Guiding-based Nonlinear Learning Predictive Control for Mobile Robots with Safety Guarantees

Yang Lu, Weijia Yao, Xinglong Zhang, Yongqian Xiao, Xin Xu⋆, *IEEE Senior Member*

Abstract— **Model predictive control (MPC) is one of the popular approaches to let a robot accurately follow a desired geometric path while avoiding obstacles that obstruct the path. However, MPC is sometimes ineffective in dealing with nonlinear system dynamics and non-convex constraints associated with moving obstacles, which can deteriorate the real-time computational performance and safety in real-world applications. To overcome these challenges and ensure safety, we propose a two-level approach called Guiding-based nonlinear Learning Predictive Control (G-LPC) for mobile robots. At the guiding level, G-LPC plans a local near-optimal path with a discrete-time composite vector field to guide robots to safely track the desired path obscured by static obstacles. At the safety level, G-LPC, equipped with a game-based barrier function, solves the Hamilton–Jacobi–Bellman (HJB) equation through online receding-horizon reinforcement learning (RHRL) and obtains an optimal control policy. In an environment with densely distributed obstacles, we compare G-LPC with several advanced planning methods, and the results show that G-LPC generates feasible paths quickly while achieving both path tracking and obstacle avoidance. We evaluate G-LPC against advanced safe control methods in the high-fidelity simulator CarSim, and the results demonstrate the superiority of G-LPC in terms of completion time, path length, and average solution time. We also conduct real-world experiments on a Hongqi E-HS3 electric vehicle to validate its effectiveness.**

I. Introduction

The tracking control of desired reference paths for mobile robots is a fundamental task. Most tracking control methods, such as preview control [1] and optimal control [2], [3], drive robots to return to the desired path while minimizing the control costs. They require a given starting and ending point to enable the robot to track the desired path of the two-point connection. Therefore, robots cannot start from an arbitrary point within the configuration space and track the reference path. In addition, these methods alone do not provide safety mechanisms to avoid possibly severe collisions with obstacles.

There are various methods used in robot planning, such as the artificial potential field (APF) method [4], the navigation function method [5], and the dynamic window (DW) method [6]. Few current planning approaches are compatible with tracking control methods since many planning approaches are open-loop while tracking control methods usually require closed-loop feedback; however, the guiding vector field provides a closed-loop feedback planning solution which can guide the robot to achieve path-tracking control and simplify the algorithm design [7], [8]. They usually ensure that the robot's position converges to the desired path from (almost) any point in the configuration space. To guide robots to track the desired reference path and avoid obstacles blocking the robot's path, a composite vector field method was proposed in [9]. Subsequently, Yao et al. [10] constructed a composite vector field by using smooth bump functions to enable path following and obstacle avoidance. Additionally, a switching vector field is designed to address the deadlock issue. These methods [7]–[10] probably cannot directly deal with *optimal control problems characterized by strong nonlinear dynamic constraints*.

Integrated planning and control methods, such as numerical optimization-based [11]–[14] and learning-based approaches [15], [16], [18], have received widespread attention in terms of ensuring the high-mobility robot maneuvers. For one thing, numerical optimization-based methods excel in addressing multi-constraint optimization problems. Li et al. [11] developed a nonlinear model predictive control (MPC) planner to facilitate obstacle avoidance. Recently, several MPC-based techniques have emerged, with the planned trajectory computed at the boundary of the state invariant set. Notably, Zeng et al. [12] proposed a safety-critical MPC method to handle existing uncertainties. Brito et al. [13] proposed a contouring MPC approach with precise obstacle boundary computation. *Under nonlinear dynamics and non-convex constraints, numerical optimization-based methods usually cannot guarantee the quality of solutions and even may not produce a feasible solution.*

For another, adaptive dynamic programming (ADP) and reinforcement learning (RL) are efficient in solving nonlinear optimal control problems. Model-based RL approaches are promising in achieving kinodynamic motion planning [15], [16]. They transform the motion planning problem into a regulation problem, where a reference signal is given to track while the safety constraints are satisfied. *A major challenge is generating a compatible and near-optimal signal quickly*. Also, the challenge of designing stable and reliable barrier functions remains. Zhang et al. [16] proposed a soft potential field approach that handles obstacle constraints using the convexification method from [14]. For online updating receding-horizon RL methods [18], [19], unstable repulsive forces (such as at the inflection points of convex obstacles) can lead to learning failures. Lu et al. [15] introduced exponential barrier functions into the cost function to handle non-convex constraints; however, the use of rule-based switching mechanisms may encounter corner cases.

Inspired by the idea from MPC, receding-horizon reinforcement learning (RHRL) methods employ multiple itera-

Yang Lu, Xinglong Zhang, Yongqian Xiao, and Xin Xu are with National University of Defense Technology, China. (xuxin_mail@263.net). Weijia Yao is with University of Groningen, the Netherlands.

tive learning steps within the prediction horizon to expedite algorithm convergence [16], [18]. Note that the optimization mechanism of RHRL methods fundamentally differs from that of MPC, as the former involves multiple iterative steps within the prediction horizon, while MPC solves the global optimization problem within the prediction horizon. Studies [16], [18] show that RHRL and MPC demonstrate comparable control performance, while RHRL surpasses MPC in some cases. Regarding collision avoidance constraints, MPC methods can calculate optimal trajectories directly, while RHRL methods require passive designs to ensure safety, such as fine-grained obstacle handling for optimal motion.

The above challenges (statements in italics above) on learning efficiency and achieving near-optimal safe control performance motivated us to propose a two-level Guiding-based Learning Predictive Control method called G-LPC for mobile robots with safety guarantees. At the guiding level, we construct a discrete-time composite vector field, aiming to guide a mobile robot to track the desired path occluded by static obstacles. The safety level incorporates a game-based barrier function, and the optimal control from solving the Hamilton–Jacobi–Bellman (HJB) equation is approximated under the framework of online receding-horizon RL. The main contributions are as follows:

1) The proposed G-LPC approach can achieve *near-optimal safety control* for robots with *unknown dynamics*. It achieves MPC-like performance by planning a near-optimal path before tracking. Notably, the approach achieves *higher computational efficiency* and obtains *more reliable solutions* compared to traditional MPC methods in solving nonlinear optimization problems.
2) The proposed discrete-time composite vector field formulation *meets robot dynamic constraints* and it is a new way of *solving the deadlock problem* autonomously. Moreover, the game-based barrier function can deal with *non-convex obstacle constraints*.

II. Preliminaries and Problem Formulation

In this section, we will provide a detailed preliminary for the composite vector field employed at the guiding level, along with a description of achieving optimal control for system (2). Finally, we present the problem formulation.

A. Composite Vector Field

Consider the following ordinary differential equation:

$$\dot{\xi} = \chi(\xi(t))$$

with the initial state $\xi(0) \in \mathbb{R}^2$, where $\chi(\cdot)$ is continuously differentiable concerning ξ, and it is designed to be a guiding vector field for path following [10].

A reference path $\mathcal{P}$ is provided initially and may be occluded by obstacles, and it is defined by

$$\mathcal{P} = \left\{\xi \in \mathbb{R}^2 : \phi(\xi) = 0\right\},$$

where $\phi : \mathbb{R}^2 \to \mathbb{R}$ is twice continuously differentiable. For example, a circle path $\mathcal{P}$ can be described by choosing $\phi(x, y) = x^2 + y^2 - R^2$, where R is the radius of the circle. To avoid collisions, Yao et al. [10] proposed a composite vector field for processing obstacle constraints. It involves a reactive boundary $\mathcal{R}_i^t$ and a repulsive boundary $\mathcal{Q}_i^t$, i.e.,

$$\mathcal{R}_i^t = \left\{\xi \in \mathbb{R}^2 : \varphi_i(\xi, t) = 0\right\}, \mathcal{Q}_i^t = \left\{\xi \in \mathbb{R}^2 : \varphi_i(\xi, t) = c_i\right\},$$

where $\varphi_i : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}$ is twice continuously differentiable, $i \in \mathcal{I} = \{1, 2, \cdots, m\}$, m is the total number of obstacles, and $c_i < 0$. The repulsive boundary $\mathcal{Q}_i^t$ is the boundary that tightly encloses the i-th obstacle at time t and a robot is forbidden to cross this boundary to avoid collision with the obstacle. The reactive boundary $\mathcal{R}_i^t$ is larger than and encloses the repulsive boundary, and its interior is a region where a robot can detect an obstacle and become reactive. We use prescripts ex and in to denote the exterior and interior regions of a boundary, respectively. For example, ${}^{\mathrm{ex}}\mathcal{Q}$ represents the exterior region of the repulsive boundary. An example of moving circular reactive and repulsive boundaries can be characterized by choosing $\varphi_i(x, y; t) = (x - t)^2 + y^2 - R^2$ and letting $|c_i| < R$. In this case, the reactive and repulsive boundaries are large and small concentric circles moving along the x-axis, respectively.

We denote the *path-following vector field* by $\chi_{\mathcal{P}}$ and the *repulsive vector field* by $\chi_{\mathcal{R}_i}$, and they are defined below:

$$\chi_{\mathcal{P}}(\xi) = \gamma_0 E \nabla\phi(\xi) - k_p \phi(\xi) \nabla\phi(\xi),$$
$$\chi_{\mathcal{R}_i}(\xi) = \gamma_i E \nabla\varphi_i(\xi) - k_{r_i} \varphi_i(\xi) \nabla\varphi_i(\xi), \ i \in \mathcal{I},$$

where $E = \left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right]$ is the rotation matrix of 90°, $\gamma_i \in \{1, -1\}$, $i \in \mathcal{I} \cup \{0\}$, determines the moving direction (clockwise or counterclockwise), and k_p, k_{r_i} are positive coefficients. The composite vector field is as follows [10]:

$$\chi(\xi) = \left(\prod_{i \in \mathcal{I}} \sqcup_{\mathcal{Q}_i}(\xi)\right) \hat{\chi}_{\mathcal{P}}(\xi) + \sum_{i \in \mathcal{I}} \left(\sqcap_{\mathcal{R}_i}(\xi) \hat{\chi}_{\mathcal{R}_i}(\xi)\right), \quad (1)$$

where $\hat{(\cdot)}$ is the normalization notation (i.e., for a nonzero vector $v \in \mathbb{R}^n$, $\hat{v} = v / \|v\|$), $\sqcup_{\mathcal{Q}}(\xi) = \frac{f_1(\xi)}{f_1(\xi) + f_2(\xi)}$, $\sqcap_{\mathcal{R}}(\xi) = \frac{f_2(\xi)}{f_1(\xi) + f_2(\xi)}$ are smooth *bump functions*, where $f_1(\xi) = 0$ if $\varphi(\xi) \le c$ and $f_1(\xi) = \exp\left(\frac{l_1}{c - \varphi(\xi)}\right)$ if $\varphi(\xi) > c$, $f_2(\xi) = \exp\left(\frac{l_2}{\varphi(\xi)}\right)$ if $\varphi(\xi) < 0$ and $f_2(\xi) = 0$ if $\varphi(\xi) \ge 0$, and $l_1, l_2 > 0$ are coefficients for changing the decaying or increasing rate. Note that for simplicity, the subscripts i of related symbols have been omitted above. These smooth bump functions blend parts of different vector fields and create a composite vector field for both path following and collision avoidance; for more details, see [10].

The *singular sets* of $\chi_{\mathcal{P}}$ and $\chi_{\mathcal{R}_i}$, denoted by $\mathcal{C}_{\mathcal{P}}$ and $\mathcal{C}_{\mathcal{R}_i}$, respectively, are defined below:

$$\mathcal{C}_{\mathcal{P}} = \left\{\xi \in \mathbb{R}^2 : \chi_{\mathcal{P}}(\xi) = 0\right\} = \left\{\xi \in \mathbb{R}^2 : \nabla\phi(\xi) = 0\right\},$$
$$\mathcal{C}_{\mathcal{R}_i} = \left\{\xi \in \mathbb{R}^2 : \chi_{\mathcal{R}_i}(\xi) = 0\right\} = \left\{\xi \in \mathbb{R}^2 : \nabla\varphi_i(\xi) = 0\right\}.$$

The elements of singular sets are called *singular points*, where vector fields vanish. Due to the presence of singular points, special designs are required to solve the deadlock problem. When employing the guiding vector field as high-level guiding signals, neglecting the kinodynamic constraints usually deteriorates the control performance.

B. Problem Formulation

Consider the following continuous-time nonlinear system

$$\dot{x} = f(x, u), \tag{2}$$

where $x \in \mathbb{R}^n$ denotes the system state, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the system transition function, $u \in \Omega_u \subset \mathbb{R}^m$ denotes the control input, and Ω_u is the control constraint set. Note that the explicit dependence on time is dropped unless needed for clarity. We assume that $f(\cdot)$ is locally Lipschitz continuous.

1) Composite Vector Field with Kinodynamic Constraints: The composite vector field acts as a local path planner and should meet the additional requirement of kinodynamic constraints, leading to the problem of Vector-Field-guided Path Planning with Kinodynamic Constraint (VF-PPKC).

Remark 1: The term "kinodynamic constraint" refers to the requirement that a robot will not collide with obstacles at different speeds. To address this issue, we have transformed it into the fulfillment of the maximum curvature.

Definition 1: (VF-PPKC) Design a continuously differentiable vector field $\chi : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}^2$ for $\xi(t) = \chi(t, \xi(t))$ such that:

1) It enables path following and collision avoidance, and the path-following error is bounded, and no deadlocks exist.
2) Given the robot's velocities v_x, v_y, and the maximum acceleration $a_{\max}$, it holds that $(v_x^2 + v_y^2)\kappa(t_2) \le a_{\max}$, where $\kappa(t_2) = (1 + \dot{\xi}(t_2)^2)^{3/2} / |\ddot{\xi}(t_2)|$ is the curvature of the trajectory at t_2.

A guiding vector field $\chi : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}^2$ is designed to generate a twice continuously differentiable reference path $\Xi = \int_0^\infty \chi(\xi(t))\mathrm{d}t$. Subsequently, we employ a learning-based predictive control approach to track the reference path and avoid static and dynamic obstacles at the same time.

2) Optimal Control with Infinite-horizon Cost: Given the nonlinear system (2) and the reference path, we can obtain its discrete-time model, i.e.,

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \tag{3}$$

where $\mathbf{x}_k = x_k - x_k^r$ is the error state, $x_k^r \in \Xi$ is the reference state, and $\mathbf{u}_k = u_k$ is the control input. We hope to utilize an optimal controller to follow the reference path Ξ and satisfy the following conditions: 1) Starts at x_0 and tracks the reference path Ξ. 2) Avoids collisions with all obstacles $\mathcal{B}_1, \dots, \mathcal{B}_q \subseteq \mathcal{W} \subseteq \mathbb{R}^2$, where $\mathcal{W}$ denotes the workspace. We define the value function as the cumulative discounted sum of infinite-horizon costs:

$$V_\infty(\mathbf{x}_k) = \textstyle\sum_{\tau=k}^{\infty} \gamma^{\tau-k} L(\mathbf{x}_\tau, \mathbf{u}_\tau), \tag{4}$$

where $0 < \gamma \le 1$, $L(\mathbf{x}_\tau, \mathbf{u}_\tau) = \mathbf{x}_\tau^\top Q \mathbf{x}_\tau + \mathbf{u}_\tau^\top R \mathbf{u}_\tau$ is the cost function, $Q \succeq 0 \in \mathbb{R}^{n \times n}$ is a positive semi-definite matrix, and $R \succ 0 \in \mathbb{R}^{m \times m}$ is a positive definite matrix.

III. Guiding-based Receding-horizon Reinforcement Learning with Safety Guarantees

To guarantee the safety of robots while they move, it's essential to generate local trajectories. This requires designing a guiding vector field that considers dynamic limitations and excludes the deadlock problem (i.e., singular points). Additionally, for local path planning, we have to factor in the uncertainty that arises from moving obstacles. To tackle this problem, we develop an online RHRL approach that employs a game-based barrier function. The overall framework is depicted in Fig. 1.

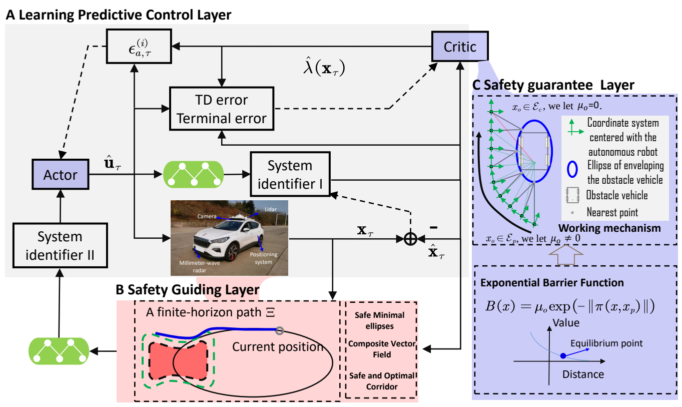
A Learning Predictive Control Layer
$\epsilon_{a,\tau}^{(i)}$
Critic
$\hat{\lambda}(\mathbf{x}_\tau)$
TD error
Terminal error
Actor
$\hat{\mathbf{u}}_\tau$
System identifier I
System identifier II
Camera
Lidar
Millimeter-wave radar
Positioning system
$\mathbf{x}_\tau$
$-$
$\hat{\mathbf{x}}_\tau$
B Safety Guiding Layer
A finite-horizon path Ξ
Current position
Safe Minimal ellipses
Composite Vector Field
Safe and Optimal Corridor
C Safety guarantee Layer
$x_o \in \mathcal{E}_c$, we let μ_o=0.
Coordinate system centered with the autonomous robot
Ellipse of enveloping the obstacle vehicle
Obstacle vehicle
Nearest point
Working mechanism
$x_o \in \mathcal{E}_p$, we let $\mu_o \neq 0$
Exponential Barrier Function
$B(x) = \mu_o \exp(-\|\pi(x, x_p)\|)$
Value
Equilibrium point
Distance

Fig. 1. The overall framework of the G-LPC algorithm.

A. Discrete-time Kinodynamic Composite Vector Field

1) Composite Vector Field with Kinodynamic Constraints: To accomplish the second control objective in Definition 1, we design the following kinodynamic composite vector field:

$$\begin{aligned}\chi_c(\xi) = &\left(\prod_{i \in \mathcal{I}} \sqcup_{\mathcal{Q}_i}(\xi)(1 - \mathcal{S}(\xi)) \right) \hat{\chi}_{\mathcal{P}}(\xi) \\ &+ \sum_{i \in \mathcal{I} \backslash \mathcal{I}'} \left(\sqcap_{\mathcal{R}_i}(\xi) \hat{\chi}_{\mathcal{R}_i}(\xi) \right) + \sum_{i \in \mathcal{I}'} \left(\mathcal{S}(\xi) \hat{\chi}_{\mathcal{R}_i}(\xi) \right),\end{aligned} \tag{5}$$

where $\mathcal{I}'$ is a set containing the index numbers of added virtual obstacles. The path generated by the original composite vector field in (1) would often require the robot to make large turns within a limited distance for collision avoidance. The role of *virtual obstacles* here is to proactively modify the vector field such that the curvature of the robot trajectory is less than the maximum allowable value as the robot enters the region $\mathcal{M} = {}^{\mathrm{ex}}\mathcal{Q} \cap {}^{\mathrm{in}}\mathcal{R}$, thereby satisfying the dynamic constraints. For example, the contour of a virtual obstacle is described by $\varphi_i(\xi) = 0, i \in \mathcal{I}'$ in Fig. 2 and its repulsive boundary is described by $\varphi_i(\xi) = c_i, i \in \mathcal{I}'$. When $\xi = (x, y) \in \mathbb{R}^2$ enters the reactive boundary, it will be attracted towards the boundary $\varphi_i(\xi) = c_i, i \in \mathcal{I}'$. This provides a direction change before ξ enters the reactive boundary $\varphi_i(\xi) = 0, i \in \mathcal{I} \backslash \mathcal{I}'$, and the virtual obstacle will not take effect after ξ enters the boundary. Based on the above analyses, $S(\xi)$ is designed to be

$$\mathcal{S}(\xi) = \begin{cases} 1 & \text{if } \varphi_i(\xi) \le 0 \text{ for } i \in \mathcal{I}' \text{ and } \varphi_i(\xi) > 0 \text{ for } i \in \mathcal{I} \backslash \mathcal{I}' \\ 0 & \text{otherwise.} \end{cases}$$

2) Analysis of the Composite Vector Field: The definition of the composite vector field (5) gives an idea of meeting the kinodynamic constraint. However, it may result in cyclic behavior within the region $\mathcal{M} = {}^{\mathrm{ex}}\mathcal{Q} \cap {}^{\mathrm{in}}\mathcal{R}$. When the boundary described by $\varphi_i(\xi) = 0$ does not completely encompass the region $\mathcal{P} \cap \mathcal{M}$, the cyclic behavior can be avoided within the mixed region when exiting $\mathcal{M}$. This enables continued tracking of the reference path after avoiding

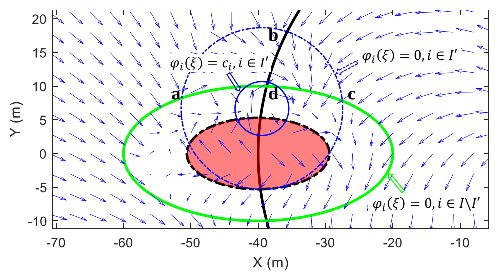

Fig. 2. An illustrative example of the kinodynamic composite vector field.

obstacles. By redesigning the characterizing functions ϕ, φ, the bump functions, and the coefficients c_i, the first condition of Definition 1 can be met in practice. The second condition can be satisfied by selecting proper positive coefficient $k_{r_i}, i \in \mathcal{I}'$. These two conditions are satisfied in simulations and experiments in this paper.

3) Discrete-time Guiding Vector Field: With $\chi_c(\xi)$, the desired path is computed with its discrete-time form, i.e.,

$$\xi_{k+1} = \xi_k + \beta\chi_c(\xi_k), \xi_0 = (x_0, y_0), \tag{6}$$

where $\xi_0 \in \mathbb{R}^2$ is the initial robot position and β is the step length. It is advisable to precompute the guiding vector field of the mesh grid map. This makes it possible to quickly identify the nearest vector $\chi_c(\xi_k)$ based on the current position ξ_k. Finally, we can obtain an N-step planned path. Note that vectors with nearly-zero norms are not selected as the current vector to avoid suffering singularity issues. *This allows us to plan guiding paths directly without causing the deadlock problem from singular points.*

In fact, the new reference path Ξ' sequentially connected by points in $\{\xi_0, \xi_1, \cdots, \xi_N\}$ are not smooth in most cases. Thus ensuring optimal control performance remains a challenge. We adopt the moving average method to smooth the path, i.e.,

$$\xi_k \leftarrow \frac{1}{T}(\xi_k + \xi_{k-1} + \ldots + \xi_{k-T+2} + \xi_{k-T+1}), \tag{7}$$

where $T \leq N$ is the size of the sliding window.

B. Online Receding-horizon Reinforcement Learning with Safety Guarantees

Motivated by the suicidal pedestrian differential game [20], we propose a barrier function design, and it is incorporated into the cost function to establish a constraint-free optimization problem, which is then solved by online RHRL.

1) Cost Function Reformulation: Consistent with [15], the barrier function used for safety is designed as

$$B(x) = \mu_o \exp\left(-\|\pi(x, x_p)\|\right), \tag{8}$$

where $\exp(\cdot)$ denotes the exponential function, μ_o is the penalty coefficient, $x_p \in \mathbb{R}^2$ is the nearest boundary coordinate of the nearest obstacle, and $\pi : \mathbb{R}^n \times \mathbb{R}^2 \to \mathbb{R}$ maps x and x_p to the distance error. According to the properties of this designed barrier function [15], the robot can keep a safe distance l_{safe} from the obstacle. However, after ensuring safety and obstacle avoidance, the effect of the barrier function should be eliminated in order to track the reference path. Therefore, a safe switching scheme is required to be designed.

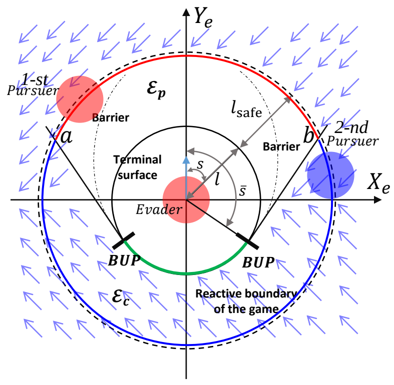

Fig. 3. Pursuer-Evader based switching barrier function: the black dashed line denotes the reactive boundary of the barrier function, $\mathcal{E}_p$ denotes the area for using the barrier function, and $\mathcal{E}_c$ denotes the area for removing it.

In a suicidal pursuit-evasion game task, the evader must take optimal actions to ensure safety. To facilitate the design, in Fig. 3, a Cartesian coordinate system is established with the direction of the evader's velocity as the y-axis. The terminal circle is divided into the *usable part* (the black line) and the *nonusable part* (the green line), separated by the boundary of the usable part (BUP). For a differential game of pursuit and evasion involving an evader and an agile pursuer [20], the solution of BUP is determined by $\bar{s} = \arccos(-v_p/v_e)$, $\bar{s} \in (\pi/2, \pi]$, where v_p and v_e represent the speeds of the pursuer and the evader, respectively. The *barrier* implies that the optimal play by both agents starting from any point will generate a path that does not penetrate this surface. The BUP connects to the barrier and meets the terminal surface at the BUP tangentially. Therefore, as shown in Fig. 3, we consider the plane formed by points a, b, and the points at BUP as a conservative barrier to facilitate safety.

Fig. 3 shows that the coordinate plane is divided into two regions: $\mathcal{E}_p = \mathcal{E}_{p,1} \cap \mathcal{E}_{p,2}$ and $\mathcal{E}_c = \mathcal{E} \setminus \mathcal{E}_p$, where $\mathcal{E}_{p,1}$ and $\mathcal{E}_{p,2}$ are defined below:

$$\begin{aligned}
\mathcal{E}_{p,1} &= \left\{\zeta = [X_e, Y_e]^\top \in \mathbb{R}^2 : \|\zeta\| \leq l + l_{\text{safe}}\right\} \\
\mathcal{E}_{p,2} &= \mathcal{H}_1 \cup \mathcal{H}_2 \cup \mathcal{H}_3,
\end{aligned}$$

with

$$\begin{aligned}
\mathcal{H}_1 &= \{[X_e, Y_e]^\top : X_e \leq -l\sin\bar{s},\ Y_e \geq \tan(\bar{s})X_e + l/\cos\bar{s}\} \\
\mathcal{H}_2 &= \{[X_e, Y_e]^\top : X_e \geq l\sin\bar{s},\ Y_e \geq -\tan(\bar{s})X_e + l/\cos\bar{s}\} \\
\mathcal{H}_3 &= \{[X_e, Y_e]^\top : X_e \in (-l\sin\bar{s}, l\sin\bar{s}),\ Y_e^2 + X_e^2 \leq l^2\}.
\end{aligned}$$

We can combine such findings with the previously designed exponential barrier functions. Specifically, in the area $\mathcal{E}_p$, we assign the penalty coefficient μ_o in (8) a non-zero value to keep a safe distance from the obstacle, while in the region $\mathcal{E}_c$, we let $\mu_o = 0$ to track the planned preliminary path. Finally, the cost function at the τ instant is rewritten as

$$L(\mathbf{x}_\tau, \mathbf{u}_\tau) = \mathbf{x}_\tau^\top Q\mathbf{x}_\tau + \mathbf{u}_\tau^\top R\mathbf{u}_\tau + B(x_\tau). \tag{9}$$

2) Online Receding-horizon Reinforcement Learning with a Game-based Barrier Function Design: We have been inspired by the concept of receding-horizon from MPC, as discussed in [21]. To elaborate, we divide $V_\infty(\mathbf{x}_k)$ into multiple sub-problems within the prediction horizon $[k, k+N]$ to compute an admissible control $\mathbf{u}_k$ by optimizing the following equation:

$$V(\mathbf{x}_\tau) = \begin{cases} F(\hat{\mathbf{x}}_{k+N}) + \sum_{j=\tau}^{k+N-1} \gamma^{j-\tau} L(\mathbf{x}_j, \mathbf{u}_j), & \tau \in [k, k+N-1] \\ F(\hat{\mathbf{x}}_{k+N}), & \tau = k+N \end{cases}$$

where $\gamma \in (0,1]$ is the discount factor and $F(\hat{\mathbf{x}}_{k+N})$ is the terminal cost function, defined by $F(\hat{\mathbf{x}}_{k+N}) = \hat{\mathbf{x}}_{k+N}^\top P \hat{\mathbf{x}}_{k+N}$, where $P \succ 0 \in \mathbb{R}^{m\times m}$ is the terminal penalty matrix.

We reformulate the optimal value function as follows:

$$\lambda^*(\mathbf{x}_\tau) = \partial V^*(\mathbf{x}_\tau)/\partial \mathbf{x}_\tau. \tag{10}$$

Therefore, the optimization objective of the critic network can be rewritten as

$$\lambda^*(\mathbf{x}_\tau) = \begin{cases} 2Q\mathbf{x}_\tau + \gamma \dfrac{\partial \hat{\mathbf{x}}_{\tau+1}}{\partial \mathbf{x}_\tau}\lambda^*(\mathbf{x}_\tau) + \dfrac{\partial B(\mathbf{x}_\tau)}{\partial \mathbf{x}_\tau}, & \tau \in [k, k+N-1] \\ 2P\hat{\mathbf{x}}_{k+N} + \dfrac{\partial B(\hat{\mathbf{x}}_{k+N})}{\partial \hat{\mathbf{x}}_{k+N}}, & \tau = k+N \end{cases}$$

With the optimal value function $\lambda^*(\mathbf{x}_\tau)$, one can compute the optimal control by setting $\partial\lambda^*(\mathbf{x}_\tau)/\partial\mathbf{u}_\tau = 0$, i.e.,

$$\mathbf{u}_\tau^* = \beta \tanh(-\frac{1}{2}(\beta R)^{-1}(\hat{g}(\mathbf{x}_\tau))^\top \lambda^*(\hat{\mathbf{x}}_{\tau+1})), \tag{11}$$

where $\tanh(\cdot)$ is the hyperbolic tangent function. The equation above illustrates the fact that the critic network determines the update of the actor network. In RL, the optimal control $\mathbf{u}_\tau^*$ and the optimal value function $\lambda^*(\mathbf{x}_\tau)$ are approximated by policy evaluation and policy improvement. They are estimated by

$$\hat{\mathbf{u}}_\tau = \mathcal{W}_{a,\tau}^\top \mathcal{K}(\mathbf{x}_\tau), \tag{12}$$

$$\hat{\lambda}(\mathbf{x}_\tau) = \mathcal{W}_{c,\tau}^\top \mathcal{K}(\mathbf{x}_\tau), \tag{13}$$

where $\mathcal{W}_a \in \mathbb{R}^{n_\mathcal{K}\times m}$ and $\mathcal{W}_c \in \mathbb{R}^{n_\mathcal{K}\times n}$ are the weights of the actor and critic networks, respectively, and $n_\mathcal{K}$ denotes the dimension of the dictionary, which is selected by approximate linear dependence [22], $\mathcal{K}(\mathbf{x}_\tau)$ is the basis function.

At the i-th iteration, the critic network aims at minimizing the temporal difference (TD) error ϵ_c, which is defined by $\epsilon_{c,\tau}^{(i)} = \frac{1}{2}\|\lambda^{(i)}(\mathbf{x}_\tau) - \hat{\lambda}^{(i)}(\mathbf{x}_\tau)\|^2$, while the actor network minimizes the estimation error ϵ_a, which is defined by $\epsilon_{a,\tau}^{(i)} = \frac{1}{2}\|\mathbf{u}_\tau^{(i)} - \hat{\mathbf{u}}_\tau^{(i)}\|^2$. Therefore, one can derive the following update rules for the critic network and the actor network, respectively:

$$\mathcal{W}_{c,\tau+1}^{(i)} = \mathcal{W}_{c,\tau}^{(i)} - \eta_c \partial\epsilon_{c,\tau}^{(i)}/\partial\mathcal{W}_{c,\tau}^{(i)} \tag{14}$$

$$\mathcal{W}_{a,\tau+1}^{(i)} = \mathcal{W}_{a,\tau}^{(i)} - \eta_a \partial\epsilon_{a,\tau}^{(i)}/\partial\mathcal{W}_{a,\tau}^{(i)}, \tag{15}$$

where $\eta_c, \eta_a > 0$ are the step coefficients of the gradients.

IV. Simulation and Experimental Results

The algorithms were deployed on a laptop computer running Windows 11 with an Intel Core i7-11800H @2.30GHZ CPU and a GeForce RTX 3080 Laptop GPU. In the testing scenarios, we compare G-LPC with several advanced approaches to demonstrate its superiority. To further verify the effectiveness, we also tested it on an actual vehicle of Hongqi EHS-3, shown in the central image of Fig. 1. The detailed algorithm pseudo-code is illustrated in Algorithm 1.

Algorithm 1: G-LPC algorithm

```
   Input: Coordinate set of obstacles
          $\mathcal{O} = \{o_1, \cdots, o_N\}$, velocity set of obstacles:
          $\{v_1, \cdots, v_N\}$, vehicle veloicty $v_e$ and its
          heading angle $\psi$.
   Output: The optimal control sequence $\{\mathbf{u}_1^*, \mathbf{u}_2^*, \cdots\}$.
1  Initialize $\hat{W}_c, \hat{W}_a$;
2  while not reaching the destination do
3      Get the current state of the system $x_0$.
4      if reaching the end of the guiding trajectory then
5          Plan a preliminary trajectory with (5) and (7).
6      end
7      Determine the penalty coefficient $\mu_o$.
8      for $i = 1, \cdots, i_{max}$ do
           // Prediction Horizon
9          for $\tau = k, \cdots, k+N$ do
10             Compute $\hat{\mathbf{u}}_\tau$ and the cost $L(\mathbf{x}_\tau, \mathbf{u}_\tau)$.
11             Compute the TD errors $\epsilon_{c,\tau}^{(i)}$ and $\epsilon_{a,\tau}^{(i)}$.
12             Update the weights $\mathcal{W}_{c,\tau}^{(i)}$ and $\mathcal{W}_{a,\tau}^{(i)}$.
13             Apply $\hat{\mathbf{u}}_\tau$ to the system (3).
14         end
15         if the weights are converged then
16             Obtain the current control policy, and exit.
17         else
18             $i = i + 1$.
19         end
20     end
       // Control Horizon
21     Compute the control $\hat{\mathbf{u}}_\tau$ and apply it to the
        vehicle.
22 end
```

A. Preliminary Path Planning in Obstacle-dense Environment

To evaluate the effectiveness of the G-LPC approach, we compare it with advanced planning approaches in a scenario containing static obstacles; see Fig. 4. We compare G-LPC with the commonly used planning approaches, such as Timed Elastic Bands (TEB) [23] and OBTPAP [24]. In these tests, we use the analytical kinematic vehicle model with the state variable $[X, Y, \psi]$, representing the X-, Y-coordinates and the heading, respectively. All the approaches are performed in MATLAB 2023a. The implementation of TEB is based on its custom functions, while the optimization problem of OBTPAP is solved by IPOPT solver [25].

1) Evaluation metrics: To assess trajectories generated by various planning methods, we consider several performance metrics. The first metric is the path length from the initial point to the destination, denoted as Length; i.e., $\mathrm{Length} = \sum_{k=1}^{N-1} \|(X_{k+1}, Y_{k+1}) - (X_k, Y_k)\|$. The second metric is the

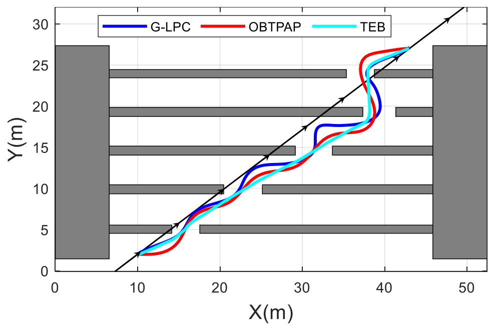

Fig. 4. The planned paths of different comparison methods.

mean lateral cost J_{ML}, defined by $J_{\mathrm{ML}} = \frac{1}{N}\sum_{k=1}^{N} J_{k,\mathrm{Lat}}$, where $J_{\mathrm{Lat}} = \|e_y\|^2$ and e_y is the lateral distance error. The third metric is the average CPU time required to complete the path planning process.

TABLE I
PERFORMANCE EVALUATIONS OF FIG. 4

Quantitative Metrics	G-LPC	TEB	OBTPAP
Length (m)	48.43	**44.32**	48.44
J_{ML}	**3.82**	4.08	4.77
Average CPU time (s)	**0.54**	5.08	8.09

2) Analyses of the results: As observed from TABLE I, the path generated by TEB is the shortest. Given that G-LPC is essentially a tracking-avoidance approach, it yields the lowest mean lateral cost J_{ML} among all approaches, albeit contributing to slightly longer planned trajectories. In terms of computational efficiency, G-LPC demonstrates the lowest average CPU time, measured at 0.54 s, which is significantly lower than those for TEB and OBTRAP.

B. Local Collision Avoidance Tests using Nonlinear CarSim Vehicle Dynamics

Tracking a desired path where static and moving obstacles are placed is a fundamental task. To demonstrate the superiority of G-LPC, we compare it with MPC-CBF [12] and LMPCC [13] under different metrics in MATLAB. They are solved by the IPOPT solver [25] under the interface of CasADi [26].

1) Simulation settings: Considering the unknown system dynamics, an idea is to utilize neural networks to obtain real dynamic characteristics [21], described by a discrete-time model with state $\mathbf{x} = [e_y, \dot{e}_y, e_\psi, \dot{e}_\psi]^\top \in \mathbb{R}^4$ and steering control $\delta_f \in \mathbb{R}$. Comparison approaches, including MPC-CBF and LMPCC, consider the analytical form of nonlinear vehicle dynamics in [15] including the state variable $[v_x, v_y, \psi, \dot{\psi}, x, y]$ and the control variable $[a_x, \delta_f]$.

2) Evaluation metrics: The desired longitudinal velocities of all the methods are set as 25 km/h. To evaluate the G-LPC approach, we compare it with other kinodynamic motion planning algorithms under several metrics, which are Aver. S.T. (average solution time of each time step), J_{Lat} (cost of the lateral error), J_{Heading} (cost of the heading error), and J_{Con} (control cost). We use a weighted average over all metrics to evaluate the overall performance, which is:

$$J_{\mathrm{MC}} = \frac{1}{N}\sum_{k=1}^{N}(Q_1 J_{k,\mathrm{Lat}} + Q_2 J_{k,\mathrm{Heading}} + J_{k,\mathrm{Con}}),$$

where $J_{\mathrm{Lat}} = \|e_y\|$, $J_{\mathrm{Heading}} = \|e_\psi\|$, $J_{\mathrm{Con}} = \mathbf{u}^\top R\mathbf{u}$ is the control cost, and N is the waypoint number of the planned path. Another two quantitative metrics are the overall path length and the time taken by different methods to complete the task is denoted by CT.

3) Analyses of the results: The testing scenario requires the vehicle to track a reference path (black line in Fig. 5) while avoiding collisions with static and moving obstacles.

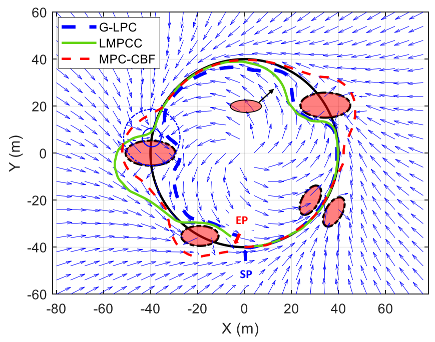

Fig. 5. Simulation results of avoiding static and moving obstacles for three methods. The red arrow denotes the moving direction of the dynamic obstacle, and the black thin line is the desired path. Labels 'EP' and 'SP' denote the endpoint and start point, respectively.

TABLE II
PERFORMANCE EVALUATIONS OF FIG. 5

Quantitative Metrics	G-LPC	MPC-CBF	LMPCC
J_{MC}	**62.12**	69.34	80.79
CT (s)	**34.05**	40.00	37.55
Length (m)	**239.54**	278.27	262.83
Aver. S.T. (s)	**0.05**	0.09	0.10

As observed from TABLE II, the computational cost of G-LPC is lower than that of MPC-CBF and LMPCC, and it generates shorter paths. Since we set the desired velocity to be 25 km/h, G-LPC achieves the least completion time due to its shorter path. As G-LPC employs a neural network to identify the system model and optimizes the nonlinear optimal motion planning problem based on RHRL, the computational time is less. However, the MPC-based methods require online solving of nonlinear optimization problems, resulting in a significant computational burden.

C. Real-World Experiments

To further substantiate the algorithm's efficacy, real-world vehicular experiments were conducted on the Hongqi E-HS3 platform[1]. The platform can be seen from the center of Fig. 1.

The experimental platform operates at an expected velocity of 2 m/s, and it can be observed from Fig. 6 that the vehicle is able to smoothly navigate around obstacles without collisions. This is primarily attributed to the G-LPC method's capability to generate locally feasible paths satisfying dynamic constraints and its online learning mechanism.

[1]https://github.com/XiaoLuYang96/GLPC/blob/main/GLPC.mp4

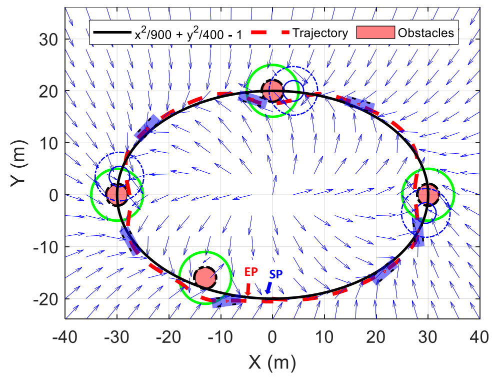

Fig. 6. The experimental results of the deployment of G-LPC on a real-world platform: As illustrated in the figure, reference path $\phi = X^2/900 + Y^2/400 - 21/20$ is obstructed by four static obstacles. Labels 'EP' and 'SP' denote the endpoint and start point, respectively.

V. Conclusion

This paper presents the Guiding-based Learning Predictive Control approach (G-LPC) for mobile robots with safety guarantees, which offers a framework for autonomous planning and tracking control of robots. For one thing, G-LPC incorporates dynamic constraints into the discrete-time guiding vector field for planning local paths and thus addresses safety and deadlock issues during high-mobility robot maneuvers. This is a notable improvement over the existing composite vector fields. For another, G-LPC considers the safety problem between the pursuer and evader; thus, a game-based barrier function is proposed, and a nonlinear optimal control problem is optimized and solved within the receding-horizon RL framework. Extensive simulation and real-world experiments validate the merits of the proposed G-LPC approach.

References

[1] J. M. Snider *et al.*, "Automatic steering methods for autonomous automobile path tracking," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

[2] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[3] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 566–580, 2007.

[4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[5] "Robot navigation functions on manifolds with boundary," *Advances in Applied Mathematics*, vol. 11, no. 4, pp. 412–442, 1990.

[6] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[7] Y. A. Kapitanyuk, A. V. Proskurnikov, and M. Cao, "A guiding vector-field algorithm for path-following control of nonholonomic mobile robots," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1372–1385, 2018.

[8] W. Yao, Y. A. Kapitanyuk, and M. Cao, "Robotic path following in 3d using a guiding vector field," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 4475–4480.

[9] W. Yao, B. Lin, and M. Cao, "Integrated path following and collision avoidance using a composite vector field," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 250–255.

[10] W. Yao, B. Lin, B. D. Anderson, and M. Cao, "Guiding vector fields for following occluded paths," *IEEE Transactions on Automatic Control*, vol. 67, no. 8, pp. 4091–4106, 2022.

[11] S. Li, Z. Li, Z. Yu, B. Zhang, and N. Zhang, "Dynamic trajectory planning and tracking for autonomous vehicle with obstacle avoidance based on model predictive control," *IEEE Access*, vol. 7, pp. 132 074–132 086, 2019.

[12] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *American Control Conference (ACC)*, 2021, pp. 3882–3889.

[13] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 2019.

[14] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM Journal on Control and Optimization*, vol. 56, no. 4, pp. 2712–2733, 2018.

[15] Y. Lu, X. Zhang, X. Xu, and W. Yao, "Learning-based near-optimal motion planning for intelligent vehicles with uncertain dynamics," *arXiv preprint arXiv:2308.03264*, 2023.

[16] X. Zhang, Y. Jiang, Y. Lu, and X. Xu, "Receding-horizon reinforcement learning approach for kinodynamic motion planning of autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 556–568, 2022.

[17] ——, "A receding-horizon reinforcement learning approach for kinodynamic motion planning of autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, 2022.

[18] C. Lian, X. Xu, H. Chen, and H. He, "Near-optimal tracking control of mobile robots via receding-horizon dual heuristic programming," *IEEE transactions on cybernetics*, vol. 46, no. 11, pp. 2484–2496, 2015.

[19] X. Xu, H. Chen, C. Lian, and D. Li, "Learning-based predictive control for discrete-time nonlinear systems with stochastic disturbances," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 12, pp. 6202–6213, 2018.

[20] I. Exarchos, P. Tsiotras, and M. Pachter, "On the suicidal pedestrian differential game," *Dynamic Games and Applications*, vol. 5, pp. 297–317, 2015.

[21] Y. Lu, W. Li, X. Zhang, and X. Xu, "Continuous-time receding-horizon reinforcement learning and its application to path-tracking control of autonomous ground vehicles," *Optimal Control Applications and Methods*, 2021.

[22] X. Xu, D. Hu, and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning," *IEEE Transactions on Neural Networks*, vol. 18, no. 4, pp. 973–992, 2007.

[23] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–6.

[24] B. Li, T. Acarman, Y. Zhang, Y. Ouyang, C. Yaman, Q. Kong, X. Zhong, and X. Peng, "Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 11 970–11 981, 2021.

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

[26] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.