Advances in Engineering Software 87 (2015) 30–42

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Simultaneous dynamic optimization: A trajectory planning method for nonholonomic car-like robots

Bai Li [a], Zhijiang Shao [a,b,*]

[a] Department of Control Science & Engineering, Zhejiang University, Hangzhou 310027, PR China
[b] State Key Laboratory of Industrial Control Technology, Hangzhou, PR China

ARTICLE INFO

Article history:
Received 30 December 2014
Received in revised form 13 April 2015
Accepted 26 April 2015

Keywords:
Car-like robot
Trajectory planning
Time-optimal control
Simultaneous dynamic optimization
Interior-point method
Computational guidance and control

ABSTRACT

Trajectory planning in robotics refers to the process of finding a motion law that enables a robot to reach its terminal configuration, with some predefined requirements considered at the same time. This study focuses on planning the time-optimal trajectories for car-like robots. We formulate a dynamic optimization problem, where the kinematic principles are accurately described through differential equations and the constraints are strictly expressed using algebraic inequalities. The formulated dynamic optimization problem is then solved by an interior-point-method-based simultaneous approach. Compared with the prevailing methods in the field of trajectory planning, our proposed method can handle various user-specified requirements and different optimization objectives in a unified manner. Simulation results indicate that our proposal efficiently deals with different kinds of physical constraints, terminal conditions and collision-avoidance requirements that are imposed on the trajectory planning mission. Moreover, we utilize a Hamiltonian-based optimality index to evaluate how close an obtained solution is to being optimal.

1. Introduction

Robotic trajectory planning refers to the process of finding a motion law to enable a robot to move from its initial configuration to a desired terminal configuration by simultaneously considering some predefined requirements [1]. Trajectory planning has become a critical aspect of automation science; the applications of this process range from satellite orbit transfer [2], missile guidance [3], unmanned aerial vehicle navigation [4–6], anti-submarine search [7,8], hazardous environment exploration [9] and autonomous parking assistance [10,11]. In this work, we focus on the trajectory planning of car-like robots.

A car-like robotic model can represent a wide range of vehicles with steering wheels and power motors [12], which may be the reason why such robots have been widely studied in the past two decades [13,14]. Although some relevant commercial productions are available, they can still be improved substantially regarding smartness and autonomy. In fact, many key parts of real-world deployment phases are ad hoc and manual [15]. Ref. [16] even pointed out that there had been few artificial systems that could outperform an experienced human driver (who also can be regarded as a system, although imperfect, yet the best one known). Therefore, further investigations are necessary to develop smart decision-making capability in robots.

Typically a car-like robot is a nonholonomic system [17]. A system is said to be holonomic if all the system constraints can be expressed in the form of $f(q,t)=0$, where q denotes the system states. By contrast, if the system constraints are written as $f(q,\dot{q},\ddot{q},t)=0$ but cannot be reduced to $f(q,t)=0$, then the system is nonholonomic [18]. In a non-holonomic system, the differential equations that describe robotic kinematics are non-integrable, which causes the total degrees of freedom to become greater than the controllable ones and eventually increases the difficulty of the trajectory planning scheme [19].

A considerable number of studies have been undertaken to plan trajectories for mobile robots. Studies utilize geometrics originate from the pioneering works of Dubins [20] and Reeds & Shepp [21], where admissible trajectories consist of circle arcs with minimal turning radii and line segments. In general, those geometric approaches calculate reference paths during the first step; then, additional effort is required to generate control motions following the obtained reference paths [22,23]. Apart from the geometric methods, knowledge-based approaches have also been developed.

* Corresponding author at: Department of Control Science & Engineering, Zhejiang University, Hangzhou 310027, PR China
E-mail addresses: libai@zju.edu.cn, libaioutstanding@163.com (B. Li), szj@zju.edu.cn (Z. Shao).

http://dx.doi.org/10.1016/j.advengsoft.2015.04.011
0965-9978/

Gorinevsky et al. introduced neural network technology to generate autonomous parking motions [24], whereas Marin applied reinforcement learning technique to grasp the kinematic principles and then to generate driving behaviors [25]. Zhao and Collins developed a fuzzy logic approach for parallel parking [26]. Khoukhi trained a neuro-fuzzy inference system to capture the dynamic behaviors of a robot [27]. In addition to the aforementioned studies, sampling-based and search-based methods have also caught the interest of researchers. Sampling-based methods compute trajectories through random sampling; two typical examples of these methods are the probabilistic roadmap method [28] and the rapidly exploring random trees [29–31]. Search-based methods, on the other hand, look for admissible paths or trajectories in predefined solution spaces [32–35]. Moreover, there have also been some previous studies that planned trajectories through control theories. Kim and Kim utilized the bang–bang control principle to induce satisfactory trajectories under the two-corner scenarios [36]. Barraquand and Latombe applied differential geometric control theory to compute maneuver-optimal solutions [17]. Balkcom and Mason applied Pontryagin's maximum principle to calculate time-optimal trajectories [37].

According to prevailing and emerging literature related to trajectory planning, judging whether one approach is better than another is difficult mainly because different methods are applied in certain scenarios or based on various kinematic models. Despite these differences, the authors believe that studying optimal trajectories based on a given optimization criterion is better than merely generating feasible trajectories [11]. Moreover, transforming the original trajectory planning scheme into another kind of problem is inappropriate if they are not equivalent. In this study, we regard trajectory planning as a dynamic optimization problem, in which robotic kinematics and additional requirements (mechanical constraints and collision-avoidance conditions) are strictly described. At present, numerical methods may be the only efficient approach to solve a complicated dynamic optimization problem.

The simultaneous approach is a numerical method that addresses dynamic optimization problems. Through discretization, all state and control profiles in time are described by collocations in a finite number of elements. In this manner, the original infinite-dimensional dynamic optimization problem is transformed into a finite-dimensional nonlinear programming (NLP) formulation. The simultaneous approach is equivalent to a fully implicit Runge–Kutta method with high order accuracy and excellent stability. It possesses various merits over its competitors when tackling dynamic optimization problems, especially ones with complicated constraints and with input instabilities [38]. Since the resulting finite-dimensional NLP problem is in large scale, it calls for a highly efficient NLP solver. The interior-point method (IPM) is an effective solver that addresses high-dimensional NLPs. IPM applies a logarithmic barrier method to inequality constraints in an NLP, solves a set of equality constrained optimization problems for a monotonically decreasing sequence of the barrier parameter, and then rapidly converges to a solution [39]. This mechanism enables IPM to solve NLPs with up to several million variables, constraints, and degrees of freedom. The IPM-based simultaneous approach has been widely used in many chemical engineering applications; however, its applications in robotics are relatively scarce.

In this work, we choose terminal moment as the minimization criterion in the formulated dynamic optimization problem with kinematics and other constraints that are strictly described. The IPM-based simultaneous approach is applied to solve this problem. Several sets of benchmark scenarios are systematically designed to test the performance of the dynamic optimization formulation and the solver. A Hamiltonian-based index is also proposed to reflect how close the obtained solutions are to optimality.

2. Problem formulation

This section focuses on describing the original trajectory planning scheme through differential equations and algebraic equalities/inequalities. The kinematic principles of a car-like vehicle are presented, along with the interior constraints (i.e., the physical and mechanical restrictions of a robot) and the exterior constraints (e.g., collision-free and two-point boundary conditions) that should be satisfied during the entire movement of a car. At the end of this section, we provide an overall framework of the formulated dynamic optimization problem, which we believe, is identical to the original trajectory planning mission.

2.1. Kinematic principles of a car-like robot

The kinematics of a front-steering car-like vehicle can be expressed as follows [11]:

$$\begin{cases} \frac{dx(t)}{dt} = v(t)\cdot\cos\theta(t) \\ \frac{dy(t)}{dt} = v(t)\cdot\sin\theta(t) \\ \frac{dv(t)}{dt} = a(t) \\ \frac{d\theta(t)}{dt} = \frac{v(t)\cdot\tan\phi(t)}{l} \\ \frac{d\phi(t)}{dt} = \omega(t) \end{cases}, \quad t\in[t_0,t_f], \tag{1}$$

where t refers to time, t_f indicates the terminal moment of the entire dynamic process (which is unknown in advance), (x,y) shows the location of the midpoint of the rear wheel axis, θ is the orientation angle, v refers to the linear velocity of point P, a refers to the corresponding acceleration, ϕ is the steering angle of the front wheels and ω refers to the corresponding angular velocity. Moreover, l, n, m and $2b$ denote the wheelbase length, front overhang length, rear overhang length and car width respectively (see Fig. 1).

In the preceding equation, $\omega(t)$ and $a(t)$ are selected as control variables, whereas the remaining five variables, i.e., $x(t), y(t), v(t), \theta(t)$ and $\phi(t)$, are regarded as state variables. Given their initial values, those state variables can be determined successfully one after another through integral provided that $\omega(t)$ and $a(t)$ are known.

2.2. Interior constraints

Apart from the vehicle kinematics described through differential equations in the preceding subsection, the bounded constraints on state/control variables should be also considered. These variables reflect the mechanical and physical constraints in a robotic system. The constraints are described in detail as follows:

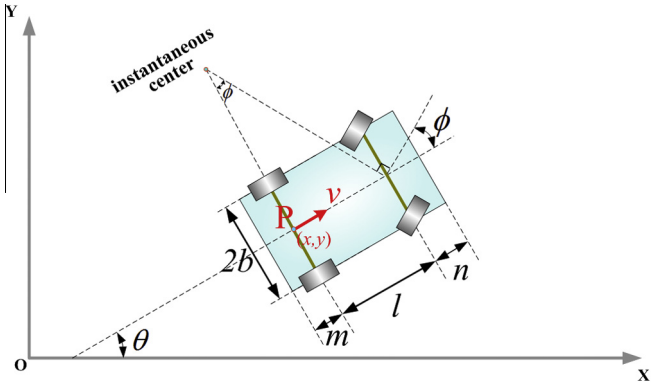

Fig. 1. Parametric notations related to robot size and kinematics.

$$\begin{cases} |a(t)| \leqslant a_{\max} \\ |v(t)| \leqslant v_{\max} \\ |\phi(t)| \leqslant \Phi_{\max} \\ |\omega(t)| \leqslant \omega_{\max} \end{cases}, \quad \text{for all } t \in [t_0, t_f]. \tag{2}$$

The reasons for imposing boundaries on $a(t)$, $v(t)$ and $\phi(t)$ are evident. To avoid the undesirable wearing of tires [40–42], many studies have planned continuous-curvature trajectories. One way to generate continuous-curvature trajectories is imposing bounds on $\omega(t)$ [35]. The rationale behind this issue as follows. Given that the instantaneous curvature $\kappa(t) = \frac{\tan\phi(t)}{l}$ and its derivative $\frac{d\kappa(t)}{dt} = \frac{\omega(t)}{l\cdot\cos^2\phi(t)}$, we expect $\kappa(t)$ to be continuous. If $\omega(t)$ is bounded, then $\frac{d\kappa(t)}{dt}$ will be bounded, then $\kappa(t)$ will be continuous, which eventually guarantees the generation of a continuous-curvature trajectory.

Mechanical and physical constraints associated with robotic kinematics have been identified. Moreover, some environmental constraints and terminal conditions must be considered when a robot moves in space, which are introduced in the next subsection.

2.3. Exterior constraints

This subsection mainly focuses on two issues: (1) how collision-free requirements are described and (2) how terminal conditions are defined.

2.3.1. Collision-free restrictions

In this work, we consider two kinds of collision-avoidance constraints. The first kind of constrain restricts a car inside a box region whereas the second kind of constraint is about avoiding obstacles in the environment.

Assume that a car-like robot is rectangular. The inside-box requirement is equivalent to that the four edge points, i.e., $A(t), B(t), C(t)$ and $D(t)$ in Fig. 2, should remain inside the box whenever $t \in [t_0, t_f]$. Here, the four edge points are given by

$$\begin{cases} A = (A_x, A_y) = (x + (l+n)\cdot\cos\theta - b\cdot\sin\theta,\ y + (l+n)\cdot\sin\theta + b\cdot\cos\theta) \\ B = (B_x, B_y) = (x + (l+n)\cdot\cos\theta + b\cdot\sin\theta,\ y + (l+n)\cdot\sin\theta - b\cdot\cos\theta) \\ C = (C_x, C_y) = (x - m\cdot\cos\theta + b\cdot\sin\theta,\ y - m\cdot\sin\theta - b\cdot\cos\theta) \\ D = (D_x, D_y) = (x - m\cdot\cos\theta - b\cdot\sin\theta,\ y - m\cdot\sin\theta + b\cdot\cos\theta) \end{cases}. \tag{3}$$

The aforementioned inside-box condition restricts a robot from colliding with the barriers. Besides that, we consider some small obstacles in the space that a car-like robot should not collide with. Assuming that the obstacles are circular, the four edge points of the robot should remain outside the obstacle region (see Fig. 3a). This necessary condition is described through Eq. (4).

$$\begin{cases} (A_x - \mathrm{CP1}_x)^2 + (A_y - \mathrm{CP1}_y)^2 \geqslant r_1^2 \\ (B_x - \mathrm{CP1}_x)^2 + (B_y - \mathrm{CP1}_y)^2 \geqslant r_1^2 \\ (C_x - \mathrm{CP1}_x)^2 + (C_y - \mathrm{CP1}_y)^2 \geqslant r_1^2 \\ (D_x - \mathrm{CP1}_x)^2 + (D_y - \mathrm{CP1}_y)^2 \geqslant r_1^2 \end{cases}, \tag{4}$$

where $(\mathrm{CP1}_x, \mathrm{CP1}_y)$ shows the location of the center of the circle and r_1 refers to the corresponding radius. However, Eq. (4) alone is insufficient to guarantee that the car will not collide with an obstacle (see the counter example in Fig. 3b). In the remainder of this part, we focus on how to judge and then to avoid the possibility shown in Fig. 3b.

To judge whether a car will hit the circular obstacle while its four edges remain outside, first, we establish a car body axis coordinate system X′GY′ (see Fig. 3b), where point $G = \left(\frac{A_x+B_x+C_x+D_x}{4}, \frac{A_y+B_y+C_y+D_y}{4}\right)$ denotes the geometric center of rectangle ABCD. Thereafter, the coordinates of the circle center in the X′GY′ frame is computed through a simple translation and rotation from its location $(\mathrm{CP2}_x, \mathrm{CP2}_y)$ in the original XOY frame, i.e.,

$$\begin{cases} \mathrm{CP2}_x^* = \left(\mathrm{CP2}_x - \frac{A_x+B_x+C_x+D_x}{4}\right)\cdot\cos\theta - \left(\mathrm{CP2}_y - \frac{A_y+B_y+C_y+D_y}{4}\right)\cdot\sin\theta \\ \mathrm{CP2}_y^* = \left(\mathrm{CP2}_x - \frac{A_x+B_x+C_x+D_x}{4}\right)\cdot\sin\theta + \left(\mathrm{CP2}_y - \frac{A_y+B_y+C_y+D_y}{4}\right)\cdot\cos\theta \end{cases}, \tag{5}$$

where $(\mathrm{CP2}_x^*, \mathrm{CP2}_y^*)$ denotes the coordinate of the center of the circle in the X′GY′ frame. Then, to avoid the possibility of the collision depicted in Fig. 3b, the following is required:

$$\begin{cases} \left|\mathrm{CP2}_x^* - \frac{m+l+n}{2}\right| \geqslant r_2, & \text{if } \left|\mathrm{CP2}_y^*\right| \leqslant b \\ \left|\mathrm{CP2}_y^* - b\right| \geqslant r_2, & \text{if } \left|\mathrm{CP2}_x^*\right| \leqslant \frac{m+l+n}{2} \end{cases}. \tag{6}$$

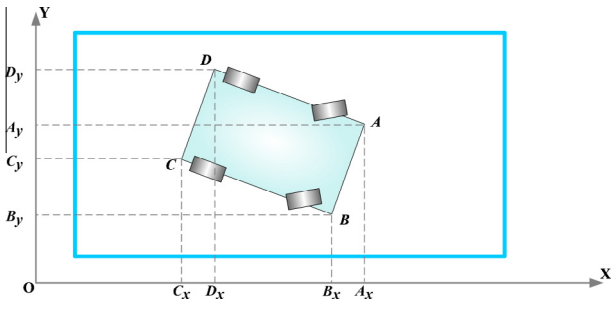

Fig. 2. Schematic of the inside-box constraint.

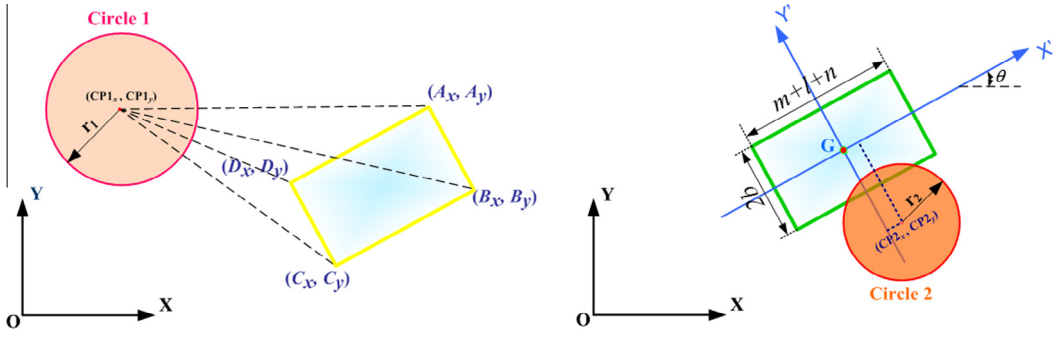

Fig. 3. Schematic of the collision-avoidance constraint: (a) a necessary condition and (b) a counter example that indicates that Eq. (4) alone is insufficient to avoid collision.

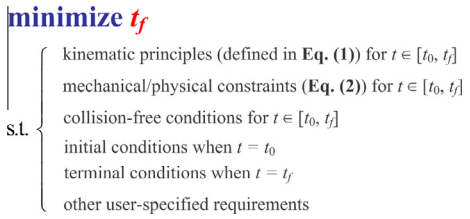

Fig. 4. Framework of the formulated dynamic optimization problem.

To briefly conclude, the necessary and sufficient condition for a car-like robot to avoid colliding into a circular obstacle is the satisfaction of Eqs. (4) and (6).

2.3.2. Terminal conditions

In addition to the kinematics and constraints, terminal conditions should be satisfied at moment $t = t_f$.

For example, a parking process should end with a full stop, that is, $v(t_f) = 0$. However, for a general trajectory planning scheme, various user-specific terminal requirements can emerge, such as requiring the car to stop in a specified box region. It should be emphasized that, although we introduce merely some simple terminal conditions in this work, our dynamic optimization model is capable of handling other complicated ones provided that they can be explicitly expressed through equalities or inequalities.

2.4. Overall dynamic optimization problem formulation

In this work, we aim to find the trajectory associated with minimum t_f. Aside from the minimization criterion (i.e., minimum driving time), other optimization criteria can also be considered whenever they are explicitly described through polynomials. Hence, our formulated dynamic optimization problem is not a specific optimization model but a unified framework that can cater to different optimization demands, conditions, or constraints. Fig. 4 schematically illustrates the unification of this framework [43].

We believe that the established optimization framework is equivalent to the original trajectory planning mission. However, this framework may be far beyond the capability of analytical methods to solve such a dynamic optimization problem uniformly. Thus, using numerical methods may be the only approach to provide the required solutions. Before introducing our concerned IPM-based simultaneous approach in Section 4, Section 3 discusses the necessary condition to judge whether an obtained trajectory is optimal.

3. Optimality evaluation of a planned trajectory

This section provides a criterion to determine whether a calculated trajectory is an optimal solution (either local or global) to the formulated dynamic optimization problem. First, we analyze the necessary conditions that should be satisfied by an optimal solution of our concerned min-time dynamic optimization problem. Then, we propose a criterion as an optimality evaluator.

3.1. Optimality conditions for an optimal control problem

The formulation illustrated in Fig. 4 can be generalized as follows:

$$\min\ t_f,\quad \text{s.t.}\ \begin{cases} \frac{dz(t)}{dt} = f(z(t),k(t),u(t)) \\ g(z(t),k(t),u(t)) \leqslant 0 \end{cases},\quad t \in [t_0, t_f], \tag{7}$$

where $z(t)$ represents the differential state variables (i.e., $x(t), y(t), v(t), \theta(t)$ and $\phi(t)$ in Section 2), $k(t)$ denotes the algebraic state variables (i.e., $A_x(t), B_x(t), C_x(t)$, etc.) and $u(t)$ represents the control variables (i.e., $\omega(t)$ and $a(t)$). The equalities and boundary conditions can be covered by $G(z(t),k(t),u(t)) \leqslant 0$ (e.g., $z(t_0) = 1$ is identical to $z(t) - 1 \leqslant 0$ and $-z(t) + 1 \leqslant 0$ at a specific moment $t = t_0$). We assume that the state equations (f and g) are smooth functions in $z(t), k(t)$ and $u(t)$ [44].

Based on Eq. (7), the final time t_f is not specified and the system is not autonomous. For the convenience of subsequent analysis, time can be normalized as $t = p_f \cdot \chi$, where $\chi \in [0,1]$ is regarded as an additional state variable and p_f denotes a scalar decision variable. Through this, Eq. (7) can be rewritten as an autonomous system with a "specified" final time as follows:

$$\min\ p_f,\quad \text{s.t.}\ \begin{cases} \frac{dz}{d\chi} = p_f \cdot f(z(\chi),k(\chi),u(\chi)) \\ g(z(\chi),k(\chi),u(\chi)) \leqslant 0 \\ \frac{d\chi}{d\chi} = 1 \\ 0 \leqslant \chi \leqslant 1 \end{cases}, \tag{8}$$

which can be further simplified as

$$\min\ p_f,\quad \text{s.t.}\ \begin{cases} \frac{dz}{d\chi} = f(z(\chi),k(\chi),u(\chi),p_f) \\ g(z(\chi),k(\chi),u(\chi)) \leqslant 0 \end{cases}. \tag{9}$$

To obtain the optimality conditions, we form the Hamiltonian

$$H(\chi) = p_f + f(z,k,u)\cdot\lambda + g(z,k,u)\cdot\gamma, \tag{10}$$

where the adjoint variables λ and γ are functions of χ. In this case, $\lambda(\chi)$ serves as the multipliers for the differential equations, whereas $\gamma(\chi)$ serves as the multipliers for all the algebraic constraints.

Next, we have

$$\frac{dH}{d\chi} = \frac{\partial H}{\partial z}\frac{dz}{d\chi} + \frac{\partial H}{\partial \lambda}\frac{d\lambda}{d\chi} + \frac{\partial H}{\partial k}\frac{dk}{d\chi} + \frac{\partial H}{\partial \gamma}\frac{d\gamma}{d\chi} + \frac{\partial H}{\partial u}\frac{du}{d\chi} + \frac{\partial H}{\partial p_f}\frac{dp_f}{d\chi}. \tag{11}$$

Pontryagin's minimum principle yields the following necessary optimality conditions [44,45]:

$$\frac{dz}{d\chi} = \frac{\partial H}{\partial \lambda}, \tag{12a}$$

$$\frac{d\lambda}{d\chi} = -\frac{\partial H}{\partial z}, \tag{12b}$$

$$\frac{\partial H}{\partial u} = 0, \tag{12c}$$

$$\frac{\partial H}{\partial k} = 0, \tag{12d}$$

$$\gamma \perp g(z,k,u) = 0. \tag{12e}$$

According to Eq. (12e), it is interesting to note that $g \cdot \frac{d\gamma}{d\chi} \equiv 0$ because if $g < 0$, then $\gamma(\chi) \equiv 0$, which yields $\frac{d\gamma}{d\chi} = 0$; on the other hand, when $g = 0$, item $g \cdot \frac{d\gamma}{d\chi}$ is directly equal to zero. Therefore,

$$\frac{\partial H}{\partial \gamma}\frac{d\gamma}{d\chi} = g \cdot \frac{d\gamma}{d\chi} \equiv 0. \tag{12f}$$

Moreover, as a scalar variable, p_f is not time-varying, thus

$$\frac{dp_f}{d\chi} = 0. \tag{12g}$$

Substituting Eqs. (12a), (12b), (12c), (12d), (12f) and (12g) into Eq. (11) yields

$$\frac{dH}{d\chi} = 0,\quad \text{for any } \chi \in [0,1], \tag{13}$$

which indicates that even for a non-autonomous system with a not specified t_f, the corresponding Hamiltonian H is constant over time $t \in [t_0, t_f]$ when optimality is achieved.

3.2. Optimality evaluator

In our concerned trajectory optimization problem, the Hamiltonian H can be written as

$$H(t) = 1 + \lambda_1(t) \cdot v(t) \cdot \cos(\theta(t)) + \lambda_2(t) \cdot v(t) \cdot \sin(\theta(t)) + \lambda_3(t) \cdot a(t) + \lambda_4(t) \cdot v(t) \cdot \tan(\phi(t)) + \lambda_5(t) \cdot \omega(t), \quad (14)$$

where $\lambda_1(t), \lambda_2(t), \lambda_3(t), \lambda_4(t)$ and $\lambda_5(t)$ denote the adjoint variables. Notably, the formulated Hamiltonian in Eq. (14) has no inequalities because of a prior assumption that the feasibility of any obtained solution must be strictly guaranteed. According to Eq. (12b), the adjoint variables are computed by

$$\begin{cases} \frac{d\lambda_1}{dt} = 0 \\ \frac{d\lambda_2}{dt} = 0 \\ \frac{d\lambda_3}{dt} = -\lambda_1(t) \cdot \cos(\theta(t)) - \lambda_2(t) \cdot \sin(\theta(t)) - \lambda_4(t) \cdot \sin(\phi(t)), \\ \frac{d\lambda_4}{dt} = \lambda_1(t) \cdot v(t) \cdot \sin(\theta(t)) - \lambda_2(t) \cdot v(t) \cdot \cos(\theta(t)) \\ \frac{d\lambda_5}{dt} = -\lambda_4(t) \cdot v(t) \cdot \frac{1}{\cos^2(\phi(t))} \end{cases} \quad (15)$$

along with the terminal condition $H(t_f) = 0$ [45].

Considering that the Hamiltonian should be constant over time when an optimal control profile is obtained, we propose a simple index in Eq. (16) to reflect the optimality of a solution. Perfectly we expect C_H to be equal to zero. The closer C_H is to 0, the closer the obtained control profile should be to being optimal.

$$C_H = \frac{\max_{\forall t \in [t_0, t_f]}(H(t)) - \min_{\forall t \in [t_0, t_f]}(H(t))}{\frac{1}{t_f - t_0} \int_{t_0}^{t_f} H(t) dt}. \quad (16)$$

4. IPM-based simultaneous approach

In this section, the IPM-based simultaneous approach is presented to solve the dynamic optimization problem formulated in Section 2.

The IPM-based simultaneous approach consists of two phases: the discretization and programming phases. In the former, all the state and control profiles in time are discretized through the collocation of finite elements. In this manner, the original dynamic optimization problem is transformed into an NLP formulation. The resulting NLP problem is then solved during the second phase. Here, the resulting NLP problem is usually in large scale (because an infinite-dimensional dynamic optimization problem has been converted into a finite-dimensional programming problem in the first phase), thus, a highly efficient NLP solver is required. IPM is an efficacious large-scale NLP solver, which is capable of solving an NLP with up to several million variables, constraints and degrees of freedom. The remainder of this section is organized as follows. First, we present the principles of the two phases respectively. Then, we focus on how this approach can be utilized to solve the dynamic optimization problem that we have formulated.

4.1. Discretization phase

Without losing generality, we consider the following general dynamic optimization problem [38]:

$$\min \ \varphi(z(t_f)), \quad \text{s.t.} \begin{cases} \frac{dz(t)}{dt} = f(z(t), \psi(t), u(t)) \\ g(z(t), \psi(t), u(t)) \leqslant 0 \end{cases}, \quad t \in [t_0, t_f], \quad (17)$$

where $z(t)$ refers to differential state variables, $\psi(t)$ denotes the algebraic state variables, $u(t)$ denotes the control variables and $\varphi(z(t_f))$ represents the minimization objective which is relevant to the terminal moment t_f. The unknown variables in Eq. (17) include $z(t), \psi(t), u(t)$ and t_f.

First, the time domain $[t_0, t_f]$ is divided into *Nfe* finite intervals $\{[t_{i-1}, t_i] | i = 1, 2, \ldots, Nfe\}$, where $t_{Nfe} = t_f$. The duration of each element can be written as $h_i = t_i - t_{i-1},\ i = 1, 2, \ldots, Nfe$. This work considers equidistant division in $[t_0, t_f]$. Thus, the duration h of each interval is equal to $\frac{t_f - t_0}{Nfe}$. For consistency with the previous studies in the field of computational science, we refer to such intervals as "finite elements".

Next, we choose $(K+1)$ interpolation points in the ith element and approximate the state using the following Lagrange polynomial:

$$z(t) = \sum_{j=0}^{K} \left(z_{ij} \cdot \prod_{k=0, \neq j}^{K} \frac{(\tau - \tau_k)}{(\tau_j - \tau_k)} \right), \quad (18)$$

where $t = t_{i-1} + h \cdot \tau,\ \tau \in [0, 1], \tau_0 = 0$ and $0 < \tau_i \leqslant 1\ (j = 1, 2, \ldots, K)$. Each τ_i refers to either Gauss or Radau points, which can be calculated off-line according to Gaussian quadrature accuracy theorem [44] when K is determined.

The Lagrange polynomial representation in Eq. (18) possesses a desirable property that $z(t_{ij}) = z_{ij}$, where $t_{ij} = t_{i-1} + h \cdot \tau_j$. That is, the polynomial function directly passes the $(K+1)$ interpolation points once they are determined on the ith finite element. This property considerably simplifies the optimization procedure in the subsequent phase.

In addition, since $z(t)$ refer to states that should be differentiated, the continuity of the differential state variables across element boundaries should be guaranteed; otherwise, the derivative at element boundaries will not exist. Thus, we have

$$z_{i+1,0} = \sum_{j=0}^{K} \left(z_{ij} \cdot \prod_{k=0, \neq j}^{K} \frac{(1 - \tau_k)}{(\tau_j - \tau_k)} \right), \quad i = 1, 2 \ldots, Nfe - 1. \quad (19)$$

Similarly, control variables $u(t)$ and algebraic states $\psi(t)$ can be represented by Lagrange interpolation profiles at K collocation points, that is,

$$u(t) = \sum_{j=1}^{K} \left(u_{ij} \cdot \prod_{k=1, \neq j}^{K} \frac{(\tau - \tau_k)}{(\tau_j - \tau_k)} \right), \quad (20)$$

and

$$\psi(t) = \sum_{j=1}^{K} \left(\psi_{ij} \cdot \prod_{k=1, \neq j}^{K} \frac{(\tau - \tau_k)}{(\tau_j - \tau_k)} \right). \quad (21)$$

Unlike in Eq. (19) for differential states $z(t)$, variables $u(t)$ and $\psi(t)$ can be discontinuous at the finite element boundaries. That means, we do not enforce $u_{i+1,0} = u_{i,K}$ or $\psi_{i+1,0} = \psi_{i,K}\ (i = 1, 2, \ldots, Nfe - 1)$.

By substituting Eqs. (19)–(21) into Eq. (17), we obtain

$$\sum_{j=0}^{K} \left(\frac{d\varsigma_j(\tau)}{d\tau}\bigg|_{\tau=\tau_k} \cdot z_{ik} \right) - h \cdot f(z_{ik}, \psi_{ik}, u_{ik}) = 0, \quad i = 1, 2 \ldots, Nfe,\ k = 1, 2, \ldots, K, \quad (22)$$

where $\varsigma_j(\tau) = \prod_{k=0,\neq j}^{K} \frac{(\tau-\tau_k)}{(\tau_j-\tau_k)}$.

Given a fixed number of elements, the NLP formulation originated from the original dynamic optimization problem, i.e., Eq. (17), can be written in the form of Eq. (23) as follows:

$$\min_{z_{ij},\psi_{ij},u_{ij}} \varphi(z(t_f)),$$
$$\text{s.t.} \begin{cases} \sum_{k=0}^{K}\left(\frac{d\varsigma_j(\tau)}{d\tau}\Big|_{\tau=\tau_k} \cdot z_{ik}\right) - h_i \cdot f(z_{ij},\psi_{ij},u_{ij}) = 0 \\ g(z_{ij},\psi_{ij},u_{ij}) \leqslant 0 \\ z_{i_1+1,0} = \sum_{j=0}^{K}\left(z_{i_1,j} \cdot \prod_{k=0,\neq j}^{K} \frac{(1-\tau_k)}{(\tau_j-\tau_k)}\right) \end{cases}, \quad i=1,2\ldots,Nfe,\ i_1=1,2\ldots,Nfe-1,\ j=1,2,\ldots,K. \tag{23}$$

Through this equation, the original dynamic optimization problem is converted into an NLP formulation during the discretization phase.

4.2. Programming phase

In this phase, IPM is adopted to optimize the discretized variables z_{ij}, ψ_{ij} and u_{ij} $(i = 1,2\ldots,Nfe, j = 1,2,\ldots,K)$ in the converted NLP formulation (i.e., Eq. (23)). The details of IPM are not discussed here for reasons of length and the focus of our paper. Interested readers may refer to [39].

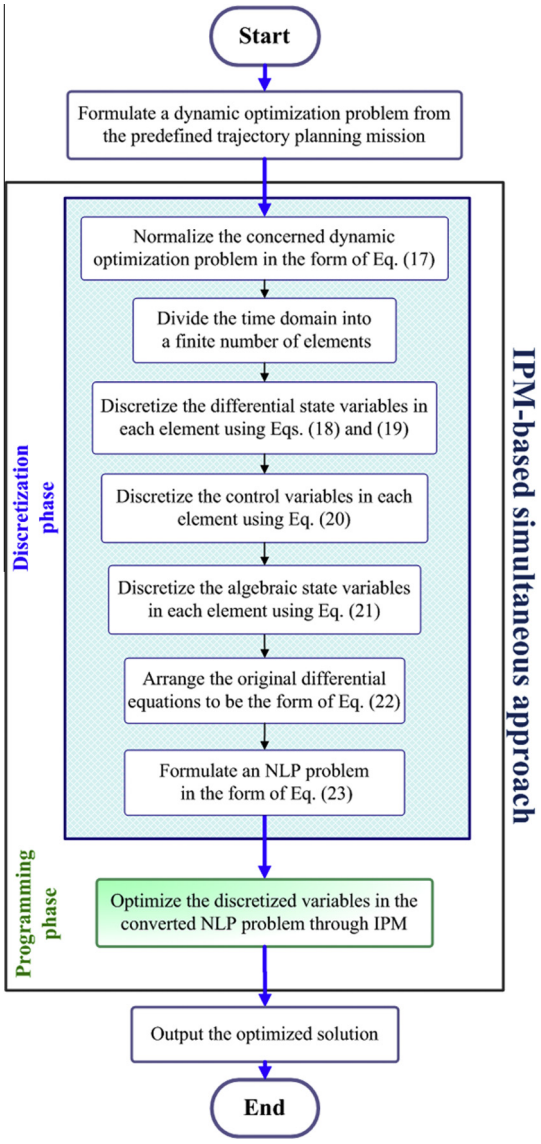

Fig. 5. Flowchart of the IPM-based simultaneous approach for dynamic optimization problems.

4.3. Overall approach for dynamic optimization

The two preceding subsections present the two phases in the IPM-based simultaneous approach. This subsection mainly focuses on how this IPM-based simultaneous approach is used to solve the dynamic optimization problem we have formulated in Section 2.

Notably, the formulated dynamic optimization problem (illustrated in Fig. 4) can be directly expressed in the form of Eq. (17). First, since we mainly pursue for time-optimal trajectories, the minimization criterion $\varphi(z(t_f))$ refers to t_f. Apart from this, other criteria can also be optimized in our formulated model easily. For example, $\varphi = \sum_{i=1}^{Nfe}\left(\sum_{j=1}^{K}\sqrt{x_{ij}^2 + y_{ij}^2}\right)$ implies that min-length solutions are pursued. Second, $z(t)$ represents all the differential state variables in Eq. (1), i.e., $x(t), y(t), v(t), \theta(t)$ and $\phi(t)$. Third, the variable $\psi(t)$ stands for all the state variables that are not differentiated, e.g., $A_x(t), B_x(t), C_x(t)$, etc. Fourth, $u(t)$ represents the control variables $\omega(t)$ and $a(t)$. Fifth, $g(z(t),\psi(t),u(t)) \leqslant 0$ covers the other equalities/inequalities and boundary conditions in Section 2.

Based on the aforementioned analyses, we manage to bridge the gap between our model and our adopted method. The overall flowchart of the IPM-based simultaneous approach is illustrated in Fig. 5 [43].

5. Simulation results and discussions

Several sets of simulations were conducted. We adopted an IPM software package called IPOPT version 3.8.0 [39]. All the simulations were executed using an Intel Core 2 Duo CPU with 2 GB RAM running at 2.53 GHz under Microsoft Windows 7. The critical parametric settings are listed in Table 1. Other parameters involved were chosen based on their default settings.

In this study, 7 cases have been designed to test the efficiency of our proposal. The scenario details are listed in Table 2. The "inside-box" requirement for Cases 3–5 means that a car-like robot should remain inside a predefined rectangular region during the entire moving process (Fig. 6). The optimization results are respectively depicted in Figs. 7–13.

Cases 1 and 2 present two scenarios in the free space. In Case 1, there is not a clearly specified terminal configuration; instead, only $y(t_f) = 2.5$ is required. Viewing Fig. 7a, we find that the concerned trajectory planning process is terminated once the car exactly reaches $y = 2.5$. In Fig. 7b, all the optimized state/control variables remain strictly inside their predefined bounds without violation, which indicate the efficiency of our optimization method. Case 2

Table 1
Parametric notations and settings.

Parameter	Description	Setting
ε_{tol}	Convergence tolerance in IPM	2.4×10^{-11}
κ_ε	Minimum absolute distance from the initial point to bound in IPM	10^{-4}
$K+1$	Interpolation point number in the simultaneous approach	4
Nfe	Number of finite elements in the simultaneous approach	–
n	Front overhang length	0.3213
l	Distance between the front and back wheel axes	1.0000
m	Rear overhang length	0.3661
$2b$	Car width	0.6243

Table 2
Details of tested trajectory planning cases.

Case no.	Initial/terminal conditions	Interior/exterior constraints ($\forall t \in [0, t_f]$)	Optimized t_f (s)	C_H index	Computation time (s)
1	$\begin{cases} x(t_0)=0 \\ y(t_0)=0 \\ v(t_0)=2 \\ \theta(t_0)=0 \\ \phi(t_0)=0 \end{cases}$ and $\begin{cases} y(t_f)=2.5 \\ v(t_f)=2 \\ \theta(t_f)=0 \\ \phi(t_f)=0 \end{cases}$	$\begin{cases} -1.5 \leqslant a(t) \leqslant 1 \\ -2 \leqslant v(t) \leqslant 2 \\ -0.585 \leqslant \phi(t) \leqslant 0.585 \\ -0.75 \leqslant \omega(t) \leqslant 0.75 \end{cases}$	3.022	9.700×10^{-4}	1.390
2	$\begin{cases} x(t_0)=1 \\ y(t_0)=1 \\ v(t_0)=0 \\ \theta(t_0)=0 \\ \phi(t_0)=0 \end{cases}$ and $\begin{cases} x(t_f)=1 \\ y(t_f)=1 \\ v(t_f)=0 \\ \theta(t_f)=\pi \\ \phi(t_f)=0 \end{cases}$	$\begin{cases} -1 \leqslant a(t) \leqslant 1 \\ -2 \leqslant v(t) \leqslant 2 \\ -1 \leqslant \phi(t) \leqslant 1 \\ -0.5 \leqslant \omega(t) \leqslant 0.5 \end{cases}$	8.471	5.082×10^{-2}	1.451
3	$\begin{cases} x(t_0)=-0.47761 \\ y(t_0)=0.4 \\ v(t_0)=2 \\ \theta(t_0)=0 \\ \phi(t_0)=0 \\ v(t_f)=0 \end{cases}$ and $\begin{cases} A_y(t_f) \leqslant 0 \\ B_y(t_f) \leqslant 0 \\ C_y(t_f) \leqslant 0 \\ D_y(t_f) \leqslant 0 \end{cases}$	$\begin{cases} -1 \leqslant a(t) \leqslant 1 \\ -2 \leqslant v(t) \leqslant 2 \\ -0.785 \leqslant \phi(t) \leqslant 0.785 \\ -0.5 \leqslant \omega(t) \leqslant 0.5 \end{cases}$ and inside-box requirement (when $BW = 2, BL = 4$)	4.951	8.253×10^{-2}	0.766
4	$\begin{cases} x(t_0)=-0.47761 \\ y(t_0)=0.4 \\ v(t_0)=2 \\ \theta(t_0)=0 \\ \phi(t_0)=0 \\ v(t_f)=0 \end{cases}$ and $\begin{cases} A_y(t_f) \leqslant 0 \\ B_y(t_f) \leqslant 0 \\ C_y(t_f) \leqslant 0 \\ D_y(t_f) \leqslant 0 \end{cases}$	$\begin{cases} -1 \leqslant a(t) \leqslant 1 \\ -2 \leqslant v(t) \leqslant 2 \\ -0.785 \leqslant \phi(t) \leqslant 0.785 \\ -0.5 \leqslant \omega(t) \leqslant 0.5 \end{cases}$ and inside-box requirement (when $BW = 2, BL = 3$)	13.527	1.418	2.823
5	$\begin{cases} x(t_0)=-0.47761 \\ y(t_0)=0.4 \\ v(t_0)=2 \\ \theta(t_0)=0 \\ \phi(t_0)=0 \\ v(t_f)=0 \end{cases}$ and $\begin{cases} A_y(t_f) \leqslant 0 \\ B_y(t_f) \leqslant 0 \\ C_y(t_f) \leqslant 0 \\ D_y(t_f) \leqslant 0 \end{cases}$	$\begin{cases} -1 \leqslant a(t) \leqslant 1 \\ -2 \leqslant v(t) \leqslant 2 \\ -0.785 \leqslant \phi(t) \leqslant 0.785 \\ -0.5 \leqslant \omega(t) \leqslant 0.5 \end{cases}$ and inside-box requirement (when $BW = 1.5, BL = 2$)	54.556	4.655	15.701
6	$\begin{cases} x(t_0)=0 \\ y(t_0)=0 \\ v(t_0)=0 \\ \theta(t_0)=0 \\ \phi(t_0)=0 \end{cases}$ and $\begin{cases} x(t_f)=4.2 \\ y(t_f)=-1 \\ v(t_f)=0 \\ \theta(t_f)=0 \end{cases}$	$\begin{cases} -1.5 \leqslant a(t) \leqslant 1 \\ -1.8 \leqslant v(t) \leqslant 1.8 \\ -0.785 \leqslant \phi(t) \leqslant 0.785 \\ -2 \leqslant \omega(t) \leqslant 2 \end{cases}$ and collision-free requirement	3.958	1.273	36.303
7	$\begin{cases} x(t_0)=1 \\ y(t_0)=0 \\ v(t_0)=0 \\ \theta(t_0)=0 \\ \phi(t_0)=0 \end{cases}$ and $\begin{cases} x(t_f)=4.2 \\ y(t_f)=-1 \\ v(t_f)=0 \\ \theta(t_f)=0 \end{cases}$	$\begin{cases} -1.5 \leqslant a(t) \leqslant 1 \\ -1.8 \leqslant v(t) \leqslant 1.8 \\ -0.785 \leqslant \phi(t) \leqslant 0.785 \\ -2 \leqslant \omega(t) \leqslant 2 \end{cases}$ and collision-free requirements	6.298	2.399	227.717

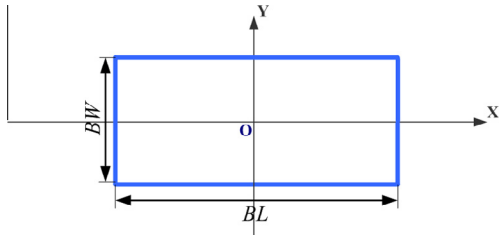

Fig. 6. Schematic of the inside-box requirement.

contains a carefully designed scenario, where the bounds imposed on the sate variables are sufficiently loose. Thus, the control profiles are more likely to be on the boundary of the control region, which causes a bang–bang control profile (Fig. 8b). Cases 3–5 are about driving in constrained rectangular regions with increasing difficulty. For example, in Case 5 the car goes back and forth repeatedly due to the tiny admissible space (Fig. 11a). The optimized state/control profiles shown in Fig. 11b can be intuitively understood with ease. Cases 6 and 7 focus on avoiding circular obstacles in the environment. In Case 6, that car passes close to but remains exactly outside the obstacle region. Case 7 is commonly regarded as a parallel parking scenario. The optimized trajectory indicates that the car needs two maneuvers (i.e., takes two halts before the terminal stop) to reach the final configuration (see Fig. 13a). Although the aforementioned simulation results are intuitively understandable and are generally in accord with common sense, yet they are computed in a fully automatic way without any human intervention.

Notably, all the collision-avoidance cases we have designed (i.e., Cases 3–7) are based on a fundamental assumption that the locations of the obstacles in the environment are fully known in advance. However, planning local trajectories (i.e., making plans dynamically according to locally sensed obstacles) seems to be relatively suitable and practical because the environment may be complex and/or time-varying [46]. At this point, we believe that the future developments in high-precision sensors and high-speed processors will make it easy to capture the time-varying environment with accuracy. In that case, the complex moving obstacles can be regarded as "static" ones at every single processing moment. When a full knowledge of the environment is available, it is better to do optimization rather than provide merely feasible solutions. In this sense, our proposal will be greatly meaningful when the relevant hardware techniques are well developed. In the remaining of this section, further discussions about the simulated results are provided.

5.1. On the selection of finite element number

In Table 2, it is notable that when the tested cases are relatively complicated, the computed solutions are unlikely to achieve optimality, because the smoothness of the corresponding

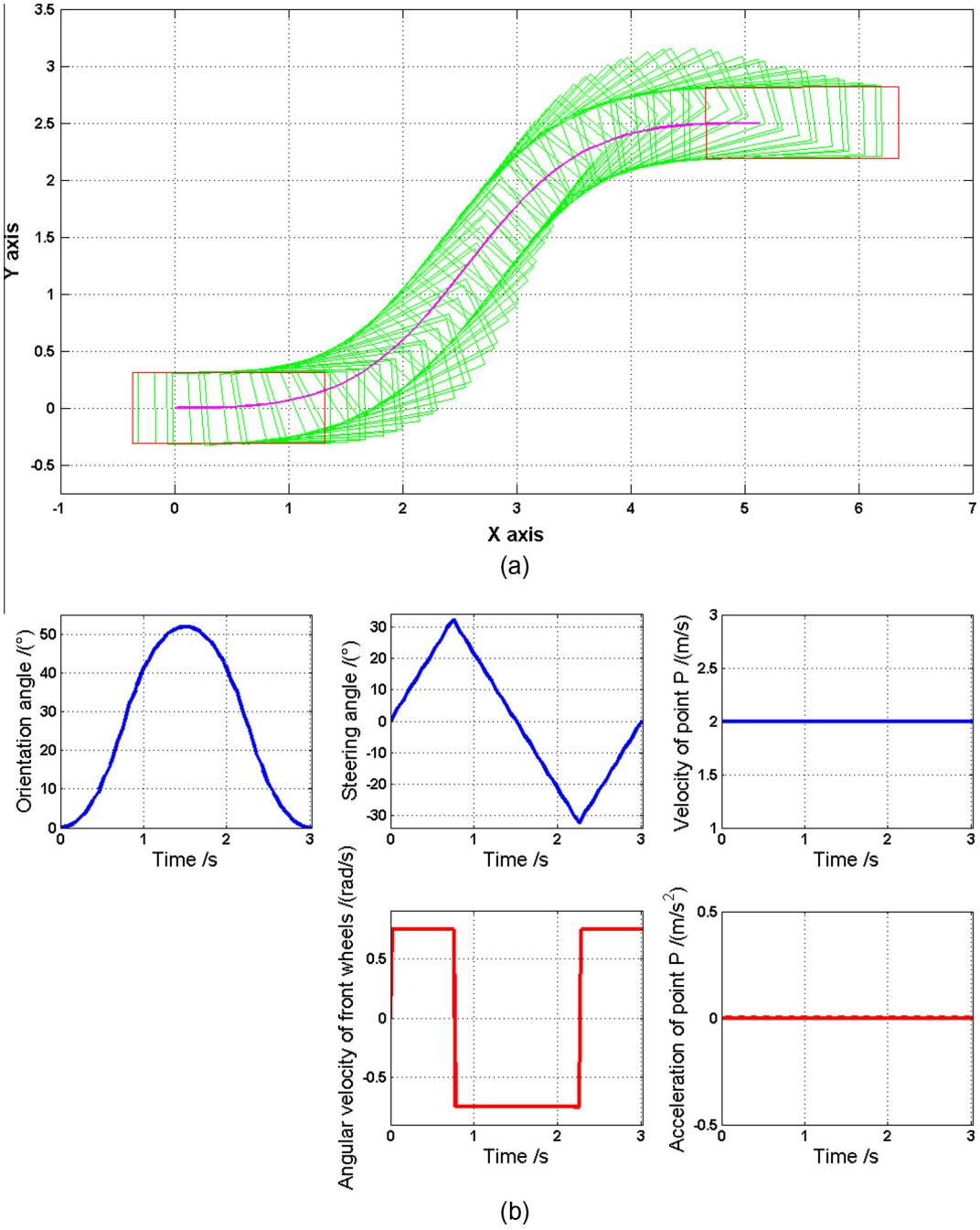

Fig. 7. Optimization results for Case 1 under the condition that $Nfe = 20$: (a) optimized path; (b) the corresponding control/state profiles. The optimized $t_f = 3.022$ s. Note that the solid curve linking the starting and terminal configurations tracks the path of point P (which locates on the midpoint of the rear wheel axis).

Hamiltonian is reduced. In this subsection, we investigate how the selection of *Nfe* can affect the optimality of a solution.

Taking Case 4 for example, we conducted simulations under the condition of different *Nfe* values. When all the optimization solutions (which are converged by the solver IPOPT) are available, we illustrate their corresponding Hamiltonian functions simultaneously in Fig. 14. Detailed results are listed in Table 3.

In Table 3, C_H gradually reduces towards zero as *Nfe* grows. This observed phenomenon indicates that the solution optimality is gradually guaranteed when *Nfe* increases. Moreover, interested readers may wonder why the optimized t_f slightly changes also. Here, it is worthwhile to notice that the finite-dimensional programming problem is similar but distinct from the formulated infinite-dimensional dynamic optimization problem. A larger *Nfe* implies that more efforts are made to portray the infinite-dimensional profiles so as to get closer to the original problem. In this sense, *Nfe* determines at what precision level is the original dynamic optimization problem solve numerically. Specially, we illustrate the optimized trajectory and the corresponding control/state profiles when $Nfe = 200$ in Fig. 15. In contrast with the profiles shown in Fig. 10b (when $Nfe = 20$), the profiles in Fig. 15b present more regular bang–bang control

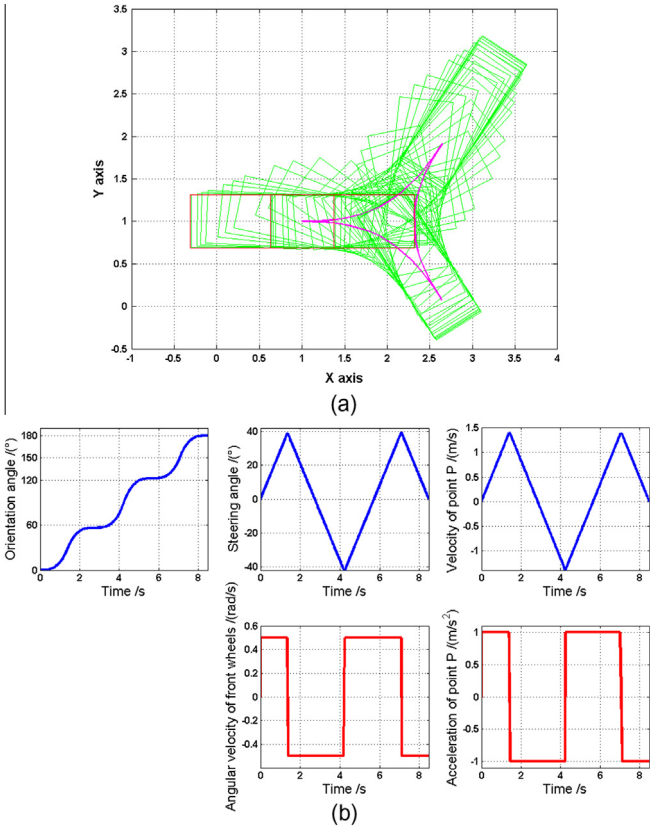

Fig. 8. Optimization results for Case 2 under the condition that $Nfe = 20$: (a) optimized path; (b) the corresponding control/state profiles. The optimized $t_f = 8.471$ s.

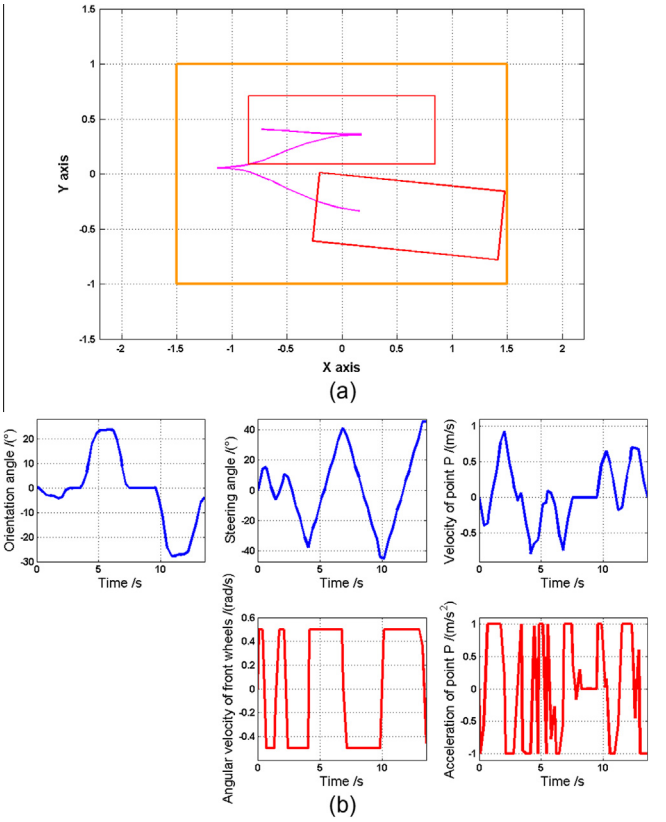

Fig. 10. Optimization results for Case 4 under the condition that $Nfe = 20$: (a) optimized path; (b) the corresponding control/state profiles. The optimized $t_f = 13.527$ s.

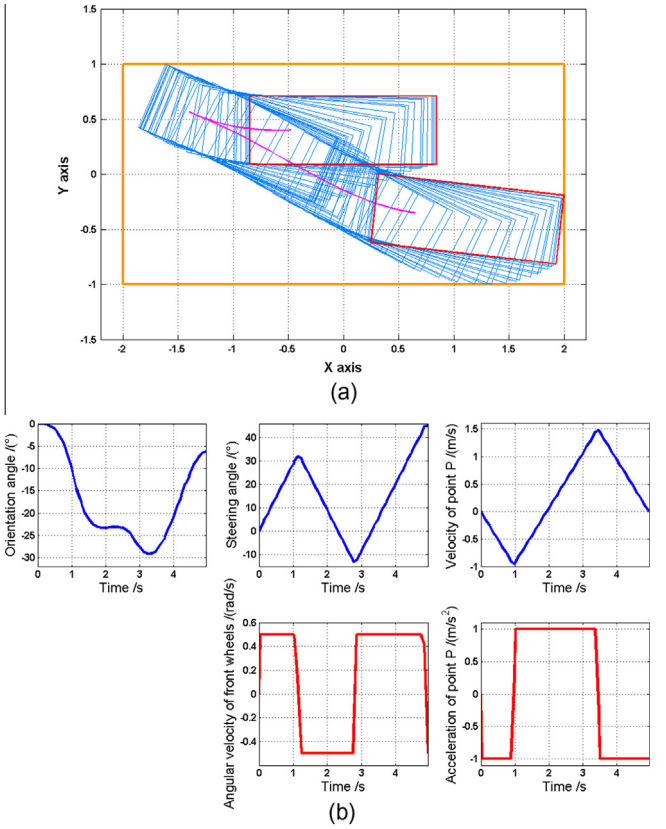

Fig. 9. Optimization results for Case 3 under the condition that $Nfe = 20$: (a) optimized path; (b) the corresponding control/state profiles. The optimized $t_f = 4.951$ s.

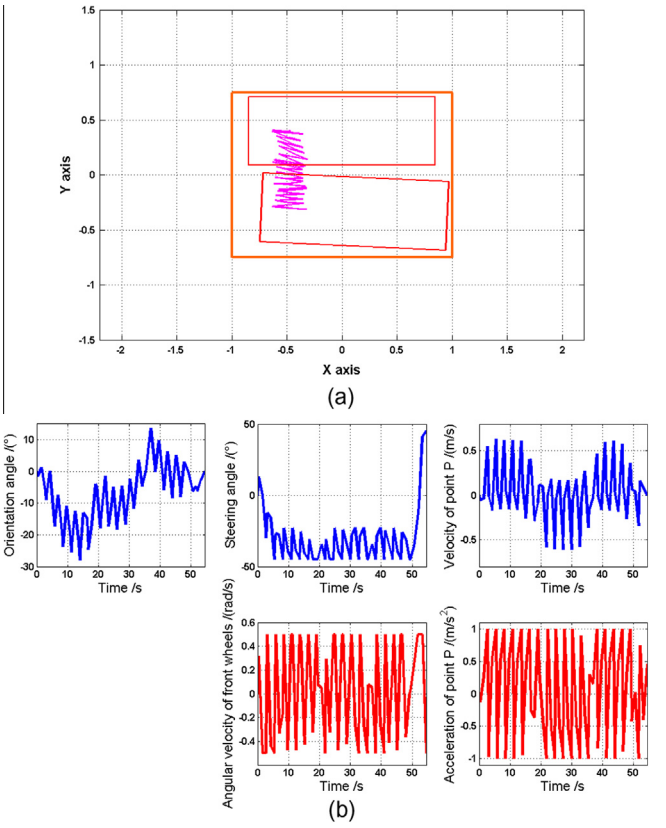

Fig. 11. Optimization results for Case 5 under the condition that $Nfe = 20$: (a) optimized path; (b) the corresponding control/state profiles. The optimized $t_f = 54.556$ s.

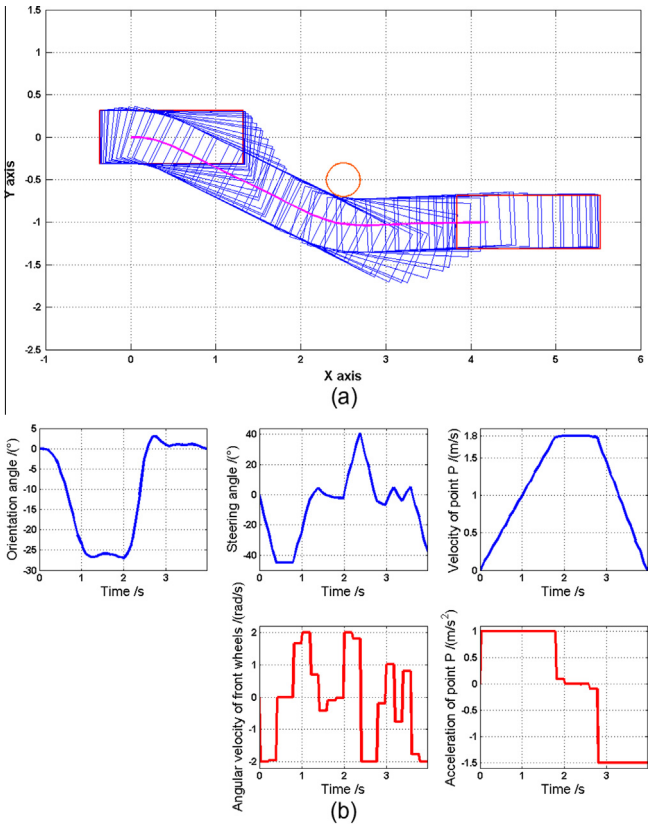

Fig. 12. Optimization results for Case 6 under the condition that $Nfe = 20$: (a) optimized path; (b) the corresponding control/state profiles. The optimized $t_f = 3.958$ s.

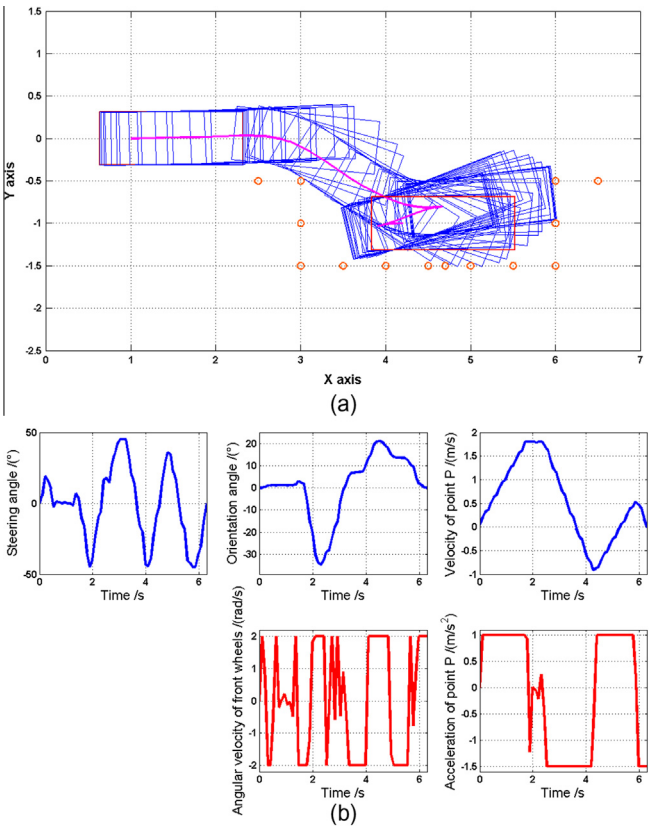

Fig. 13. Optimization results for Case 7 under the condition that $Nfe = 20$: (a) optimized path; (b) the corresponding control/state profiles. The optimized $t_f = 6.298$ s.

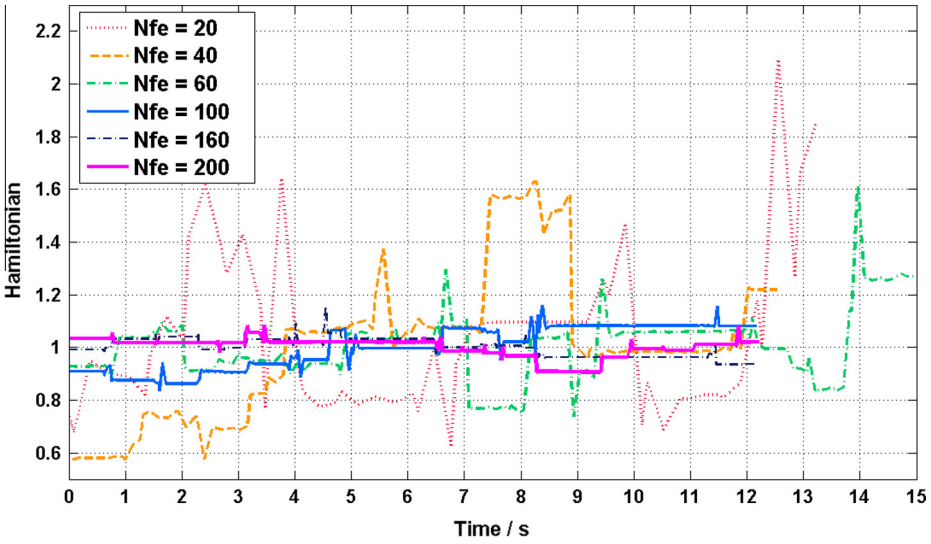

Fig. 14. Hamiltonian functions of the optimized solutions under various numbers of finite elements.

profiles. Regarding the computation efficiency issue, we notice that in general the computational time increases with the growth of *Nfe* but a clear relationship is not found. For instance, the computation time is 58.173 s when $Nfe = 60$ while it is merely 13.019 s when $Nfe = 100$. This fact implies that for the gradient-based solver IPM, to handle a larger-scale programming problem does not necessarily add to the solving difficulty. Regarding the largest-scale problem in Table 3 (i.e., the case when $Nfe = 200$), there are 11,787 variables to optimize along with 19,176 equality/inequality constraints. So complicated an optimization problem might be beyond the ability of the emerging and prevailing computational methods to provide even feasible solutions in a same amount of time [47,48].

5.2. Comparisons between min-time and min-length trajectories

In this work, the gross driving time is chosen as our minimization criterion. However, there have been also studies pursue for

Table 3
Details of the optimized solutions under different numbers of finite elements.

Nfe	Total number of variables/constraints	C_H index	Optimized t_f (s)	Computation time (s)
20	1167/1896	1.418	13.527	2.823
40	2347/3816	1.055	12.661	13.620
60	3527/5736	0.870	15.077	58.173
100	5887/9576	0.325	12.213	13.019
160	9427/15336	0.222	12.205	28.730
200	11787/19176	0.179	12.242	233.471

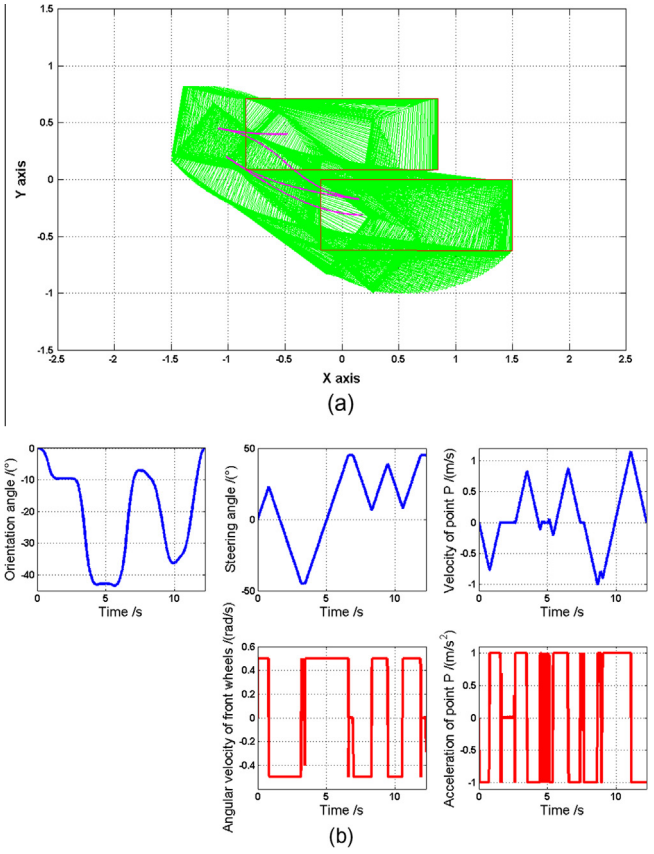

Fig. 15. Optimization results for Case 4 under the condition that $Nfe = 200$: (a) optimized path; (b) the corresponding control/state profiles. The optimized $t_f = 12.242$ s.

minimum path lengths. Since our formulation does accommodate other optimization objectives than t_f, in this subsection, min-length optimizations are conducted based on Cases 1 and 2 respectively, aiming to preliminarily investigate the difference between a min-time and a min-length solution.

With regarded to the case shown in Fig. 16, the min-length path is 0.517 meters shorter than the min-time path. Interestingly, to execute the min-length trajectory, a car-like robot should first slow down (see Fig. 16b). This phenomenon is understandable: since a larger $|\phi|$ enables a car turn around more easily, and that the formulated optimization mission no longer cares about the driving time, thus, it is willing to "wastes" time to wait for the steering wheels reach the left bound. A similar simulation result is obtained based on Case 2 (see Fig. 17). During its entire moving process, the car moves slowly (at a speed of 10^{-3} m/s or so) while the steering angle changes relatively fast. This makes the whole min-length path nearly a combination of three circular arcs. At this point the simulated result shown in Fig. 17a is in accord with Reeds &

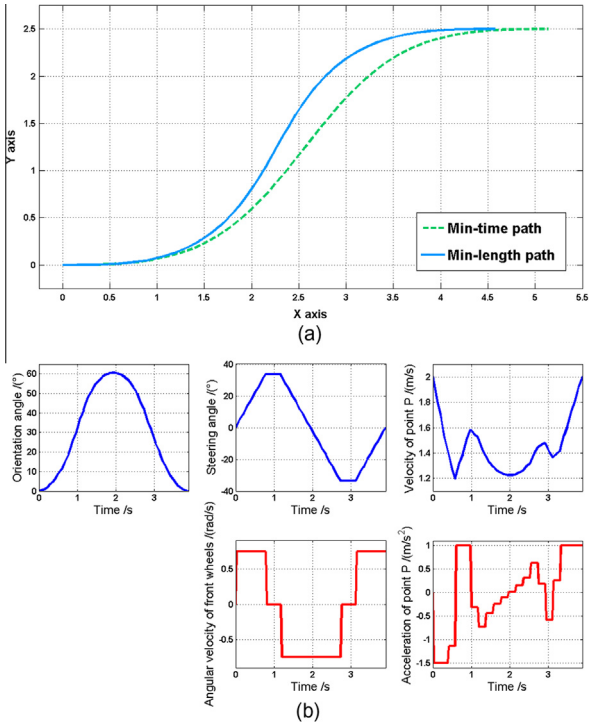

Fig. 16. Comparative min-length optimization based on Case 1: (a) optimized min-time and min-length paths; (b) the corresponding control/state profiles of the min-length solution. The optimized min-time trajectory takes 3.022 s to accomplish whereas the min-length trajectory takes 3.900 s. On the other hand, the min-time path is 6.043 m long whereas the min-length path is 5.526 m long.

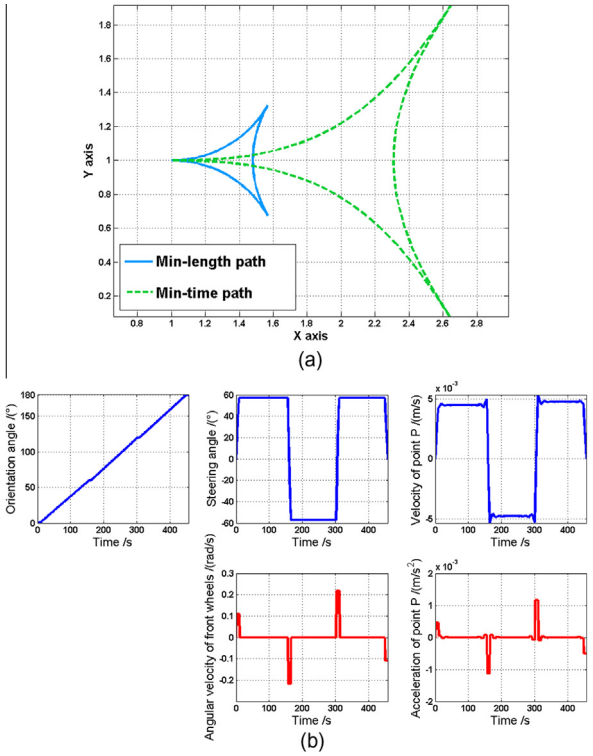

Fig. 17. Comparative min-length optimization based on Case 2: (a) optimized min-time and min-length paths; (b) the corresponding control/state profiles of the min-length solution. The optimized min-time trajectory takes 8.471 s to accomplish whereas the min-length trajectory takes 456.801 s. On the other hand, the min-time path is 5.978 m long whereas the min-length path is 2.052 m long.

Shepp's analysis [21]. If a min-length solution takes far longer time to implement, it may be better to choose a min-time solution to follow (in our concerned case, we do not mind going that 3.926 meters farther).

6. Conclusion

We have investigated the process of planning time-optimal motions for car-like robots. The underlying contributions of this study lie in the following aspects.

First, unlike some prevailing methods/models that involve mission transformation, our proposed method aims to deal with the original problem (i.e., the trajectory planning mission) directly. That is why we adopt differential equations to describe how a car-like robot moves and utilize algebraic inequalities to strictly describe the collision-avoidance conditions and mechanical constraints. Through this, we have actually established an open dynamic optimization framework that can deal with a wide range of optimization criteria and user-specified requirements. The high efficiency of the IPM-based simultaneous approach supports such a unified formulation.

Second, we utilize a Hamiltonian-based necessary condition in the optimal control theory to determine the quality of the planned trajectories. Compared to most of the trajectory planners which obtain feasible/optimized solutions only, our proposed criterion C_H is capable of telling how close one solution is to being optimal.

Third, we have investigated (1) the impact of *Nfe* selection on the optimization results and (2) the differences between min-length and min-time solutions.

Acknowledgements

The authors would like to thank the anonymous referees for their invaluable comments and suggestions. This work is supported by the 973 Program of China under Grant No. 2009CB320603, the National Nature Science Foundation under Grant No. 61374167, and the 6th National College Students' Innovation and Entrepreneurial Training Program under Grant No. 201210006050. Many thanks also go to Ms. R. Zhou and Dr. K. Wang for their kind help.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.advengsoft.2015.04.011.

References

[1] Gasparetto A, Zanotto V. Optimal trajectory planning for industrial robots. Adv Eng Softw 2010;41(4):548–56.
[2] Huang W. Optimal multi-impulse orbit transfer using nonlinear relative motion dynamics. J Astronaut Sci 2014;59(1):237–58.
[3] Pharpatara P, Pepy R, Hérissé B, Bestaoui Y. Missile trajectory shaping using sampling-based path planning. In: Proceedings of the 2013 IEEE/RSJ international conference on intelligent robots and systems; 2013. p. 2533–8.
[4] Li B, Gong L, Yang W. An improved artificial bee colony algorithm based on balance-evolution strategy for unmanned combat aerial vehicle path planning. Sci World J 2014;2014(232704):1–10.
[5] Alejo D, Cobano A, Heredia G, Ollero A. Collision-free 4D trajectory planning in unmanned aerial vehicles for assembly and structure construction. J Intell Rob Syst 2014;73(1):783–95.
[6] Li B, Chiong R, Lin M. A two-layer optimization framework for UAV path planning with interval uncertainties. In: 2014 IEEE symposium on computational intelligence in production and logistics systems (CIPLS); 2014. p. 120–7.
[7] Chung T, Hollinger G, Isler V. Search and pursuit evasion in mobile robotics. Auton Rob 2011;31(4):299–316.
[8] Li B, Chiong R, Gong L. Search-evasion path planning for submarines using the artificial bee colony algorithm. IEEE Congr Evol Comput 2014:528–35.
[9] Soldan S, Welle J, Barz T, Kroll A, Schulz D. Towards autonomous robotic systems for remote gas leak detection and localization in industrial environments. In: Yoshida K, Tadokoro S, editors. Field and service robotics. Berlin, Heidelberg: Springer; 2014. p. 233–47.
[10] Moon J, Bae I, Cha J, Kim S. A trajectory planning method based on forward path generation and backward tracking algorithm for automatic parking systems. In: Proceedings of the IEEE 17th international conference on intelligent transportation systems; 2014.p. 719–24.
[11] Li B, Shao Z. Time-optimal maneuver planning in automatic parallel parking using simultaneous dynamic optimization approach, unpublished.
[12] Martinez-Marin T. Learning optimal motion planning for car-like vehicles. Proceedings of the international conference on computational intelligence for modeling 2005:601–12.
[13] Soueres P, Laumond J. Shortest paths synthesis for a car-like robot. IEEE Trans Autom Control 1996;41(5):672–88.
[14] Galceran E, Carreras M. A survey on coverage path planning for robotics. Rob Auton Syst 2013;61(12):1258–76.
[15] Andreasson H, Bouguerra A, Cirillo M, Dimitrov D, Driankov D, Karlsson L, et al. Autonomous transport vehicles: where we are and what is missing. IEEE Rob Automat Mag 2015;22(1):64–75.
[16] Czubenko M, Kowalczuk Z, Ordys A. Autonomous driver based on an intelligent system of decision making. Cogn Comput 2015. http://dx.doi.org/10.1007/s12559-015-9320-5.
[17] Barraquand J, Latombe J. On nonholonomic mobile robots and optimal maneuvering. In: Proceedings of the IEEE international symposium on intelligent control; 1989. p. 340–7.
[18] David J, Manivannan P. Control of truck-trailer mobile robots: a survey. Intel Serv Robot 2012;7(4):245–58.
[19] Soueres P, Boissonnat J. Optimal trajectories for nonholonomic mobile robots. In: Lammond J, editor. Robot motion planning and control. Berlin, Heidelberg: Springer; 1998. p. 93–170.
[20] Dubins L. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. Am J Math 1957;x(x):497–516.
[21] Reeds J, Shepp L. Optimal paths for a car that goes both forward and backwards. Pac J Math 1990;145(2):367–93.
[22] Wang X, Wang J, Rao Z. An adaptive parametric interpolator for trajectory planning. Adv Eng Softw 2010;41(2):180–7.
[23] Elbanhawi M, Simic M, Jazar R. Continuous path smoothing for car-like robots using B-spline curves. J Intell Rob Syst 2015;x(x):1–34.
[24] Gorinevsky D, Kapitanovsky A, Goldenberg A. Neural network architecture for trajectory generation and control of automated car parking. IEEE Trans Control Syst Technol 1996;4(1):50–6.
[25] Martinez-Marin T. Learning optimal motion planning for car-like vehicles. In: IEEE international conference on computational intelligence for modeling, control, and automation and international conference on intelligent agents, web technologies, and internet commerce; 2005. p. 601–12.
[26] Zhao Y, Collins E. Robust automatic parallel parking in tight spaces via fuzzy logic. Rob Auton Syst 2005;51(2):111–27.
[27] Khoukhi A. Data-driven multi-stage multi-objective motion planning of mobile robots, application to near minimum power fuzzy parking. Comput Electr Eng 2015. http://dx.doi.org/10.1016/j.compeleceng.2014.12.008.
[28] Kavraki L, Svestka P, Latombe J, Overmars M. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Rob Automat 1996;12(4):566–80.
[29] Karaman S, Walter M, Perez A, Frazzoli E, Teller S. Anytime motion planning using the RRT*. In: IEEE international conference on robotics and automation (ICRA) 2011. p. 1478–83.
[30] Qureshi A, Ayaz Y. Intelligent bidirectional rapidly exploring random trees for optimal motion planning in complex cluttered environments. Rob Auton Syst 2015;68:1–11.
[31] Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning. Int J Robot Res 2011;30(7):846–94.
[32] Liang J, Lee C. Efficient collision-free path planning of multiple mobile robots system using efficient artificial bee colony algorithm. Adv Eng Softw 2015;79:47–56.
[33] Hassan S, Yoon J. Haptic assisted aircraft optimal assembly path planning scheme based on swarming and artificial potential field approach. Adv Eng Softw 2014;69:18–25.
[34] Ma Y, Wang H, Xie Y, Guo M. Path planning for multiple mobile robots under double-warehouse. Inf Sci 2014;278:357–79.
[35] Kondak K, Hommel G. Computation of time optimal movements for autonomous parking of non-holonomic mobile platforms. In: Robotics and Automation, 2001. Proceedings of the IEEE International Conference on ICRA, vol. 3; 2001. p. 2698–703.
[36] Kim Y, Kim B. Efficient time-optimal two-corner trajectory planning algorithm for differential-driven wheeled mobile robots with bounded motor control inputs. Rob Auton Syst 2015;64:35–43.
[37] Balkcom D, Mason M. Time optimal trajectories for bounded velocity differential drive vehicles. Int J Robot Res 2002;21(3):199–217.

[38] Biegler L. An overview of simultaneous strategies for dynamic optimization. Chem Eng Process: Process Intensification 2007;46(11):1043–53.
[39] Wächter A, Biegler L. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math Program 2006;106(1):25–57.
[40] Vorobieva H, Minoiu-Enache N, Glaser S, Mammar S. Geometric continuous-curvature path planning for automatic parallel parking. In: Proceedings of the 10th IEEE international conference on networking, sensing, and control; 2013. p. 418–23.
[41] Vorobieva H, Glaser S, Minoiu-Enache N, Mammar S. Automatic parallel parking in tiny spots:path planning and control. IEEE Trans Intell Transp Syst 2015;16(1):396–410.
[42] Vorobieva H, Glaser S, Minoiu-Enache N, Mammar S. Automatic parallel parking with geometric continuous-curvature path planning. In: IEEE intelligent vehicles symposium proceedings; 2014. p. 465–71.
[43] Li B, Shao Z. A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles. Knowl Based Syst 2015. http://dx.doi.org/10.1016/j.knosys.2015.04.016.
[44] Biegler LT. Nonlinear programming: concepts, algorithms, and applications to chemical processes. SIAM; 2010. vol. 10.
[45] Kirk D. Optimal control theory: an introduction. Courier Corporation; 2012. p. 227–32.
[46] Abiyev R, Ibrahim D, Erin B. Navigation of mobile robots in the presence of obstacles. Adv Eng Softw 2010;41(10):1179–86.
[47] Salimi H. Stochastic fractal search: a powerful metaheuristic algorithm. Knowl-Based Syst 2015;75:1–18.
[48] Gonçalves S, Lopez H, Miguel F. Search group algorithm: a new metaheuristic method for the optimization of truss structures. Comput Struct 2015;153:165–84.