learning models (e.g. deep neural networks or reinforcement learning). As pointed out in [41], such predict-and-learn technique may help solve many difficult problems in practice.

References

[1] L. Li and F.-Y. Wang, *Advanced Motion Control and Sensing for Intelligent Vehicles*. Springer, 2007.

[2] A. Eskandarian, *Handbook of Intelligent Vehicles*. London, U.K.: Springer, 2012.

[3] T. J. Gordon and M. Lidberg, "Automated driving and autonomous functions on road vehicles," *Veh. Syst. Dyn.*, vol. 53, no. 7, pp. 958–994, 2015.

[4] S.-G. Kong and B. Kosko, "Adaptive fuzzy systems for backing up a truck-and-trailer," *IEEE Trans. Neural Netw.*, vol. 3, no. 2, pp. 211–223, Mar. 1992.

[5] H. Ichihashi and M. Tokunaga, "Neuro-fuzzy optimal control of backing up a trailer truck," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 1. Mar./Apr. 1993, pp. 306–311.

[6] K. Tanaka, T. Kosaki, and H. O. Wang, "Backing control problem of a mobile robot with multiple trailers: Fuzzy modeling and LMI-based design," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 3, pp. 329–337, Aug. 1998.

[7] M. Omae, H. Shimizu, and T. Fujioka, "GPS-based automatic driving control in local area with course of large curvature and parking space," *Veh. Syst. Dyn.*, vol. 42, nos. 1–2, pp. 59–73, 2010.

[8] S.-J. Huang and G.-Y. Lin, "Parallel auto-parking of a model vehicle using a self-organizing fuzzy controller," *Proc. Inst. Mech. Eng. D, Transp. Eng.*, vol. 224, no. 8, pp. 997–1012, 2010.

[9] A. Ohata and M. Mio, "Parking control based on nonlinear trajectory control for low speed vehicles," in *Proc. IEEE Int. Conf. Ind. Electron., Control Instrum. (IECON)*, Oct. 1991, pp. 107–112.

[10] D. Gorinevsky, A. Kapitanovsky, and A. Goldenberg, "Neural network architecture for trajectory generation and control of automated car parking," *IEEE Trans. Control Syst. Technol.*, vol. 4, no. 1, pp. 50–56, Jan. 1996.

[11] G. Chen and D. Zhang, "Back-driving a truck with suboptimal distance trajectories: A fuzzy logic control approach," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 3, pp. 369–380, Aug. 1997.

[12] M. Wada, K. S. Yoon, and H. Hashimoto, "Development of advanced parking assistance system," *IEEE Trans. Ind. Electron.*, vol. 50, no. 1, pp. 4–17, Feb. 2003.

[13] B. Müller, J. Deutscher, and S. Grodde, "Continuous curvature trajectory design and feedforward control for parking a car," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 3, pp. 541–553, May 2007.

[14] F. Gómez-Bravo, F. Cuesta, A. Ollero, and A. Viguria, "Continuous curvature path generation based on β-spline curves for parking manoeuvres," *Robot. Auto. Syst.*, vol. 56, no. 4, pp. 360–372, 2008.

[15] J. Moon, I. Bae, J.-G. Cha, and S. Kim, "A trajectory planning method based on forward path generation and backward tracking algorithm for automatic parking systems," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 719–724.

[16] H. Vorobieva, S. Glaser, N. Minoiu-Enache, and S. Mammar, "Automatic parallel parking in tiny spots: Path planning and control," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 396–410, Feb. 2015.

[17] X. Du and K. K. Tan, "Autonomous reverse parking system based on robust path generation and improved sliding mode control," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1225–1237, Jun. 2015.

[18] L. Li and F.-Y. Wang, "Parking guidance system for front wheel steering vehicles using trajectory generation," in *Proc. IEEE Intell. Transp. Syst.*, vol. 2. Oct. 2003, pp. 1770–1775.

[19] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[20] J. Ackermann, J. Guldner, W. Sienel, R. Steinhauser, and V. I. Utkin, "Linear and nonlinear controller design for robust automatic steering," *IEEE Trans. Control Syst. Technol.*, vol. 3, no. 1, pp. 132–143, Mar. 1995.

[21] L. Li, F.-Y. Wang, and Q. Zhou, "Integrated longitudinal and lateral tire/road friction modeling and monitoring for vehicle motion control," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 1–19, Mar. 2006.

[22] G. Rill, *Road Vehicle Dynamics: Fundamentals and Modeling*. Boca Raton, FL, USA: CRC Press, 2011.

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[24] J. Stephant, A. Charara, and D. Meizel, "Virtual sensor: Application to vehicle sideslip angle and transversal forces," *IEEE Trans. Ind. Electron.*, vol. 51, no. 2, pp. 278–289, Apr. 2004.

[25] T. Qu, H. Chen, D. Cao, H. Guo, and B. Gao, "Switching-based stochastic model predictive control approach for modeling driver steering skill," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 365–375, Feb. 2015.

[26] H. Zhang and J. Wang, "Vehicle lateral dynamics control through AFS/DYC and robust gain-scheduling approach," *IEEE Trans. Veh. Technol.*, vol. 65, no. 1, pp. 489–494, Jan. 2016.

[27] J. C. Doyle, K. Glover, P. P. Khargonekar, and B. A. Francis, "State-space solutions to standard H_2 and H_∞ control problems," *IEEE Trans. Autom. Control*, vol. 34, no. 8, pp. 831–847, Aug. 1989.

[28] L. Li, G. Jia, X. Ran, J. Song, and K. Wu, "A variable structure extended Kalman filter for vehicle sideslip angle estimation on a low friction road," *Veh. Syst. Dyn.*, vol. 52, no. 2, pp. 280–308, 2014.

[29] L. Li, K. Yang, G. Jia, X. Ran, J. Song, and Z.-Q. Han, "Comprehensive tire–road friction coefficient estimation based on signal fusion method under complex maneuvering operations," *Mech. Syst. Signal Process.*, vols. 56–57, pp. 259–276, May 2015.
[30] Y. Suzuki, M. Koyamaishi, T. Yendo, T. Fujii, and M. Tanimoto, "Parking assistance using multi-camera infrastructure," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2005, pp. 106–111.
[31] L. Li, J. Song, F.-Y. Wang, W. Niehsen, and N.-N. Zheng, "IVS 05: New developments and research trends for intelligent vehicles," *IEEE Intell. Syst.*, vol. 20, no. 4, pp. 10–14, Jul. 2005.
[32] S. Li and Y. Hai, "Easy calibration of a blind-spot-free fisheye camera system using a scene of a parking space," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 1, pp. 232–242, Mar. 2011.
[33] C. Wang, H. Zhang, M. Yang, X. Wang, L. Ye, and C. Guo, "Automatic parking based on a bird's eye view vision system," *Adv. Mech. Eng.*, vol. 6, p. 847406, Mar. 2014.
[34] S.-E. Shih and W.-H. Tsai, "A convenient vision-based system for automatic detection of parking spaces in indoor parking lots using wide-angle cameras," *IEEE Trans. Veh. Technol.*, vol. 63, no. 6, pp. 2521–2532, Jul. 2014.
[35] X. Cheng *et al.*, "Electrified vehicles and the smart grid: The ITS perspective," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1388–1404, Aug. 2014.
[36] N. Zhang, F. Y. Wang, F. Zhu, D. Zhao, and S. Tang, "DynaCAS: Computational experiments and decision support for ITS," *IEEE Intell. Syst.*, vol. 23, no. 6, pp. 19–23, Nov. 2008.
[37] F.-Y. Wang, "Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 630–638, Sep. 2010.
[38] L. Li and D. Wen, "Parallel systems for traffic control: A rethinking," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1179–1182, Apr. 2016.
[39] L. Li, Y. Lin, D. Cao, N.-N. Zheng, and F.-Y. Wang, "Parallel learning-a new framework for machine learning," *ACTA Autom. Sinica*, vol. 43, no. 1, pp. 1–8, 2017.
[40] F.-Y. Wang, J. Zhang, Q. Wei, X. Zheng, and L. Li, "PDP: Parallel dynamic programming," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 1, pp. 1–5, Jan. 2017.
[41] Y. Lecun, "Predictive learning," in *Proc. Speech NIPS*, 2016.

Wei Liu is working toward the Ph.D. degree with the Department of Automation, Tsinghua University, Beijing, China. He mainly studies intelligent vehicle and electric vehicle design.

Zhiheng Li (M'05) received the Ph.D. degree in control science and engineering from Tsinghua University, Beijing, China, in 2009. He is an Associate Professor with the Department of Automation, Tsinghua University, and the Graduate School at Shenzhen, Tsinghua University, Shenzhen, China. His research interests include traffic operation, advanced traffic management system, urban traffic planning, and intelligent transportation systems. He serves as an Associate Editor of IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.

Li Li (S'05–M'06–SM'10) is an Associate Professor with the Department of Automation, Tsinghua University, Beijing, China, where he is involved in the fields of complex and networked systems, intelligent control and sensing, intelligent transportation systems, and intelligent vehicles. He has authored over 50 SCI indexed international journal papers and over 50 international conference papers as a first/corresponding author. He serves as an Associate Editor of IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.

Fei-Yue Wang (S'87-M'89-SM'94-F'03) received the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990. He joined University of Arizona in 1990 and became a Professor and the Director of the Robotics and Automation Laboratory and the Program in Advanced Research for Complex Systems. In 1999, he founded the Intelligent Control and Systems Engineering Center at the Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, China, under the support of the Outstanding Overseas Chinese Talents Program from the State Planning Council and the 100 Talent Program from CAS. In 2002, he was appointed as the Director of the Key Laboratory of Complex Systems and Intelligence Science, CAS. In 2011, he became the State Specially Appointed Expert and the Director of the State Key Laboratory of Management and Control for Complex Systems.

His research focuses on methods and applications for parallel systems, social computing, and knowledge automation. He is Fellow of INCOSE, IFAC, ASME, and AAAS. In 2007, he received the 2nd Class National Prize in Natural Sciences of China and the Outstanding Scientist by ACM for his work in intelligent control and social computing. He received the IEEE ITS Outstanding Application and Research Awards in 2009 and 2011, respectively. In 2014, he received the IEEE SMC Society Norbert Wiener Award. He was the Founding Editor-in-Chief (EiC) of *International Journal of Intelligent Control and Systems* from 1995 to 2000, the Founding EiC of the *IEEE ITS Magazine* from 2006 to 2007, the EiC of *IEEE Intelligent Systems* from 2009 to 2012, and the EiC of IEEE TRANSACTIONS ON ITS from 2009 to 2016. He is currently the EiC of IEEE/CAA JOURNAL OF AUTOMATICA SINICA and *Chinese Journal of Command and Control*. Since 1997, he has served as the General or Program Chair of over 20 IEEE, INFORMS, ACM, and ASME conferences. He was the President of the IEEE ITS Society from 2005 to 2007, the Chinese Association for Science and Technology, USA, in 2005, and the American Zhu Kezhen Education Foundation from 2007 to 2008, and the Vice President of the ACM China Council from 2010 to 2011. Since 2008, he has been the Vice President and the Secretary General of the Chinese Association of Automation.