Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Fast Trajectory Planning in Cartesian rather than Frenet Frame: A Precise Solution for Autonomous Driving in Complex Urban Scenarios

Bai Li*, Youmin Zhang**

** College of Mechanical and Vehicle Engineering, Hunan University, Changsha, China (e-mail: libai@hnu.edu.cn, libaioutstanding@163.com).*
*** Department of Mechanical, Industrial and Aerospace Engineering, Concordia University, Montreal, Canada (e-mail: ymzhang@encs.concordia.ca).*

Abstract: On-road trajectory planning is a direct reflection of an autonomous vehicle's intelligence level when traveling on an urban road. The prevalent on-road trajectory planners include the spline-based, sample-and-search-based, and optimal-control-based methods. Path-velocity decomposition and Frenet frame have been widely adopted in the aforementioned methods, which, nonetheless, largely degrade the trajectory planning quality when the road curvature is large and/or the scenario is complex. This paper aims to plan precise and high-quality on-road trajectories, thus we choose to describe the concerned scheme as an optimal control problem, wherein the urban road scenario is described completely in the Cartesian frame rather than in the Frenet frame. The formulated optimal control problem should be numerically solved in real-time. To that end, a light-weighted iterative computation architecture is built. In each iteration, a tunnel construction strategy tractablely models the collision-avoidance constraints, and a constraint softening strategy helps to find an intermediate trajectory for constructing the tunnels in the next iteration. Efficacy of the proposed on-road trajectory planner is validated by simulations on a high-curvature urban road wherein the ego vehicle is surrounded by multiple social vehicles at various speeds.

Keywords: Autonomous driving, trajectory planning, computational optimal control, nonlinear program (NLP), Frenet frame

1. INTRODUCTION

Autonomous driving on a complex urban road requires the cooperation of multiple modules, including localization, perception, prediction, planning, and control (Levinson et al., 2011). Planning techniques are important as they are a direct reflection of the intelligence level of an autonomous vehicle (Li et al., 2019a). The existing planning methods are typically developed for parking in a tiny parking lot (Li and Shao, 2015) and cruising on a structured urban road (Claussmann et al., 2020). Commonly a cruising-oriented planner cannot deal with parking cases while a parking-oriented planner may not bring out the best in a cursing task either. This paper is focused on the local trajectory planning on an urban road.

1.1 Related Studies

The prevailing on-road trajectory planners, in the early years, are the spline-based methods, which use clothoid curves or polynomials to present a path/trajectory. If a spline-based method only plans a path, then extra efforts are needed to attach a time course along the derived path to form a trajectory. Commonly a spline-based method evenly samples some offset candidates ahead to produce spline candidates, then evaluates their performances w.r.t. underlying collision risks, vehicle kinematics violations, passenger comfort, and travel efficiency (Hu et al., 2018). Since a spline-based method usually samples multiple splines for selection, it is also referred to as a lattice planner in the literature. However, a spline-based planner is inefficacious when the on-road situation becomes complex, because there may not be even one feasible solution among the multiple sampled splines. As a remedy, sample-and-search-based planners are proposed, which discretize the spatial or spatial-temporal space into a multi-layer grid map, and derive the best one via graph searches (McNaughton et al., 2011; Ajanovic et al., 2018). Sample-and-search-based planners usually ensure resolution completeness or even resolution optimality, but the derived coarse path/trajectory is not smooth, which would render difficulties in the subsequent control module. A virtual controller is deployed to simulate the tracking process of the coarse path/trajectory, and the tracking result is forwarded to the control module (Ma et al., 2015). Alternatively, one may adopt a computational optimal control approach, which is warm-started by the coarse path/trajectory (Meng et al., 2019; Lim et al., 2018; Ziegler et al., 2014; Ding et al., 2019; Chen et al., 2019). Compared with the virtual-controller-based methods, the optimal-control-based methods describe the trajectory smoothing (also known as trajectory optimization) scheme as a continuous-time optimization, which means there are degrees of freedom to construct a smooth trajectory. Assuming that a coarse trajectory is already derived by a sample-and-search-based planner, this work particularly focuses on the optimal-control-based trajectory optimization.

The predominant optimal-control-based trajectory optimizers are typically subject to two limitations: i) they describe the driving process in the Frenet frame rather than the Cartesian

frame, and ii) they decouple trajectory planning into path + velocity planning or lateral + longitudinal planning. More explicitly, the usage of Frenet frame uniformly eases the modeling difficulties of a road's trend (Werling et al., 2010), but the mapping between Frenet and Cartesian frames is theoretically not stable (Wang et al., 2002), which leads to impractical results when the road curvature is large (Fig. 1). Even when the road curvature is small, the real-world curvature is seldom reflected in a Frenet-frame-based trajectory planning method (Barfoot and Clark, 2004), yielding that the derived trajectory might not be able to be efficaciously tracked by the controller. The second limitation involves dividing a problem into multiple highly dependent parts but conquering them independently. Such a decoupling operation harms the algorithm optimality or even completeness when the scenario is complex (Miller et al., 2018).

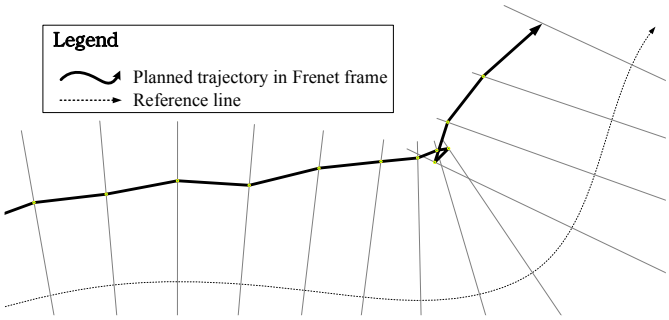

Fig. 1. An unrealistic trajectory presented in the Frenet frame when road curvature is large.

1.2 Contributions

This work aims to provide accurate trajectory planning solutions to generic on-road autonomous driving cases. To that end, we describe the on-road trajectory planning scheme as an optimal control problem, the solution to which is a trajectory rather than a path or other partial elements of a trajectory. Also, the on-road driving scenario is described in the Cartesian rather than Frenet frame, thereby making the derived trajectories easy to track. Nonetheless, formulating an on-road trajectory planning scheme as a standard optimal control problem in the Cartesian frame and solving it numerically like off-road parking maneuver planning would take long processing time, which is not acceptable in an on-road driving task.

This paper proposes a fast solution to the aforementioned optimal control problem. Concretely, a tunnel-based strategy is adopted to model the collision-avoidance constraints in a tractable way, and a constraint softening strategy is proposed to facilitate the numerical optimization process. A light-weighted iterative computation framework is built, wherein the tunnel construction and numerical optimization are alternatively executed. Since the two steps in each iteration are simple, the iteration process usually converges quickly.

1.3 Organization

In the rest of this paper, Section 2 formulates the nominal optimal control problem for our concerned on-road trajectory planning scheme. Section 3 introduces our proposed method, which aims to provide *accurate*, *fast*, and *optimal* solutions. Simulation setups, results, and discussions are provided in Section 4, and finally, Section 5 concludes the paper.

2. NOMINAL PROBLEM FORMULATION

In this section, the on-road trajectory planning scheme is nominally formulated as an optimal control problem, which consists of a cost function and many constraints:

$$\begin{aligned}
&\text{minimize } \Gamma\left(\mathbf{x}(t),\ \mathbf{u}(t)\right)\\
&\text{subject to}\\
&f\left(\mathbf{x}(t),\ \mathbf{u}(t)\right)=0,\\
&h_{\text{box}}\left(\mathbf{x}(t)\right)\le 0,\ h_{\text{box}}\left(\mathbf{u}(t)\right)\le 0,\\
&h_{\text{collision}}\left(\mathbf{x}(t)\right)\le 0,\\
&\mathbf{x}(0)=\mathbf{x}_{\text{init}},\ \mathbf{u}(0)=\mathbf{u}_{\text{init}},\\
&g\left(\mathbf{x}(\mathrm{t_f}),\ \mathbf{u}(\mathrm{t_f})\right)\le 0.
\end{aligned} \tag{1}$$

Herein, $t\in[0,\mathrm{t_f}]$ denotes the time index, and the terminal moment $\mathrm{t_f}$ is fixed. $\mathbf{x}(t)=\left[x(t),y(t),\theta(t),v(t),\phi(t)\right]$ contains the state profiles, $\mathbf{u}(t)=\left[a(t),\omega(t)\right]$ contains the control profiles, and the concrete definitions follow (Li and Shao, 2015). $f\left(\mathbf{x}(t),\ \mathbf{u}(t)\right)=0$ represents the vehicle kinematic principles described by differential equations. $h_{\text{box}}\left(\bullet\right)\le 0$ represents the inequality box constraints, and $h_{\text{collision}}\left(\bullet\right)\le 0$ denotes the algebraic inequality collision-avoidance constraints. $g\left(\mathbf{x}(\mathrm{t_f}),\ \mathbf{u}(\mathrm{t_f})\right)\le 0$ stands for the implicit boundary conditions at $t=\mathrm{t_f}$. The cost function Γ is defined as

$$\Gamma=\int_{\tau=0}^{\mathrm{t_f}}\left(\left\|\mathbf{w}_1\cdot(\mathbf{x}-\mathbf{x}_{\text{ref}})\right\|^2+\left\|\mathbf{w}_2\cdot(\mathbf{u}-\mathbf{u}_{\text{ref}})\right\|^2\right)\cdot\mathrm{d}\,\tau, \tag{2}$$

wherein $\mathbf{w}_1,\mathbf{w}_2$ are weighting vectors, $\mathbf{x}_{\text{ref}}$ and $\mathbf{u}_{\text{ref}}$ denote the nominal driving states and operations, respectively. Eq. (2) reflects the smoothness of the state/control profiles.

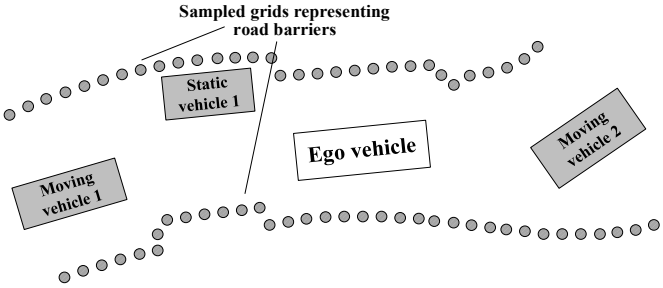

Fig. 2. A typical on-road driving scenario.

The primary difficulties in (1) lie in the collision-avoidance constraints described in the Cartesian frame. As depicted in Fig. 2, the ego vehicle needs to avoid collisions with the static/moving obstacles as well as the road barriers. Since the road barriers may be in irregular shapes, they need to be sampled as grids and regarded as static obstacles in $h_{\text{collision}}\le 0$, which makes (1) intractable.

3. FAST NUMERICAL SOLUTION

This section introduces a fast solution method that solves (1) numerically. A fundamental method is presented first, which is further improved by developing an iterative computation framework.

3.1 Basic Method

Recall that the primary difficulties in (1) are caused by the intractable collision-avoidance constraints. Given that the ego vehicle does not have chances to collide with all the static/moving obstacles at every moment, a tunnel-based strategy is adopted to convert the complicated collision-avoidance constraints into a fixed scale of within-tunnel constraints.

A spatial-temporal tunnel is constructed that separates the ego vehicle from the obstacles in the scenario. In building a tunnel, the first step is to plan a coarse trajectory, which determines the homotopic route on the road. The coarse trajectory can be derived via a sample-and-search-based planner such as (Xu et al., 2014). In the second step, we use two discs to even cover the rectangular body of the ego vehicle, and denote the disc centers as $P_{\mathrm{f}}=(x_{\mathrm{f}},y_{\mathrm{f}})$ and $P_{\mathrm{r}}=(x_{\mathrm{r}},y_{\mathrm{r}})$ (Li et al., 2019b):

$$\begin{aligned} x_{\mathrm{f}}(t)&=x(t)+\frac{1}{4}(3\mathrm{L_W}+3\mathrm{L_F}-\mathrm{L_R})\cdot\cos\theta(t),\\ y_{\mathrm{f}}(t)&=y(t)+\frac{1}{4}(3\mathrm{L_W}+3\mathrm{L_F}-\mathrm{L_R})\cdot\sin\theta(t),\\ x_{\mathrm{r}}(t)&=x(t)+\frac{1}{4}(\mathrm{L_W}+\mathrm{L_F}-3\mathrm{L_R})\cdot\cos\theta(t),\\ y_{\mathrm{r}}(t)&=y(t)+\frac{1}{4}(\mathrm{L_W}+\mathrm{L_F}-3\mathrm{L_R})\cdot\sin\theta(t). \end{aligned} \tag{3a}$$

The radius $\mathrm{R_C}$ of either disc is determined according to

$$\mathrm{R_C}=\frac{1}{2}\sqrt{\left(\frac{\mathrm{L_R}+\mathrm{L_W}+\mathrm{L_F}}{2}\right)^2+(\mathrm{L_B})^2}, \tag{3b}$$

where $\mathrm{L_W}$, $\mathrm{L_F}$, $\mathrm{L_R}$, and $\mathrm{L_B}$ represent the vehicle wheelbase, front overhang length, rear overhang length, and width, respectively. Suppose the derived coarse trajectory records the movement of rear-axle center point $(x(t),y(t))$, in the third step, we need to specify the movements of points P_{f} and P_{r} via (3). For convenience, the trajectory for P_{f} is denoted as $Traj_{\mathrm{f}}$ and the trajectory for P_{r} as $Traj_{\mathrm{r}}$. Thereafter, two tunnels are paved along $Traj_{\mathrm{f}}$ and $Traj_{\mathrm{r}}$, respectively. Suppose that the tunnel for P_{f} consists of $\mathrm{N_{box}}$ local boxes. In specifying the kth box for P_{f}, we capture the space occupied by the static/moving obstacles at $t_k=\mathrm{t_f}/\mathrm{N_{box}}\cdot k$ to form an instantaneous map, and then spirally expand from the point $(Traj_{\mathrm{f}}.x(t_k),Traj_{\mathrm{f}}.y(t_k))$ until collisions with the instantaneous map occur (Fig. 3). Applying such procedures to all t_k $(k=1,...,\mathrm{N_{box}})$ renders the tunnel for P_{f}. The tunnel for P_{r} can be derived similarly.

Finally, the original collision-avoidance constraints are converted to the within-tunnel constraints, which require that either disc center stays in its tunnel at sampled instances, e.g.,

$$\begin{aligned} &\mathbf{lb}_k\le\mathbf{dc}_k\le\mathbf{ub}_k,\\ &\mathbf{dc}_k=\left[x_{\mathrm{f}}(t_k),y_{\mathrm{f}}(t_k),x_{\mathrm{r}}(t_k),y_{\mathrm{r}}(t_k)\right],\\ &t_k=\frac{\mathrm{t_f}}{\mathrm{N_{box}}}\cdot k,\ k=1,...,\mathrm{N_{box}}. \end{aligned} \tag{4}$$

By replacing $h_{\mathrm{collision}}(\mathbf{x}(t))\le 0$ with (3) and (4) in (1), a tunnel-based optimal control problem (denoted as $\mathrm{P_0}$) is formulated:

$$\begin{aligned} &\text{minimize } \Gamma(\mathbf{x}(t),\ \mathbf{u}(t))\\ &\text{subject to}\\ &f(\mathbf{x}(t),\ \mathbf{u}(t))=0,\\ &h_{\mathrm{box}}(\mathbf{x}(t))\le 0,\ h_{\mathrm{box}}(\mathbf{u}(t))\le 0,\\ &\text{Eqs. (3), (4),}\\ &\mathbf{x}(0)=\mathbf{x}_{\mathrm{init}},\ \mathbf{u}(0)=\mathbf{u}_{\mathrm{init}},\\ &g(\mathbf{x}(\mathrm{t_f}),\ \mathbf{u}(\mathrm{t_f}))\le 0. \end{aligned} \tag{5}$$

In solving $\mathrm{P_0}$ numerically, $\mathrm{P_0}$ is first discretized into a nonlinear program (NLP) problem and then optimized by a gradient-based NLP-solver such as the interior-point method (IPM). The technical details of the tunnel-based method are elaborated in (Li et al., 2020).

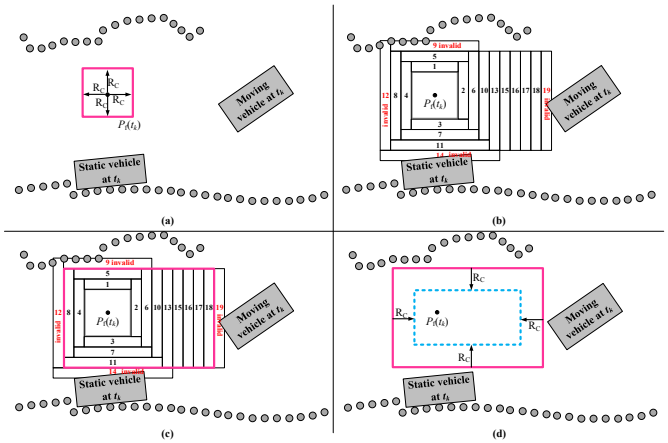

Fig. 3. Procedures to specify a local box at t_k for P_{f}. Note that the dashed box in (d) is the desired local box.

3.2 Iterative Computation Framework

The preceding subsection has presented how to formulate $\mathrm{P_0}$, the numerical solution to which is the desired on-road trajectory. Instead of solving $\mathrm{P_0}$ directly, this subsection introduces an iterative computation strategy that runs faster.

Our proposal is inspired by the fact that the only nonlinear constraints in $\mathrm{P_0}$ are i) the kinematic constraints $f=0$, and ii) the disc center definitions (3a). If they are linearized or softened as part of the cost function, $\mathrm{P_0}$ would become a linearly constrained nonlinear optimization problem, which is easier to solve.

This work chooses to soften the aforementioned nonlinear constraints: i) $f=0$ is softened as a polynomial Ψ, and ii) (3a) is softened as a polynomial Υ. Concretely, Ψ is defined as

$$\Psi=\int_{\tau=0}^{t_\mathrm{f}}\left\|f\left(\mathrm{x}(\tau),\ \mathrm{u}(\tau)\right)\right\|^2\cdot\mathrm{d}\tau. \tag{6a}$$

Similarly, Υ measures the violation degree of each equality in (3a), e.g., the first equality of (3a) is softened as

$$\int_{\tau=0}^{t_\mathrm{f}}\left|x_\mathrm{f}(\tau)-x(\tau)-\frac{1}{4}(3\mathrm{L_W}+3\mathrm{L_F}-\mathrm{L_R})\cdot\cos\theta(\tau)\right|^2\cdot\mathrm{d}\tau. \tag{6b}$$

With Ψ and Υ at hand, we replace (2) with $(\Psi+\Upsilon)$, thereby formulating a new optimal control problem $\mathrm{P_1}$:

$$\begin{aligned}&\text{minimize } \Psi+\Upsilon,\\&\text{subject to}\\&h_\mathrm{box}\left(\mathbf{x}(t)\right)\le 0,\ h_\mathrm{box}\left(\mathbf{u}(t)\right)\le 0,\\&\text{Eq. (4)},\\&\mathbf{x}(0)=\mathbf{x}_\mathrm{init},\ \mathbf{u}(0)=\mathbf{u}_\mathrm{init},\\&g\left(\mathbf{x}(\mathrm{t_f}),\ \mathbf{u}(\mathrm{t_f})\right)\le 0.\end{aligned} \tag{7}$$

Since (7) only contains inequality box constraints, it never suffers from the risk of being infeasible. Solving (7) numerically is easy and fast, but if the cost function value is larger than 0, then the derived trajectory violates $f=0$ and/or (3a). The violations occur if the constructed tunnels are not appropriate. To address this issue, it is natural to consider updating the tunnels according to the optimized trajectory, thereby updating $\mathrm{P_1}$. This process continues until the optimized cost function value becomes 0, which means a feasible solution (denoted as χ) to the original problem $\mathrm{P_0}$ is derived. Generally, starting a constrained optimization process with a feasible solution is easier than with an infeasible one, thus $\mathrm{P_0}$ with the latest tunnels is numerically solved, which is warm-started by χ. Through this, an optimal trajectory is obtained finally.

As a summary of this section, the pseudo-codes of our proposed on-road trajectory planner are presented as follows.

Algorithm 1. On-road Trajectory Planner in Cartesian Frame

***Input*:** Road scenario setups, initial configuration, and parametric settings;
***Output*:** A local on-road trajectory for the ego vehicle;

Generate a coarse trajectory χ_coarse via a search-and-sample method;
Construct tunnel-based constraints according to χ_coarse;
Form optimal control problem $\mathrm{P_1}$;
Set χ_coarse as the initial guess, solve $\mathrm{P_1}$ numerically, denote the derived optimum as χ_precise, and record $(\Psi+\Upsilon)$;
Initialize $iter=0$;
while $(\Psi+\Upsilon>10^{-2})$, **do**
  $iter\leftarrow iter+1$;
  if $(iter>\mathrm{Iter_{max}})$, **then**
    return with a failure;
  end if
  Update tunnel-based constraints according to χ_precise;
  Update optimal control problem $\mathrm{P_1}$;
  Set χ_precise as the initial guess, solve $\mathrm{P_1}$ numerically, use the derived optimum to update χ_precise, and update $(\Psi+\Upsilon)$;
end while
Update tunnel-based constraints according to χ_precise;
Update optimal control problem $\mathrm{P_0}$ according to the latest tunnels;
Set χ_precise as the initial guess, solve $\mathrm{P_0}$ numerically;
Output the derived optimal trajectory;
return with success.

4. SIMULATIONS

Simulations were performed in MATLAB 2014b and executed on an i5-8250U CPU with 8 GB RAM that runs at 1.60 × 2 GHz. Basic parametric settings are listed in Table 1. An urban road segment is selected as the driving scenario, which contains a right-turn and a U-turn (Fig. 4). The driving status of each surrounding social vehicle is randomly determined. More details about the parametric settings and scenario setups are provided in the source codes at https://github.com/libai1943/OnRoadPlanner_IFAC2020.

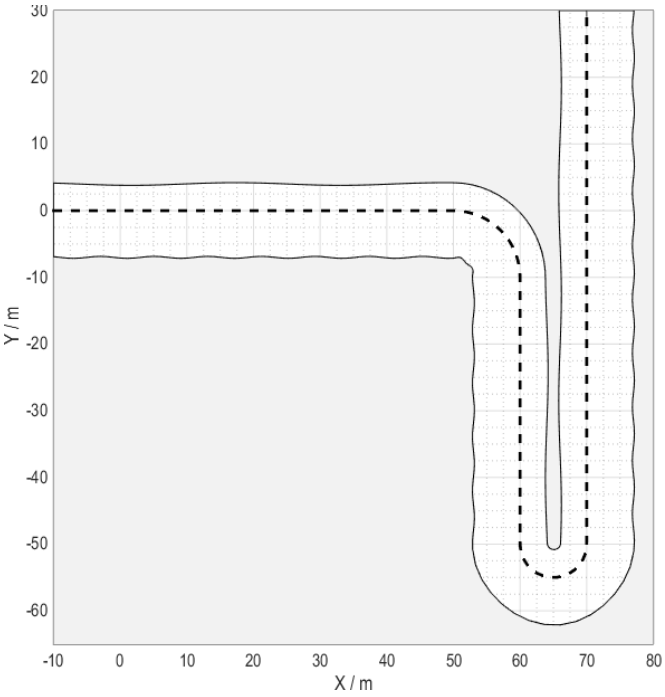

Fig. 4. On-road driving scenario in our simulations.

Table 1. Basic parametric settings.

Parameter(s)	Description	Setting(s)
$\mathrm{L_W}$, $\mathrm{L_R}$, $\mathrm{L_F}$, $\mathrm{L_B}$	Geometric size of ego vehicle	2.80 m, 0.929 m 0.96 m, 1.942 m
$\mathrm{t_f}$	Specified terminal moment	8.0 s
$\mathrm{a_{max}}$, $\mathrm{v_{max}}$, Φ_max, Ω_max	Bounds on $\lvert a(t)\rvert$, $v(t)$, $\lvert\phi(t)\rvert$, and $\lvert\omega(t)\rvert$	5 m/s^2, 25 m/s 0.7 rad, 1.0 rad/s
$\mathbf{x}_\mathrm{init}$	Initial state of ego vehicle	$[0,-0.78,0,0.18,20]$
$\mathbf{w}_1,\mathbf{w}_2$	Weighting vectors in (2)	$[0,0,0,0]^\mathrm{T}$, $[1,10]^\mathrm{T}$
$\mathrm{N_{box}}$	Number of local boxes in constructing each tunnel	160
$\mathrm{Iter_{max}}$	Maximum allowable iterations in Algorithm 1	6

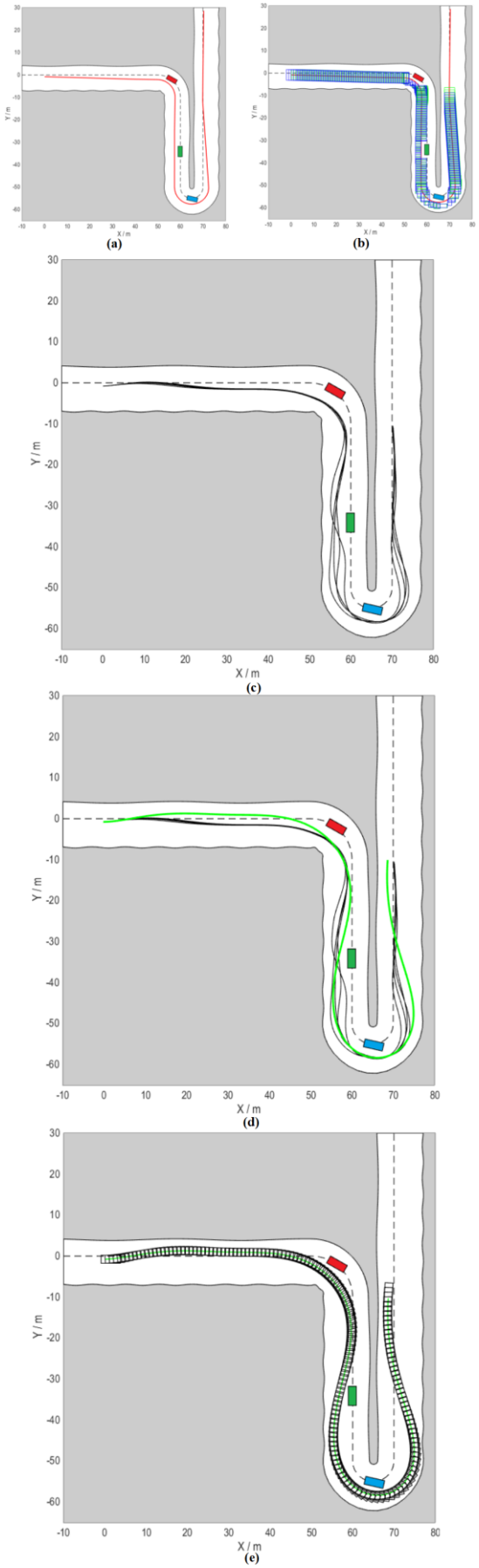

Fig. 5. Intermediate and final results derived by Algorithm 1 when there are three static obstacles in the environment.

Let us begin with a simple case with three static obstacles. Fig. 5a shows the derived coarse trajectory, Fig. 5b plots the tunnels constructed along that coarse trajectory, Fig. 5c depicts the intermediate solutions to P_1 in the iterations, Fig. 5d shows the final on-road trajectory, and Fig. 5e plots the footprints. The coarse trajectory in Fig. 5a is planned in the Frenet frame, thus it does not care about the curvature limitation issue when the ego vehicle is taking the right turn or U-turn. By contrast, the optimized trajectory depicted in Fig. 5d reflects the vehicle kinematics better. The footprints plotted in Fig. 5e indicate that the adopted tunnel-based formulation is efficacious to avoid collisions. The performance of Algorithm 1 to handle moving obstacles is demonstrated in Fig. 6.

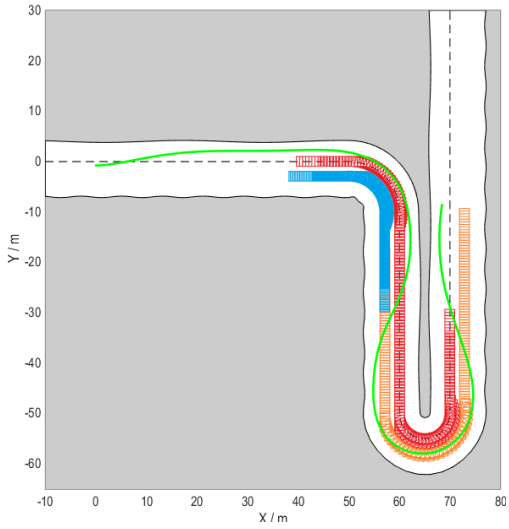

Fig. 6. On-road trajectory derived by Algorithm 1 when there are three moving obstacles in the environment.

To further investigate the benefit of building the iterative computation architecture as in Algorithm 1, we define a comparative algorithm (denoted as Algorithm 2) and make comparisons w.r.t. computational time and success rate in 500 randomly generated simulation cases wherein the obstacles are moving.

Algorithm 2. A comparative planner that directly solves P_0

***Input*:** Road scenario setups, initial configuration, and parametric settings;
***Output*:** A local on-road trajectory for the ego vehicle;

1. Generate a coarse trajectory χ_{coarse} via a search-and-sample method;
2. Construct tunnel-based constraints according to χ_{coarse};
3. Form optimal control problem P_0;
4. Set χ_{coarse} as the initial guess, solve P_0 numerically;
5. **if** the numerical optimization fails, **then**
6. **return** with a failure;
7. **end**
8. Output the derived optimal trajectory;
9. **return** with success.

Table 2. Comparative simulation results.

Algorithm ID	Average CPU time/sec	Success rate
Algorithm 1	4.484	94.8 %
Algorithm 2	1.267	11.6 %

Comparisons between Algorithms 1 and 2 reflect the necessity of designing an iterative computation framework. Algorithm 2 is not efficient because i) all the coupled constraints in P_0 have to be simultaneously handled, and more importantly ii) the tunnels paved along the coarse trajectory do not contain a feasible trajectory, i.e., P_0 may be an infeasible problem in Algorithm 2. By contrast, Algorithm 1 can handle most of the simulation cases. Algorithm 1 takes longer time than Algorithm 2 because it involves repeated tunnel construction procedures, each of which is time-consuming when executed in Matlab. The reported CPU time in Table 2 could be largely shortened if the source codes are written in the C++ compilation environment.

Typical simulation results derived by Algorithm 1 are shown in a video, which is available at https://www.bilibili.com/video/BV13541147GJ/.

5. CONCLUSIONS

This paper has proposed an iterative computation framework for precisely and fast solving on-road trajectory planning problems in the Cartesian frame. Highlights of the paper include i) the on-road trajectory planning scheme is formulated in the Cartesian frame rather than the commonly used Frenet frame, and ii) an iterative framework is proposed to facilitate the numerical optimization difficulties. One of our future works is to handle the failed simulation cases by providing fault-tolerant solutions.

ACKNOWLEDGMENTS

This work was supported by the Fundamental Research Funds for the Central Universities, the Natural Sciences and Engineering Research Council of Canada, as well as the JDX Automated Driving R&D Center of JD.com. Bai Li thanks Z. Yin and L. Zhang for the support given during this work.

REFERENCES

Ajanovic, Z., et al. (2018). Search-based optimal motion planning for automated driving. In *2018 IEEE/RSJ International Conference on Intelligent Robots Systems (IROS)* (4523–4530).

Barfoot, T. D., & Clark, C. M. (2004). Motion planning for formations of mobile robots. *Robotics and Autonomous Systems*, 46(2), 65–78.

Bender, J. P., Dang, T., & Stiller, C. (2014). Trajectory planning for Bertha—A local, continuous method. In *2014 IEEE Intelligent Vehicles Symposium (IV)* (450–457).

Chen, J. Y., Zhan, W., & Tomizuka, M. (2019). Autonomous driving motion planning with constrained iterative LQR. *IEEE Transactions on Intelligent Vehicles*, 4(2), 244–254.

Claussmann, L., Revilloud, M., Gruyer D., & Glaser S. (2020). A review of motion planning for highway autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(5), 1826–1848.

Ding, W., Zhang, L., Chen, J., & Shen, S. (2019). Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor. *IEEE Robotics and Automation Letters*, 4(3), 2997–3004.

Hu, X., et al. (2018). Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles. *Mechanical Systems and Signal Processing*, 100, 482–500.

Levinson, J., et al. (2011). Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)* (163–168).

Li, B., & Shao, Z. (2015). A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles. *Knowledge-Based Systems*, 86, 11–20.

Li, B., et al. (2019a). Tractor-trailer vehicle trajectory planning in narrow environments with a progressively constrained optimal control approach. *IEEE Transactions on Intelligent Vehicles*, Accepted.

Li, B., Jia, N., Li, P., & Li Y. (2019b). Incrementally constrained dynamic optimization: A computational framework for lane change motion planning of connected and automated vehicles. *Journal of Intelligent Transportation Systems*, 23(6), 557–568.

Li, B., et al. (2020). Maneuver planning for automatic parking with safe travel corridors: A numerical optimal control approach. In *2020 European Control Conference (ECC)*, 1993–1998.

Lim, W., Lee, S., Sunwoo, M., & Jo, K. (2018). Hierarchical trajectory planning of an autonomous car based on the integration of a sampling and an optimization method. *IEEE Transactions on Intelligent Transportation Systems*, 19(2), 613–626.

Ma, L., et al. (2015). Efficient sampling-based motion planning for on-road autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 16(4), 1961–1976.

McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *2011 IEEE International Conference on Robotics and Automation (ICRA)* (4889–4895).

Meng, Y., Wu, Y., Gui, Q., & Liu, L. (2019). A decoupled trajectory planning framework based on the integration of lattice searching and convex optimization. *IEEE Access*, 7, 130530–130551.

Miller, C., Pek, C., & Althoff, M. (2018). Efficient mixed-integer programming for longitudinal and lateral motion planning of autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)* (1954–1961).

Wang, H., Kearney, J., & Atkinson, K. (2002). Robust and efficient computation of the closest point on a spline curve. In *5th International Conference on Curves and Surfaces* (397–406).

Werling, M., Ziegler, J., Kammel, S., & Thrun, S. (2010). Optimal trajectory generation for dynamic street scenarios in a Frenet frame. In *2010 IEEE International Conference on Robotics and Automation (ICRA)* (987–993).

Xu, W., Pan, J., Wei, J., & Dolan, J. M. (2014). Motion planning under uncertainty for on-road autonomous driving. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2507–2512).