Online Trajectory Replanning for Sudden Environmental Changes During Automated Parking: A Parallel Stitching Method

Bai Li, *Member, IEEE*, Zhuyan Yin, Yakun Ouyang, *Student Member, IEEE*, Youmin Zhang, *Senior Member, IEEE*, Xiang Zhong, and Shiqi Tang

***Abstract*—Trajectory planning for automated parking has been widely known as more challenging than that for on-road driving due to the nonconvex kinematics, high-dimensional collision-avoidance constraints, and difficulty to determine the global optimum among many local optima. Changes in a dynamic environment easily make a previously planned parking trajectory invalid. Compared with the on-road planners which evade newly emerged obstacles via a nudge or side pass, a parking planner has to replan a completely different trajectory for evasion. A qualified online trajectory replanner should run fast, ensure trajectory continuity, and avoid extra halfway stops. This paper proposes a parallel stitching strategy to fulfill these demands. When the ego vehicle tracks an originally planned parking trajectory online, an evasive trajectory begins to be replanned once a new obstacle is found to block the way. Thereafter, a connective trajectory is planned, the two ends of which are future poses along the original trajectory and the evasive trajectory, respectively. The two ends of the connective trajectory are chosen by greedily evaluating various candidates via parallel computation, which ensures that our replanner runs fast with high solution quality. According to our real-world experiments and simulations, the proposed replanner outperforms the existing ones w.r.t. solution speed and quality. As an interesting feature, our proposed replanner also supports switching to a better homotopy class online.**

***Index Terms*—Automated parking, trajectory planning, trajectory replanning, numerical optimal control, parallel stitching.**

Manuscript received 14 January 2022; revised 19 February 2022; accepted 26 February 2022. Date of publication 7 March 2022; date of current version 24 October 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62103139, in part by the Fundamental Research Funds for the Central Universities under Grant 531118010509, in part by the Natural Sciences and Engineering Research Council of Canada, and in part by the Natural Science Foundation of Hunan Province, China under Grant 2021JJ40114. *(Corresponding authors: Yakun Ouyang; Xiang Zhong.)*

Bai Li, Zhuyan Yin, Yakun Ouyang, Xiang Zhong, and Shiqi Tang are with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, China (e-mail: libai@zju.edu.cn; 1026078242@qq.com; yakun@hnu.edu.cn; zx5587@126.com; tangshiqi@hnu.edu.cn).

Youmin Zhang is with the Department of Mechanical, Industrial and Aerospace Engineering, Concordia Institute of Aerospace Design and Innovation, Concordia University, Montreal, QC H3G 1M8, Canada (e-mail: ymzhang@encs.concordia.ca).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TIV.2022.3156429.

Digital Object Identifier 10.1109/TIV.2022.3156429

I. Introduction

AUTOMATED parking refers to driving an autonomous vehicle from an initial pose to a desired goal pose in a parking lot, during which the vehicle should avoid collisions with the surrounding obstacles [1]–[4]. As a critical module in an automated parking system, trajectory planning is responsible for finding a kinematically feasible and collision-free curve with the traverse time and energy minimized [5]–[12]. In contrast with the on-road planners that generate smooth trajectories [13], [14], a parking-oriented planner deals with more complex problems. Herein, the complicity is reflected by the following few factors. First, the equations used to model the vehicle kinematics are nonconvex. Second, the geometric constraints used to restrict vehicle-to-obstacle collisions (denoted as the collision-avoidance constraints) are highly nonconvex and even non-differentiable [2]. Third, the dimension of the collision-avoidance constraints is tightly combined with the obstacle density, which easily makes the runtime of a trajectory planner change drastically. Fourth, huge numbers of local optima are observed in a parking planning problem, which are caused not only by the topological homotopy classes [15], but also by the maneuver times and the phases of the endpoint angle [5]. These factors account for the reason why most of the well-developed on-road trajectory planners cannot handle parking cases [16], [17]. Therefore, automated parking trajectory planning is a special task in an autonomous driving system.

Changes in a dynamic environment can easily make a previously planned parking trajectory invalid. Compared with the on-road planners that can easily evade newly emerging obstacles through a nudge or side pass, a parking planner may have to replan a completely different trajectory for evasion (Fig. 1). This means that the originally planned parking trajectory is no longer useful because it cannot serve as a valid initial guess to warm-start the replanning procedure. As another challenge, the trajectory replanning process should be fast because the ego vehicle is moving with a nonzero velocity online. If the replanner runs slowly, then the ego vehicle would deviate much from the initial pose considered in the replanner, thus making the replanned trajectory invalid. Intuitively, one may stop the vehicle once it detects new obstacles along the predefined parking trajectory, replan an evasive trajectory, and restart the trajectory tracking process. However, this behavior renders more

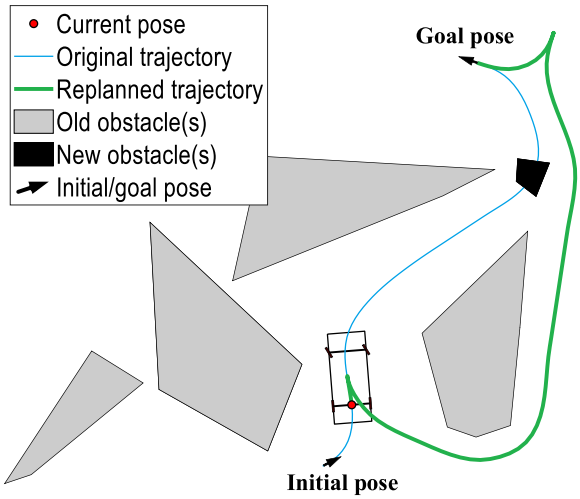

Fig. 1. Originally planned trajectory and an online replanned trajectory from the spot that a new obstacle is detected at the current pose. Note that the two trajectories are completely different w.r.t. homotopy class and maneuver time, which make replanning for parking far more challenging than that for on-road driving.

intermediate stops, which may further lead to delays and emissions. Therefore, it deserves to investigate how to replan a fast and high-quality trajectory in automated parking.

Broadly speaking, environmental changes in a parking-related scenario are caused by the existence of 1) unpredictable moving agents such as pedestrians, 2) obstacles that were previously out of the vision range, or 3) obstacles suddenly drop to the terrain where the ego vehicle moves. The first two factors are related to perception and prediction techniques while the third factor is the focus of this study.

A. Related Works

Generally speaking, online trajectory replanning problems can be handled by three types of methods, which are reviewed as follows.

The first type of method plans fail-safe trajectories. Concretely, an emerging obstacle is regarded as a failure if it blocks the original trajectory. Assuming that such failures may happen in the near future, a fail-safe planner generates evasive trajectories as precautions [18]. However, this method is more suitable for on-road cruising scenarios because the underlying failures in an unstructured parking scenario are difficult to enumerate.

The second type of method replans a local evasive trajectory around the emerging obstacles so that the ego vehicle bypasses them via nudge behaviors locally. However, this method is still not fully suitable for automated parking schemes because the newly added obstacles easily make thorough changes to the parking trajectory rather than minor ones as depicted in Fig. 1.

The third type of method is about quickly replanning an evasive trajectory online with the latest scenario considered. Given that the ego vehicle is not static during the online replanning process, it easily renders a mismatch if the evasive trajectory is planned from the vehicle's current pose [19]. This is because the ego vehicle has moved to a different spot and no longer stays at the current pose when the evasive trajectory is available. To address this issue, a stitching method was proposed in [20], which would plan an evasive trajectory from a future pose rather than the current pose along the original trajectory. This method does not involve any mismatch if the evasive trajectory is derived before the ego vehicle reaches the specified future pose. This stitching strategy sounds intuitively reasonable but it easily renders solution failures. The rationale behind this argument is given as follows. The existing parking trajectory planners that run sufficiently fast to handle the concerned replanning task mainly include sampling-and-search-based and numerical-optimization-based methods [21]. A numerical optimizer typically relies on a sampling-and-search-based planner to warm start [5], [22], [23], which indicates that nearly all parking-oriented planners rely on sampling-and-search-based methods. If sampling in the state space, a sampling-and-search-based method is unaware of the control profiles, thus it may require the ego vehicle to go backward when the forward velocity and acceleration are still large. If so, a solution failure occurs because the planned trajectories are beyond the ego vehicle's physical limits. Sampling in the control space can address this issue but it involves the curse of dimensionality, which consumes more runtime. Even if the solution failure risk can be ignored, some prevalent sampling-and-search-based methods such as the hybrid A$*$ algorithm [24] do not ensure finding a good homotopy class, which makes the solution quality in question. Given that direct stitching is imperfect due to the embedded sampling-and-search-based planners, Jang *et al.* [18] proposed a computational stitching strategy with the help of a transitional trajectory. Concretely, an evasive trajectory is replanned the moment that new obstacles are detected online; thereafter, a connective trajectory is quickly planned to steer the ego vehicle toward the evasive trajectory via model predictive control (MPC); a complete replanning result is constructed by sequentially combining the original trajectory in part, the connective trajectory, and the evasive trajectory in part. However, this method does not ensure that such a connective trajectory is always collision-free.

As a conclusion, there has not been a qualified trajectory replanner that runs fast with a high solution quality for parking-related cases.

B. Motivation and Contribution

The purpose of this paper is to develop a parking-oriented online trajectory replanner for suddenly appearing obstacles. We expect that the replanner runs fast without extra halfway stops, considers the vehicle kinematics well, ensures safety, and ensures sufficient continuity of the replanned trajectory. To that end, we choose the third type of method mentioned in Section I.A because the first two types cannot handle generic parking cases. To reduce the side effects of sampling-and-search-based planners, we follow the computational stitching strategy proposed in [18] and modify it. The core contribution of this paper lies in the proposal of a parallel stitching strategy for parking-oriented online trajectory replanning problems. In our proposed method, candidate connective trajectories with their two ends imposed

to various spots along the original and evasive trajectories are quickly planned through parallel computation so that different stitching attempts are tried before the minimal-cost one that passes the collision check is greedily selected. Our proposal addresses the issues in the direct stitching method and the computational stitching method w.r.t. the solution feasibility, speed, and optimality.

C. Organization

In the remainder of this paper, the replanning scheme is stated in Section II. Our proposed parallel stitching method is introduced in Section III. Simulation and real-world experimental results are discussed in Section IV. Conclusions are drawn in Section V finally.

II. Problem Statement

This section defines the concerned parking-oriented trajectory replanning problem. Suppose that an original parking trajectory is planned offline before the vehicle begins to move at $t = 0$. We assume that the original trajectory is valid, i.e., it connects the initial and goal poses while ensuring kinematic feasibility and collision avoidance. Let us denote the original trajectory as $Traj_{\text{original}}(t), t \in [0, \mathrm{t_f}]$,where $\mathrm{t_f}$ stands for the moment that the ego vehicle should reach the goal pose if it perfectly tracks $Traj_{\text{original}}$. At any moment $t \in [0, \mathrm{t_f}]$, the state and control profiles related to the vehicle kinematics are all recorded in $Traj_{\text{original}}(t)$. This work adopts the well-known bicycle model [5] to describe the ego vehicle's kinematic principle during the low-speed parking process:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \\ v(t) \\ \phi(t) \end{bmatrix} = \begin{bmatrix} v(t)\cdot\cos\theta(t) \\ v(t)\cdot\sin\theta(t) \\ v(t)\cdot\tan\phi(t)/\mathrm{L_W} \\ a(t) \\ \omega(t) \end{bmatrix}, \forall t. \tag{1}$$

In (1), $(x(t), y(t))$ refers to the coordinate value of the ego vehicle's rear-axle midpoint in the Cartesian frame, $\theta(t)$ is the yaw angle, $v(t)$ is the longitudinal velocity, $a(t)$ denotes the acceleration, $\phi(t)$ is the steering angle, $\omega(t)$ is the angular velocity of $\phi(t)$, and $\mathrm{L_W}$ denotes the vehicle wheelbase. The state profiles comprise $x(t), y(t), \theta(t), v(t)$, and $\phi(t)$, while the control profiles include $a(t)$ and $\omega(t)$.

Suppose that there used to be $\mathrm{N_{OBS}}$ static obstacles (denoted by obs_{old}) taken into consideration when planning the original trajectory prior to $t = 0$. After the ego vehicle begins to track $Traj_{\text{original}}$ at $t = 0$, a static obstacle obs_{new} is newly detected at $t = \mathrm{t_0} < \mathrm{t_f}$ and a collision to obs_{new} happens soon if the ego vehicle continues to track the original trajectory $Traj_{\text{original}}$. The trajectory replanning task is about finding a new trajectory from $t = \mathrm{t_0}$ to a future moment t_{end} (unknown *a priori*) to guide the ego vehicle to the predefined goal pose without colliding with $obs_{\text{new}} \cup obs_{\text{old}}$.

This work uses an optimal control problem (OCP) to describe the trajectory replanning scheme. The constraints in the concerned OCP comprised the kinematic constraints, two-point boundary constraints, and collision-avoidance constraints. The kinematic constraints include (1) and

$$\begin{aligned} &-\mathrm{a_{max}} \le a(t) \le \mathrm{a_{max}}, \\ &-\mathrm{v_{max}} \le v(t) \le \mathrm{v_{max}}, \\ &-\Omega_{\text{max}} \le \omega(t) \le \Omega_{\text{max}}, \\ &-\Phi_{\text{max}} \le \phi(t) \le \Phi_{\text{max}}, \forall t. \end{aligned} \tag{2}$$

The two-point boundary constraints are

$$\begin{bmatrix} x(\mathrm{t_0}) \\ y(\mathrm{t_0}) \\ \theta(\mathrm{t_0}) \\ v(\mathrm{t_0}) \\ \phi(\mathrm{t_0}) \\ a(\mathrm{t_0}) \\ \omega(\mathrm{t_0}) \end{bmatrix} = \begin{bmatrix} Traj_{\text{original}}(\mathrm{t_0}).x \\ Traj_{\text{original}}(\mathrm{t_0}).y \\ Traj_{\text{original}}(\mathrm{t_0}).\theta \\ Traj_{\text{original}}(\mathrm{t_0}).v \\ Traj_{\text{original}}(\mathrm{t_0}).\phi \\ Traj_{\text{original}}(\mathrm{t_0}).a \\ Traj_{\text{original}}(\mathrm{t_0}).\omega \end{bmatrix}, \begin{bmatrix} x(t_{\text{end}}) \\ y(t_{\text{end}}) \\ \theta(t_{\text{end}}) \\ v(t_{\text{end}}) \\ \phi(t_{\text{end}}) \\ a(t_{\text{end}}) \\ \omega(t_{\text{end}}) \end{bmatrix} = \begin{bmatrix} \mathrm{x_{end}} \\ \mathrm{y_{end}} \\ \theta_{\text{end}} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3}$$

wherein $(\mathrm{x_{end}}, \mathrm{y_{end}}, \theta_{\text{end}})$ denotes the predefined goal pose. The collision-avoidance constraints are briefly described as

$$footprint\,(x(t), y(t), \theta(t)) \not\subset obs_{\text{new}} \cup obs_{\text{old}}, \forall t \in [\mathrm{t_0}, t_{\text{end}}]. \tag{4}$$

Herein, $footprint(\cdot)$ is a mapping from the vehicle pose to its footprint, thus (4) ensures that the ego vehicle body does not overlap with any of the obstacles in the environment throughout the entire parking process. A concrete version of (4) can be presented by various methods such as [2], [5], or [23].

The cost function of the OCP is simply defined as the traverse time:

$$J = t_{\text{end}} - \mathrm{t_0}, \tag{5}$$

As a summary, the concerned trajectory replanning task can be presented as

$$\begin{aligned} &\mathrm{Minimize}(5), \\ &\mathrm{s.t.,kinematic\ constraints}(1), (2); \\ &\text{two point boundary constraints}(3); \\ &\text{collision avoidance constraints}(4). \end{aligned} \tag{6}$$

III. Online Trajectory Replanning Method

Suppose that the ego vehicle would collide with a buffer region of obs_{new} at $t = \mathrm{t_1}$ if it continues to track the original trajectory after finding obs_{new} at $t = \mathrm{t_0}$. Herein, the buffer of obs_{new} is created via inflating obs_{new} by a small user-specified distance $\mathrm{S_{buffer}} > 0$. From Section III.A to III.D, we temporarily hold an assumption that $(\mathrm{t_1} - \mathrm{t_0})$ is sufficiently long so that the ego vehicle has sufficient time to finish the online trajectory replanning. In the last subsection, we introduce a fail-safe strategy to do without this assumption.

A. Overall Framework

As shown in Fig. 2, the entire replanned trajectory comprises three consecutive segments.

The first segment is part of the original trajectory. This means that we follow the basic stitching strategy so that the ego vehicle continues to track the original trajectory for some time after

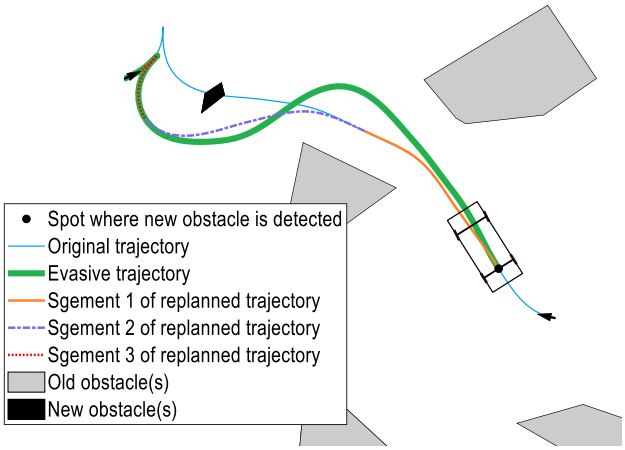

Fig. 2. Schematics on the components of a replanned trajectory.

$t = \mathrm{t}_0$, during which the ego vehicle keeps thinking about how to evade $obs_{\text{new}} \cup obs_{\text{old}}$. We request the online thinking process to complete within $\mathrm{T}_{\text{think}}$ seconds, i.e., the thinking process finishes before $t = \mathrm{t}_0 + \mathrm{T}_{\text{think}}$. The second segment is denoted as a connective trajectory, while the third one is part of an evasive trajectory.

Interestingly, these three segments are determined in inverse order as opposed to their occurrences along the timeline. Concretely, from $t = \mathrm{t}_0$ to $t = \mathrm{t}_0 + \mathrm{T}_{\text{think}}$, an evasive trajectory is computed first, followed by the identification of a connective trajectory, the two ends of which naturally determine the stitching spots along the original and evasive trajectory.

B. Step One: Planning an Evasive Trajectory

Once obs_{new} is detected at $t = \mathrm{t}_0$, an evasive trajectory begins to be planned based on the latest scenario layout. The evasive trajectory planning scheme is about generating a trajectory to connect the ego vehicle's current pose at $Traj_{\text{original}}(\mathrm{t}_0)$ and the predefined goal pose $(\mathrm{x}_{\text{end}}, \mathrm{y}_{\text{end}}, \theta_{\text{end}})$. This scheme is similar to (6), except that some boundary constraints at $t = \mathrm{t}_0$ are reset. Concretely, the following settings are deployed to replace the corresponding ones listed in (3):

$$v(\mathrm{t}_0) = \phi(\mathrm{t}_0) = a(\mathrm{t}_0) = \omega(\mathrm{t}_0) = 0. \tag{7}$$

The usage of (7) indicates that the ego vehicle is treated as static at $(Traj_{\text{original}}(\mathrm{t}_0).x, Traj_{\text{original}}(\mathrm{t}_0).y, Traj_{\text{original}}(\mathrm{t}_0).\theta)$, which is not the true status of the ego vehicle at t_0. At this point, we clarify the following two points. First, this particular design reduces the solution failure risks if a sampling-and-search-based planner (such as the hybrid A$*$ algorithm) is involved in the evasive trajectory planning process, as we have mentioned in Section I.A. Second, although (7) seems to cause trajectory discontinuity at t_0, it is not really a problem because the stitching spot is not set to t_0, as we will explain later.

To solve the concerned evasive trajectory planning problem, this work adopts a lightweight two-layer parking trajectory planner [5], which searches a coarse trajectory via the hybrid A$*$ algorithm and thereafter finds a local optimum by solving a sequence of lightweight OCPs numerically. The resultant trajectory is denoted as $Traj_{\text{evasive}}(t)$, the horizon of which ranges from $t = \mathrm{t}_0$ to $\mathrm{t}_0 + t_{\text{end}}$. Notably, t_{end} becomes known when $Traj_{\text{evasive}}(t)$ is available; t_{end} no longer stands for the terminal moment of the parking process due to our special design in (7); $Traj_{\text{evasive}}(t)$ may not be locally optimal because [5] is a near-optimal planner, which sacrifices the optimality for the planning speed; $Traj_{\text{evasive}}(t)$ may not be close to the global optimum because the adopted planner [5] relies heavily on the hybrid A$*$ algorithm to provide an initial guess, but the hybrid A$*$ algorithm does not guarantee global optimality.

C. Step Two: Generating Connective Trajectory Candidates

The evasive trajectory planned in Section III.B is not directly useful because 1) it violates the vehicle's true state at t_0, and 2) the ego vehicle is no longer located at $Traj_{\text{original}}(\mathrm{t}_0)$ when the evasive trajectory is available. Given that the evasive trajectory is imperfect and not directly useful, one may flexibly use part of it. One may deploy a connective trajectory, the two ends of which link the original and evasive trajectories, respectively. If the two ends of the connective trajectory are smartly selected, the overall three-segment replanned trajectory would have a smaller cost value than simply using an evasive trajectory. Thus a connective trajectory deserves to be introduced.

This subsection introduces how to generate connective trajectory candidates, each of which connects the original trajectory and the evasive trajectory smoothly at its two ends. Let us sample $\mathrm{N}_{\text{original}}$ equidistant poses along the original trajectory in the temporal range $[\mathrm{t}_0 + \mathrm{T}_{\text{think}}, \mathrm{t}_{\text{brake}}]$. Herein, $\mathrm{t}_{\text{brake}} < \mathrm{t}_1$ denotes the latest moment for the ego vehicle to begin a full brake so that a collision with the buffer region of obs_{new} can be avoided. Thus $\mathrm{t}_{\text{brake}}$ is the latest moment that satisfies

$$\left| \frac{0 - Traj_{\text{original}}(\mathrm{t}_{\text{brake}}).v}{a_{max}} \right| + \mathrm{t}_{\text{brake}} < \mathrm{t}_1. \tag{8}$$

Without loss of generality, let us focus on the ith sampled pose (denoted as $Traj_{\text{original}}(\mathrm{t}_i^{\text{sample}})$) among all $\mathrm{N}_{\text{original}}$ ones. The pose $Traj_{\text{original}}(\mathrm{t}_i^{\text{sample}})$ is taken as the starting point of a candidate connective trajectory. The underlying endpoints of this candidate trajectory include $\mathrm{N}_{\text{evasive}}$ equidistant poses along the evasive trajectory $Traj_{\text{evasive}}(t)$ in the temporal range $[min(\mathrm{t}_i^{\text{sample}} + \Delta\mathrm{T}_{\text{start}}, t_{\text{end}}), min(\mathrm{t}_i^{\text{sample}} + \Delta\mathrm{T}_{\text{end}}, t_{\text{end}})]$. $\Delta\mathrm{T}_{\text{start}}$ and $\Delta\mathrm{T}_{\text{end}}$ are user-specified parameters, which are set in proportional with the already known t_{end}. As an example, let us set the jth pose along the evasive trajectory as the endpoint of the candidate connective trajectory and denote it as $Traj_{\text{evasive}}(\mathrm{t}_j^{\text{sample2}})$. We introduce a variable called t_{OE} to describe the movement that the ego vehicle reaches the end endpoint $Traj_{\text{evasive}}(\mathrm{t}_j^{\text{sample2}})$. Herein, it deserves to emphasize that t_{OE} has no relationship with $\mathrm{t}_j^{\text{sample2}}$ at all. The following two-point boundary constraints are needed if one formulates a

candidate connective trajectory planning problem:

$$\begin{bmatrix} x(\mathrm{t}_i^{\text{sample}}) \\ y(\mathrm{t}_i^{\text{sample}}) \\ \theta(\mathrm{t}_i^{\text{sample}}) \\ v(\mathrm{t}_i^{\text{sample}}) \\ \phi(\mathrm{t}_i^{\text{sample}}) \\ a(\mathrm{t}_i^{\text{sample}}) \\ \omega(\mathrm{t}_i^{\text{sample}}) \end{bmatrix} = \begin{bmatrix} Traj_{\text{original}}(\mathrm{t}_i^{\text{sample}}).x \\ Traj_{\text{original}}(\mathrm{t}_i^{\text{sample}}).y \\ Traj_{\text{original}}(\mathrm{t}_i^{\text{sample}}).\theta \\ Traj_{\text{original}}(\mathrm{t}_i^{\text{sample}}).v \\ Traj_{\text{original}}(\mathrm{t}_i^{\text{sample}}).\phi \\ Traj_{\text{original}}(\mathrm{t}_i^{\text{sample}}).a \\ Traj_{\text{original}}(\mathrm{t}_i^{\text{sample}}).\omega \end{bmatrix},$$
$$\begin{bmatrix} x(t_{\text{OE}}) \\ y(t_{\text{OE}}) \\ \theta(t_{\text{OE}}) \\ v(t_{\text{OE}}) \\ \phi(t_{\text{OE}}) \\ a(t_{\text{OE}}) \\ \omega(t_{\text{OE}}) \end{bmatrix} = \begin{bmatrix} Traj_{\text{evasive}}(\mathrm{t}_j^{\text{sample2}}).x \\ Traj_{\text{evasive}}(\mathrm{t}_j^{\text{sample2}}).y \\ Traj_{\text{evasive}}(\mathrm{t}_j^{\text{sample2}}).\theta \\ Traj_{\text{evasive}}(\mathrm{t}_j^{\text{sample2}}).v \\ Traj_{\text{evasive}}(\mathrm{t}_j^{\text{sample2}}).\phi \\ Traj_{\text{evasive}}(\mathrm{t}_j^{\text{sample2}}).a \\ Traj_{\text{evasive}}(\mathrm{t}_j^{\text{sample2}}).\omega \end{bmatrix}. \tag{9}$$

A candidate connective trajectory is obtained through solving the following time-optimal OCP:

$$\begin{aligned} &\text{Minimize } \left(t_{\text{OE}} - \mathrm{t}_i^{\text{sample}}\right), \\ &\text{s.t., kinematic constraints}(1), (2); \\ &\text{two point boundary constraints}(9). \end{aligned} \tag{10}$$

In solving OCP (10) numerically, one discretizes it as a nonlinear programming (NLP) problem and solves the NLP via a gradient-based optimizer such as the interior-point method (IPM) [2]. To facilitate the gradient-based optimization process, an initial guess is provided by the Reeds-Shepp method [25] or the line connection method [26].

Compared with (6), (10) is easier to solve because it does not include any collision-avoidance constraint. If the solution to (10), which is denoted as $Traj_{\text{candidate\_}ij}(t)$, does not involve collisions with the obstacles [27], then it is approved and stored for future usage.

Within the online computation period $[\mathrm{t}_0, \mathrm{t}_0 + \mathrm{T}_{\text{think}}]$, an evasive trajectory should be planned first, followed by planning $\mathrm{N}_{\text{original}} \cdot \mathrm{N}_{\text{evasive}}$ connective trajectory candidates in a sequence. Note that the $\mathrm{N}_{\text{original}} \cdot \mathrm{N}_{\text{evasive}}$ independent candidates can be planned in parallel if multiple CPU cores are available. The calculation process is terminated at $t = \mathrm{t}_0 + \mathrm{T}_{\text{think}}$ even if not all of the $\mathrm{N}_{\text{original}} \cdot \mathrm{N}_{\text{evasive}}$ combinations have been tried.

D. Step Three: Selecting an Overall Replanned Trajectory

Suppose that $\mathrm{N_A}$ valid candidate connective trajectories have been stored by the moment $t = \mathrm{t}_0 + \mathrm{T}_{\text{think}}$. Each candidate trajectory is used to build an overall replanning trajectory. Without losing generality, we focus on one candidate trajectory $Traj_{\text{candidate\_}ij}(t)$ among all of the $\mathrm{N_A}$ ones. Suppose that the candidate connective trajectory $Traj_{\text{candidate\_}ij}(t)$ connects the original trajectory at the spot $Traj_{\text{original}}(\mathrm{t}_i^{\text{sample}})$ and connects the evasive trajectory at $Traj_{\text{evasive}}(\mathrm{t}_j^{\text{sample2}})$, the overall replanned trajectory is defined as

$$Traj_{\text{replanned}}(t) = \begin{cases} Traj_{\text{original}}(t), \text{if} t \in [\mathrm{t}_0, \mathrm{t}_i^{\text{sample}}) \\ Traj_{\text{candidate\_}ij}(t), \text{if} t \in [\mathrm{t}_i^{\text{sample}}, t_{\text{OE}}) \\ Traj_{\text{evasive}}(t), \text{if} t \in \\ [t_{\text{OE}}, t_{\text{OE}} + t_{\text{end}} - \mathrm{t}_j^{\text{sample2}}] \end{cases}. \tag{11}$$

A replanned trajectory is built as per (11) for each of the $\mathrm{N_A}$ candidate trajectories. The overall replanned trajectory with the lowest cost function value (i.e., $t_{\text{OE}} + t_{\text{end}} - \mathrm{t}_j^{\text{sample2}}$) is selected as the output of our proposed online trajectory replanner.

E. Removal Basic Assumption via a Fail-Safe Strategy

A trajectory replanner has been introduced in the preceding few subsections. It is efficient if the assumption that the ego vehicle completes the online trajectory replanning process within $\mathrm{T}_{\text{think}}$ seconds would hold. However, this assumption does not always hold because 1) the adopted evasive trajectory planner does have probabilities to fail [22], and 2) all of the candidate connective trajectories may involve collisions, which results in $\mathrm{N_A} = 0$. If the evasive trajectory planning process fails during $t \in [\mathrm{t}_0, \mathrm{t}_0 + \mathrm{T}_{\text{think}}]$, then there is no need to consider the connective trajectory planning step further. If $\mathrm{N_A}$ turns out to be 0, then the online replanning process should be taken as a failure. If either of the aforementioned two events occurs, the ego vehicle runs into an emergency status and immediately brakes until it fully stops. The ego vehicle would not begin to move again until a valid trajectory is successfully planned from the vehicle's current stopped pose regardless of fast or slow planners one may use.

Another assumption we held in the preceding subsections is that $(\mathrm{t}_1 - \mathrm{t}_0)$ is sufficiently long. Without it, $\mathrm{t}_{\text{brake}} < \mathrm{t}_0 + \mathrm{T}_{\text{think}}$ is possible to hold, which is rather risky because the ego vehicle would encounter an inevitable collision status if no solution could be found by $t = \mathrm{t}_0 + \mathrm{T}_{\text{think}}$. To address this issue, we request that the ego vehicle should simply begin to brake at the moment t_0 until it fully stops if $\mathrm{t}_{\text{brake}} < \mathrm{t}_0 + \mathrm{T}_{\text{think}}$. The ego vehicle would begin to move again after a valid trajectory is planned successfully from the full-stop spot.

F. Overall Principle

Our proposed replanning method is summarized into the following pseudo-codes.

IV. Experimental Results and Discussion

This section reports the simulation and experimental results. Simulations are conducted to show that our replanner outperforms the existing methods in the solution quality. Real-world experiments are conducted to investigate the online solution speed of our method.

A. Simulation Setup

Simulations were implemented in a MATLAB platform and were executed on an Intel Core i5-4460T CPU with 8.0 GB RAM that runs at 1.90×2 GHz. A $20\mathrm{m} \times 20\mathrm{m}$ workspace

Algorithm 1: Online Trajectory Replanning Method for Automated Parking.

Input: Original trajectory $Traj_{\text{original}}$, current time t_0, old obstacles obs_{old}, and newly detected obstacle obs_{new};
Output: A replanned trajectory;

```
Identify t_1 and t_brake;
If (t_brake < t_0 + T_think), then
  Build a fail-safe trajectory that fully brakes with maximum acceleration from t_0;
  Return;
End if
  Plan an evasive trajectory Traj_evasive(t) via [5];
  Initialize a set ϒ = ∅ for candidate trajectories;
For each i ∈ {1, ..., N_original}, do
  For each j ∈ {1, ..., N_evasive}, do
    Plan a candidate connective trajectory Traj_candidate(t) from the pose Traj_original(t_i^sample) to the pose Traj_evasive(t_j^sample2) by solving (10);
    If footprint(Traj_candidate(t)) ⊄ obs_new ∪ obs_old, then
      Emplace Traj_candidate(t) in ϒ;
    End if
      End for
    End for
    If (Traj_evasive(t) = ∅) ∪ (N_A = 0), then
    Build a fail-safe trajectory that fully brakes with maximum acceleration from t_0 + T_think;
    End if
    Initialize J_cur_best = +∞;
For each Traj_candidate(t) ∈ ϒ, do
  Construct an overall replanned trajectory Traj via (11) and measure its cost J_cur;
  If (J_cur_best > J_cur), then
    Set Traj_replanned(t) ← Traj, J_cur_best ← J_cur;
  End if
end for
Return.
```

is defined as the concerned parking scenario. Our simulations are tested on eight typical cases among the benchmark cases reported in [5]. In each case, the original parking trajectory is planned offline via [2]. A new obstacle is added online with $\mathrm{t}_0 = \gamma_0 \cdot \mathrm{t}_\mathrm{f}$ and $\mathrm{t}_1 = \gamma_1 \cdot \mathrm{t}_\mathrm{f}$, where $\gamma_0 \in [0.1, 0.3]$ and $\gamma_1 \in [0.6, 0.9]$ are uniformly distributed random values. The new obstacle is a polygon with four vertices, the area of which is a random value ranging from 0 to 9.0m^2. The evasive trajectory is planned online via [5], wherein the NLP solver is an open-source version of IPM, namely IPOPT in its 3.13.4 version [28], while the linear solver embedded in IPOPT is set to ma27 [29]. Basic parametric settings for simulations are listed in Table I, while other settings can be found in https://github.com/libai1943/ParkingTrajectoryReplanner.

TABLE I
PARAMETRIC SETTINGS FOR SIMULATIONS

Parameter	Description	Setting
$\mathrm{L_F}$	Front hang length of ego vehicle	0.96 m
$\mathrm{L_W}$	Wheelbase of ego vehicle	2.80 m
$\mathrm{L_R}$	Rear hang length of ego vehicle	0.929 m
$\mathrm{L_B}$	Width of ego vehicle	1.942 m
$\mathrm{a_{max}}$	Upper bound of $\lvert a(t)\rvert$	2.0 m/s^2
$\mathrm{v_{max}}$	Upper bound of $\lvert v(t)\rvert$	3.0 m/s
Φ_{max}	Upper bound of $\lvert \phi(t)\rvert$	0.85 rad
Ω_{max}	Upper bound of $\lvert \omega(t)\rvert$	0.7 rad/s
$\mathrm{s_{buffer}}$	Width of buffer region around obs_{new}	2.0 m
$\mathrm{T_{think}}$	Predefined time to replan trajectories online	1.2 s
$\mathrm{N_{original}}$	Number of sampled candidates as start point of connective trajectory	5
$\mathrm{N_{evasive}}$	Number of sampled candidates as end point of connective trajectory	6
$\Delta\mathrm{T_{start}}$, $\Delta\mathrm{T_{end}}$	Parameters used to determine the lower and upper bounds of an interval, where end point candidates are sampled	$0.05 \cdot t_{\text{end}}$, $0.30 \cdot t_{\text{end}}$

B. On the Efficiency of Our Replanner

This subsection uses a typical benchmark case to show the efficiency of the proposed replanner. The replanned trajectory is depicted in Fig. 3(a), the continuity of which is further reflected by the steering angle and velocity profiles in Fig. 3(b). Recall that a replanned trajectory consists of three segments, Fig. 3(b) clearly shows that no discontinuity exists between adjacent segments. Fig. 3(c) illustrates the footprints of the ego vehicle along the replanned trajectory, which shows that the replanned trajectory does not involve any collision with the obstacles. Fig. 3(d) presents all valid candidate connective trajectories derived by $\mathrm{t}_0 + \mathrm{T}_{\mathrm{think}}$, among which the one that makes the gross cost minimized is selected. Simulation results on two extra cases are depicted in Fig. 4. More results are available at www.bilibili.com/video/BV1HP4y1H7jy.

Particularly in Fig. 4(b), the replanned trajectory is not homotopic with the evasive trajectory. This means that our replanner can alter the homotopy class if a better one is found during the selection of a qualified connective trajectory [30], [31]. This property allows the proposed replanner to improve the global optimality of the solution especially when the hybrid A$*$ algorithm (or other sampling-and-search-based ones that do not guarantee optimality) is involved in the replanning process.

C. On the Superiority of Our Replanner

This subsection investigates the solution quality of the proposed trajectory replanner by comparing it with some existing methods. Table II defines the comparable methods and reports the comparative results.

Table II reflects the algorithm performance w.r.t. solution quality. In this table, Algorithm 2 is about solving the nominal replanning problem directly via numerical optimization [2], which is widely known to run slowly. Although Algorithm 2 has no chance to be an online replanner, it serves as a baseline

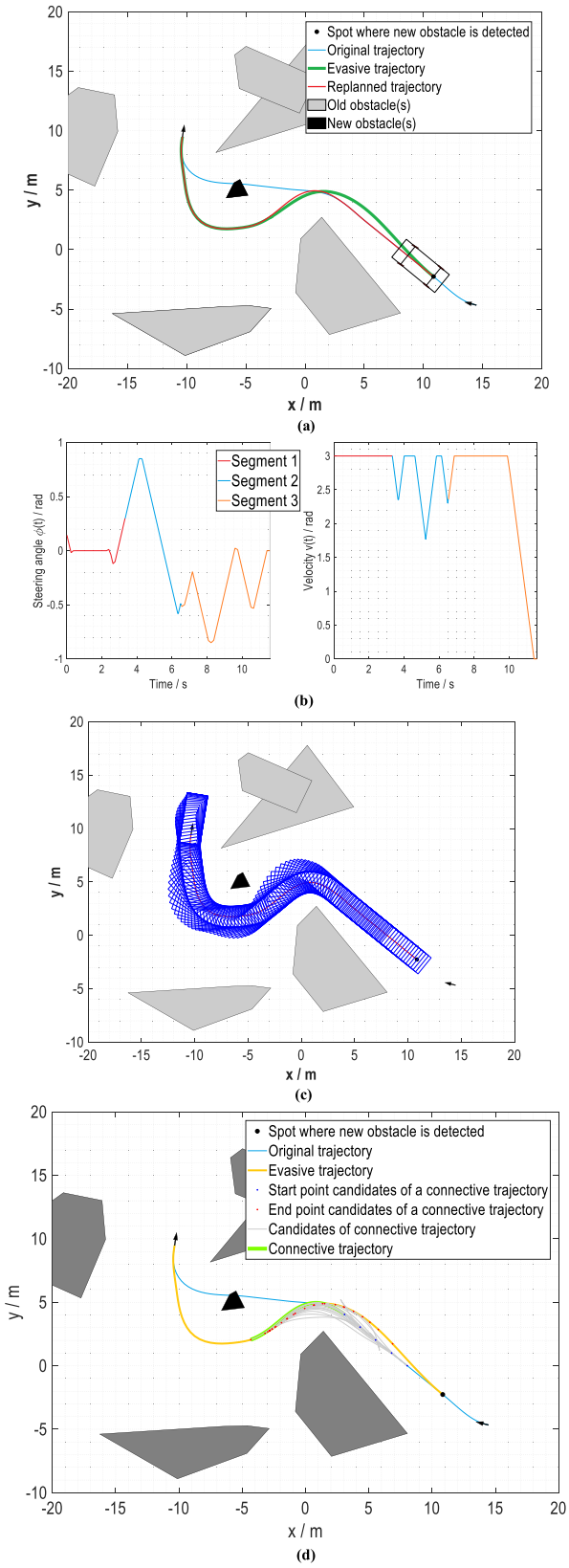

Fig. 3. Trajectory replanning result of a typical simulation case: (a) Comparison among original, evasive, and replanned trajectories; (b) Continuity of state profiles in association with the derived replanned trajectory; (c) Footprints along the replanned trajectory; and (d) Valid candidate connective trajectories.

TABLE II
DEFINITION OF COMPARABLE METHODS AND COMPARATIVE RESULTS

Algorithm ID	Description	Average rate of optimality loss
Alg. 1	The trajectory replanner proposed in this paper	19.72%
Alg. 2	Directly solving (6) via [2] to provide an offline replanned trajectory	-
Alg. 3	Basic online stitching method [20]	45.10%
Alg. 4	Computational stitching method [19]	34.16%

TABLE III
SIMULATION RESULTS ABOUT PARAMETRIC SENSITIVITY

Parametric setting	Average rate of optimality loss
$N_{original} = 1,\ N_{evasive} = 3$	30.04%
$N_{original} = 1,\ N_{evasive} = 6$	28.34%
$N_{original} = 3,\ N_{evasive} = 6$	21.27%
$N_{original} = 5,\ N_{evasive} = 3$	21.52%
$N_{original} = 5,\ N_{evasive} = 6$	20.92%
$N_{original} = 5,\ N_{evasive} = 9$	20.80%
$N_{original} = 5,\ N_{evasive} = 18$	20.48%

approach to evaluate the optimality of other online replanners. In dealing with a specified case, we assume that method X obtains a replanned trajectory with a cost value c_x while Algorithm 2 obtains one with c_2, then the optimality loss rate of method X, denoted as r_x, is defined as

$$r_x \equiv \frac{c_x - c_2}{c_2}. \tag{12}$$

The average optimality loss rate index is calculated based on 300 random cases. Notably, we only focus on the cases that all of the four comparable algorithms succeed for fairness. According to Table II, one can conclude that the computational stitching method (Algorithm 4) is more efficacious to improve the solution quality than the conventional stitching method, while our proposed method outperforms both of them (Algorithms 3 and 4).

D. Parametric Sensitivity of Our Replanner

This subsection investigates the parametric sensitivity of the proposed replanner by changing some critical parameters listed in Table I. In this work, we focus on making changes to $N_{original}$ and $N_{evasive}$. The solution quality is measured by the criterion called the average rate of optimality loss. It is calculated in the same way as introduced in the preceding subsection, but this time, the sampling number grows to 500.

The simulation results reported in Table III indicate that the performance of our proposal is generally stable unless $N_{original}$ is set too small. Setting $N_{original}$ or $N_{evasive}$ excessively large does not have enough benefits in improving the cost function value.

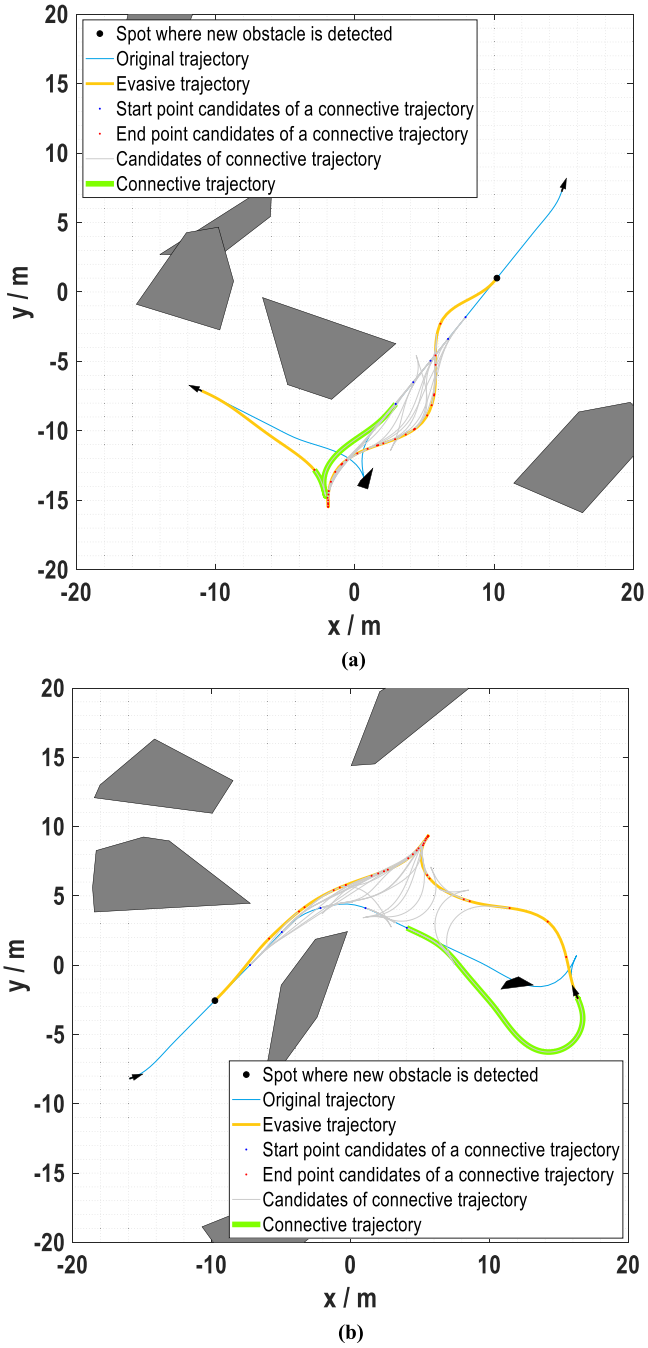

Fig. 4. Trajectory replanning simulation results of another two benchmark cases.

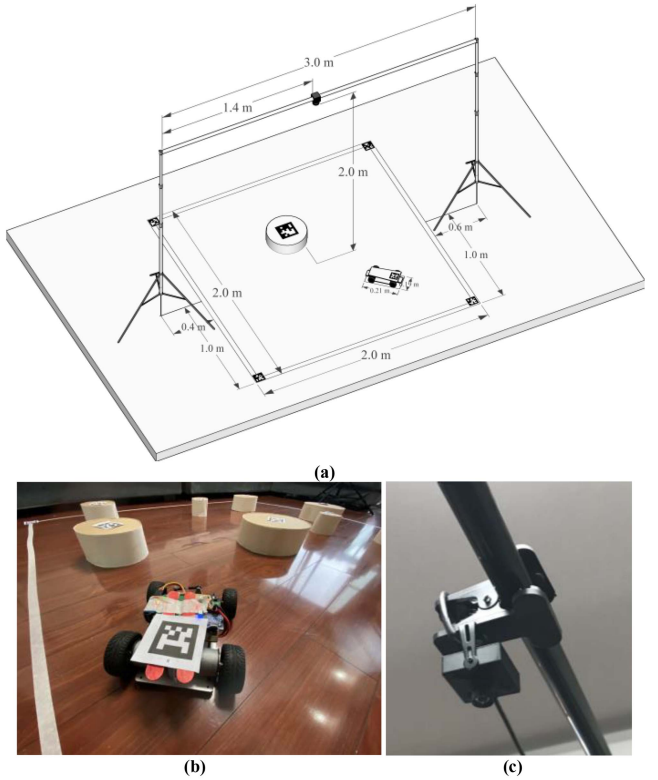

Fig. 5. Real-world experimental scenario and setups: (a) scenario layout; (b) car-like robot; (c) birds-eye-view camera.

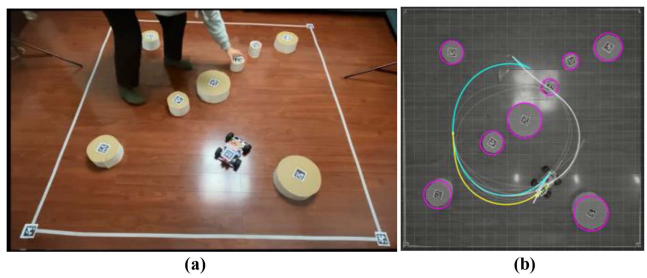

Fig. 6. Typical experimental results: (a) an obstacle addition action during online tracking process; (b) visualization of replanning performance in Rviz of ROS.

E. Experimental Setups and Results

Aside from simulations, real-world experiments are conducted, which show the realizability and runtime of our proposed replanner.

As depicted in Fig. 5(a), a small-sized car-like robot drives in a 2 m $\times$ 2 m indoor scenario occupied by circular obstacles. Those circular obstacles are approximately treated as polygons with many vertices in using our proposed replanner. Fig. 5(b) shows that a unique AprilTag is placed on the top of the car-like robot and each obstacle for visual localization through a birds-eye-view camera equipped on a two-meter-high cross-beam (Fig. 5(c)). The derived localization information is sent to a desktop PC, which is configured as an AMD R7-5800H CPU running at 3.80 $\times$ 8 GHz with 32.0 GB RAM. The source codes of the proposed replanner are written in C++ and implemented in ROS under the Linux (Ubuntu20.04) environment. The camera monitors the environmental changes and vehicle location with a frequency of 30Hz. Once a critical environmental change is detected, the replanning module is triggered in the desktop PC [32], and the resultant replanned trajectory is sent to the car-like robot via ZigBee for closed-loop tracking. In trajectory tracking, a PID controller is used for longitudinal tracking and a pure-pursuit controller tracks laterally. Both controllers work with a frequency of 30Hz.

In using the proposed replanner, we adopt the following critical settings: $N_{\text{original}} = 5$, $N_{\text{evasive}} = 6$, and $T_{\text{think}} = 2.0$s. Averagely, it takes 79.60ms to generate a candidate connective trajectory. Given that eight threads work in parallel to plan those candidates, the gross CPU time consumed for online replanning (including the generation of an evasive trajectory) is 323.00ms on average, which does not exceed the assigned two-second duration for T_{think}. This indicates that the proposed replanner is efficacious. A typical experimental result is depicted in Fig. 6. More results are presented in the supplementary video.

V. Conclusion

This paper has introduced a trajectory replanning method for automated parking. Selecting among candidate connective trajectories greedily is an innovation in our proposal, which has a chance to promote the solution quality with no sacrifice in the solution speed. According to our conducted simulations, the proposed replanner ensures collision avoidance, kinematic feasibility, and parameter insensitivity. According to our real-world experiments, the proposed replanner runs fast enough so that the algorithm design is easy to implement online.

As an interesting phenomenon we found during the simulations, the deployment of a connective trajectory allows that the replanned trajectory is not homotopic with the evasive one, which may inspire future studies to examine how to enhance the global optimality in the community of robotics motion planning.

Acknowledgment

The authors are grateful to the anonymous reviewers and handling editor for their valuable comments and suggestions. The authors sincerely thank Zhuoyang Du for her support in the real-world experiments of this work.

References

[1] D. Thornton, K. Redmill, and B. Coifman, "Automated parking surveys from a LiDAR equipped vehicle," *Transp. Res. Part C, Emerg. Technol.*, vol. 39, pp. 23–35, 2020.

[2] B. Li and Z. Shao, "A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles," *Knowl.-Based Syst.*, vol. 86, pp. 11–20, 2015.

[3] G. Wu *et al.*, "Optimal design and planning for compact automated parking systems," *Eur. J. Oper. Res.*, vol. 273, no. 3, pp. 948–967, 2019.

[4] Y. Lin, L. Li, X. Dai, N. Zheng, and F. Wang, "Master general parking skill via deep learning," in *Proc. IEEE Intell. Veh. Symp.*, 2017, pp. 941–946.

[5] B. Li *et al.*, "Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework," *IEEE Trans. Intell. Transp. Syst.*, to be published, doi: 10.1109/TITS.2021.3109011.

[6] W. Liu, Z. Li, L. Li, and F. Y. Wang, "Parking like a human: A direct trajectory planning solution," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 12, pp. 3388–3397, Dec. 2017.

[7] R. Vieira, E. Argento, and T. Revoredo, "Trajectory planning for car-like robots through curve parametrization and genetic algorithm optimization with applications to autonomous parking," *IEEE Latin Amer. Trans.*, vol. 20, no. 2, pp. 309–316, Feb. 2022.

[8] Y. Dong, Y. Zhong, and J. Hong, "Knowledge-biased sampling-based path planning for automated vehicles parking," *IEEE Access*, vol. 8, pp. 156 818–156 827, 2020.

[9] Y. Tazaki, H. Okuda, and T. Suzuki, "Parking trajectory planning using multiresolution state roadmaps," *IEEE Trans. Intell. Veh.*, vol. 2, no. 4, pp. 298–307, Dec. 2017.

[10] B. Li, K. Wang, and Z. Shao, "Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3263–3274, Nov. 2016.

[11] A. Martin, R. Lattarulo, A. Zubizarreta, J. Perez, and P. Lopez-Garcia, "Trajectory planning for automated buses in parking areas," in *Proc. 25th Int. Conf. Syst. Theory, Control Comput.*, 2021, pp. 688–694.

[12] L. Xiong, J. Gao, Z. Fu, and K. Xiao, "Path planning for automatic parking based on improved hybrid A∗ algorithm," in *Proc. 5th CAA Int. Conf. Veh. Control Intell.*, 2021, pp. 1–5.

[13] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 1879–1884.

[14] X. Li *et al.*, "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles," *Mech. Syst. Signal Process.*, vol. 87, pp. 118–137, 2017.

[15] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robot. Auton. Syst.*, vol. 88, pp. 142–153, 2017.

[16] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[17] B. Li, Y. Ouyang, L. Li, and Y. Zhang, "Autonomous driving on curvy roads without reliance on frenet frame: A cartesian-based trajectory planning method," *IEEE Trans. Intell. Transp. Syst.*, to be published, doi: 10.1109/TITS.2022.3145389.

[18] C. Pek and M. Althoff, "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization," *IEEE Trans. Robot.*, vol. 37, no. 3, pp. 798–814, Jun. 2021.

[19] C. Jang, C. Kim, S. Lee, S. Kim, S. Lee, and M. Sunwoo, "Re-plannable automated parking system with a standalone around view monitor for narrow parking lots," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 2, pp. 777–790, Feb. 2020.

[20] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha — A local, continuous method," in *Proc. IEEE Intell. Veh. Symp.*, 2014, pp. 450–457.

[21] B. Li, T. Acarman, X. Peng, Y. M. Zhang, X. Bian, and Q. Kong, "Maneuver planning for automatic parking with safe travel corridors: A numerical optimal control approach," in *Proc. Eur. Control Conf.*, 2020, pp. 1993–1998.

[22] J. Zhou *et al.*, "Autonomous driving trajectory optimization with dual-loop iterative anchoring path smoothing and piecewise-jerk speed optimization," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 439–446, Apr. 2021.

[23] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *Proc. IEEE Conf. Decis. Control*, 2018, pp. 4327–4332.

[24] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 485–501, 2010.

[25] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, 1990.

[26] A. J. Häusler, A. Saccon, A. P. Aguiar, J. Hauser, and A. M. Pascoal, "Energy-optimal motion planning for multiple robotic vehicles with collision avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 3, pp. 867–883, May 2016.

[27] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004, pp. 46–47.

[28] A. Wächter and L. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[29] HSL, "A collection of fortran codes for large scale scientific computation." [Online]. Available: http://www.hsl.rl.ac.uk/

[30] B. Li, Y. Ouyang, Y. Zhang, T. Acarman, Q. Kong, and Z. Shao, "Optimal cooperative maneuver planning for multiple nonholonomic robots in a tiny environment via adaptive-scaling constrained optimization," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1511–1518, Apr. 2021.

[31] K. Kolur, S. Chintalapudi, B. Boots, and M. Mukadam, "Online motion planning over multiple homotopy classes with Gaussian process inference," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 2358–2364.

[32] Z. Jian, S. Chen, S. Zhang, Y. Chen, and N. Zheng, "Multi-model-based local path planning methodology for autonomous driving: An integrated framework," *IEEE Trans. Intell. Transp. Syst.*, to be published, doi: 10.1109/TITS.2020.3042603.

Bai Li (Member, IEEE) received the B.S. degree from the School of Advanced Engineering, Beihang University, Beijing, China, in 2013, and the Ph.D. degree from the College of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2018. From November 2016 to June 2017, he visited the Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, MI, USA, as a joint training Ph.D. Student. He is currently an Associate Professor with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha, China. From 2018 to 2020, he was with the JDX R&D Center of Automated Driving, JD Inc., China, as an Algorithm Engineer. He has been the first author of 60 journal/conference papers and two books in the community of robotics. His research focuses on the motion planning of automated vehicles. He was the recipient of the International Federation of Automatic Control 2014–2016 Best Journal Paper Prize. Since 2022, he serves as an Associate Editor for the IEEE Transactions on Intelligent Vehicles.

Zhuyan Yin received the B.S. degree in operations research and management from the Applied Technical School, Hunan Institute of Engineering, Xiangtan, China, in 2016. Since 2021, she has been a Research Assistant with Hunan University, Changsha, China. Her research focuses on optimization-based trajectory planning of an intelligent vehicle.

Yakun Ouyang (Student Member, IEEE) received the B.S. degree from the School of Information Engineering, Nanchang University, Nanchang, China, in 2020. He is currently working toward the master's degree with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha, China. His research interests include decision making, trajectory planning, control, and software engineering of an autonomous vehicle system. He was the first-prize recipient of the 2019 National University Students Intelligent Car Race.

Youmin Zhang (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Northwestern Polytechnical University, Xi'an, China, in 1983, 1986, and 1995, respectively. He is currently a Professor with the Department of Mechanical, Industrial, and Aerospace Engineering, and the Concordia Institute of Aerospace Design and Innovation, Concordia University, Montreal, QC, Canada. He has authored eight books, more than 550 journal and conference papers, and book chapters. His main research interests include fault detection and diagnosis, fault-tolerant control, fault-tolerant cooperative control of single and multiple unmanned aerial/space/ground/marine vehicles, smart grids, and applications of unmanned systems to forest fires, power lines, environment, natural resources and disasters monitoring, detection, and protection by combining with remote sensing techniques. He is a Fellow of CSME, a Senior Member of AIAA, President of the International Society of Intelligent Unmanned Systems, and a Member of the Technical Committee for several scientific societies. He has been the Editorial Board Member, Editor-in-Chief, Editor-at-Large, Editor, and an Associate Editor of several international journals, such as the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and *Journal of Intelligent & Robotic Systems*.

Xiang Zhong received the B.S. degree from the Department of Electrical Engineering, Hunan University, Changsha, China, in 1993, and the M.S. degree from the School of Software, Hunan University, in 2014. He is currently an Assistant Professor with the College of Mechanical and Vehicle Engineering, Hunan University, the President of Hunan Intelligent Transportation Industry Association, and Deputy Director of Hunan Provincial Key Laboratory of Big Data Research and Applications. His research interests include intelligent transportation, traffic Big Data analysis, and vehicle road collaboration.

Shiqi Tang received the B.S. degree in mechatronics and control engineering from the Wuhan University of Technology, Wuhan, China, in 2020. He is currently working toward the master's degree with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha, China. His research interests include fail-safe motion planning and control of an autonomous vehicle system.