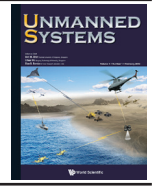
Unmanned Systems, Vol. 14, No. 1 (2026) 239–255

DOI: 10.1142/S230138502550092X

UNMANNED SYSTEMS
https://www.worldscientific.com/worldscinet/us



Data-driven Predictive Control for Safe Motion Planning

Li Dai*, Teng Huang*, Yulong Gao†,‡, Sihang Li*, Yunshan Deng*, Yuanqing Xia*

*School of Automation, Beijing Institute of Technology, Haidian District, Beijing 100081, P. R. China

†Department of Electrical and Electronic Engineering, Imperial College London, London, UK

Controlling a constrained dynamic system in an environment with multiple obstacles is important yet challenging. Many existing methods are either heuristic (e.g. A* algorithm) or model-based (e.g. optimal control). In contrast to these methods, this paper addresses scenarios where the mathematical model of the dynamic system is unknown, relying solely on input–output data and environmental information. We propose a new data-driven framework to achieve safe path planning and efficient tracking control by integrating sample-based methods with more recent data-enabled predictive control. In the offline phase, we develop a safe path planning algorithm to generate a sequence of convex safe sets from the initial point to the target set. This is achieved by leveraging a sample-based planning algorithm and solving bi-linear optimization problems. The resulting adjacent safe sets have a nonempty intersection, and the distance between each safe set and any obstacle exceeds the required safe distance. In the online phase, we develop an efficient data-enabled predictive tracking control algorithm with the core of safe set contraction constraints to sequentially track the safe sets. The proposed algorithm transforms the nonconvex obstacle avoidance control problem into a convex optimization problem, which can be solved efficiently. We demonstrate that the proposed framework is safe, efficient, and scalable through quadcopter simulations, comparison simulations, and unmanned vehicle experiments.

Keywords: Motion planning; data-driven control; model predictive control (MPC); mobile robot.

US

1. Introduction

1.1. *Motivations and contributions*

Motion planning is one of the most important steps toward deploying robotic systems in real life [1]. It is of great importance to develop efficient planning algorithms that enable robots to safely complete desired tasks [2]. The applications include but are not limited to autonomous driving [3, 4], air traffic management [5], and multi-robot cooperation [6, 7]. However, due to the obstacles in the environment, the planning problem is highly nonconvex, which is difficult for real-time implementation. In particular, the popular heuristic methods and optimization-based algorithms often fail to guarantee that the trajectory between discrete points remains free from intersections with obstacles, as highlighted in Fig. 1. Furthermore, thorough modeling and identification of robotic systems are usually expensive. The central problem of this paper is how to design a safe and computationally efficient motion planning algorithm for unknown systems subject to input and output constraints in an environment with multiple polytopic obstacles.

In order to solve this problem, we propose a novel data-driven framework that convexifies the classical data-enabled predictive control (DeePC) [8] for safe planning in a nonconvex environment. The key idea is to transform the planning problem to be a two-phase problem. The offline phase is to plan a sequence of safe convex sets by using a popular sample-based algorithm, e.g. A^* algorithm. We note that this stage is independent of the system modeling, that

Received 25 April 2024; Revised 29 November 2024; Accepted 29 November 2024; Published 17 January 2025. This paper was recommended for publication in its revised form by editorial board member, Lin Zhao.
Email Address: ‡*yulong.gao@imperial.ac.uk*

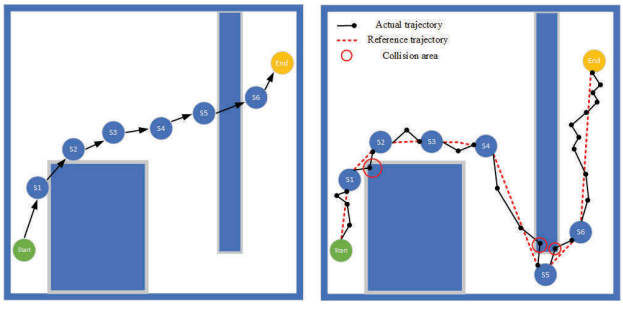


Fig. 1. The path between sample points cuts the corner of the obstacles.

is, it only needs to access the environment and obstacle information while not requiring any knowledge of the system. The online phase is to sequentially track these safe convex sets by using the DeePC. The DeePC is a new variant of model predictive control (MPC), which however does not rely on a parametric system representation but uses system data to directly describe system behaviors. Different from the DeePC in the literature [9, 10], the tracking control in our paper does not require a given point-based reference, but leverages a set contraction approach. Our data-driven framework provides an alternative solution to the important yet challenging safe planning problem. The main contribution of this paper is threefold.

(i) We propose an efficient approach to convert nonconvex obstacle avoidance control problems into convex solvable optimization problems. Specifically, by integrating sample-based motion planning algorithms (such as the A* algorithm), we generate a sequence of convex and safe sets from the initial point to the target set. Any two adjacent sets have a nonempty intersection, and the distance between each set and any obstacle is greater than the required safe distance. Additionally, each individual safe set maximally covers obstacle-free space to reduce algorithm conservatism. Furthermore, we demonstrate that each safe set can be computed by solving a bi-linear optimization problem, which can be solved in polynomial time complexity for any tolerance. Therefore, the proposed path planning algorithm in this paper has the advantages of efficiency, low conservatism, and guaranteed safety performance.

(ii) We extend the application scope of DeePC to tracking control of unknown systems. Based on the safe sets sequence constructed in this paper, in the design of the tracking controller, we innovatively introduce set contraction constraints, ensuring that the system state remains within the safe sets sequence. Through the safe set contraction strategy, the system state continuously converges toward the target point. We also develop a target switching strategy, where the system state entering the next safe set, the target point is switched, and the optimization problem is restructured. This allows tracking control for a series of target points rather than a single one, ensuring that the system state consistently meets safety constraints.

(iii) We show how to handle the noisy data in our framework, which is inspired by the recent results on regularization for robust control. We examine our results through quadcopter simulations, comparison simulations, and unmanned vehicle experiments. The results of simulation and experiments validate the effectiveness and efficiency of our algorithms.

1.2. *Related work*

Motion planning by and large refers to planning a collision-free trajectory from an initial point to a target region in a given working environment with obstacles [11]. There are numerous works on robotic motion planning, either theoretical or algorithmic (or both). The problem setups vary largely in different works, depending on the environment assumption, performance criteria, and system kinetics [12]. In the following, we will restrict our attention to the several classical motion planning methods that are related to this paper.

Graph-based and sample-based methods. The graph-based algorithms build a search tree as quickly as possible from the original point to the target point until a feasible path is found [13]. One popular algorithm is A^* algorithm [14], which utilizes a heuristic function to find the shortest path. It has been shown that A^* algorithm has the properties of completeness, optimality, and optimal efficiency [13]. Sample-based algorithms rely on the random sampling in the workspace. One example is the rapidly exploring random tree (RRT) algorithm [15, 16], which constructs a tree that attempts to quickly and uniformly search the workspace. Another well-known variant of RRT algorithm is RRT* algorithm [17, 18]. The RRT and RRT* algorithms are shown to be probabilistic complete. In general, both graph-based and sample-based methods return a sequence of discrete points, which are independent of the system dynamics. The robotic systems then track these discrete points through real-time control. It may be hard to ensure the safety of real-time tracking, in the case that the sampled points are sparse so that the segment line of two points cuts the corner of obstacles [19]. In order to solve this problem, the planned discrete points from the graph-based and sample-based methods (which we term "heuristic motion planning methods") are used in our paper to provide guidance for constructing the connected safe sets (see Sec. 4).

Optimization-based methods. The optimization-based methods work by solving optimization problems that

naturally take into account system dynamics and various constraints (e.g. control input limitation and safety requirement). Among them, mixed integer programming (MIP) [20] and MPC [4] are two popular formulations. The MIP-based methods use integer variables to encode the collision avoidance between the robot and the obstacles [20]. However, these methods are computationally expensive for real-time implementation. The MPC-based motion planning problem also encounters similar computation issues due to the nonconvex nature of the problem. Recently, some effort has been devoted to convexifying the nonconvex feasible region, resulting in convex MPC formulation [21]. In [22], the MPC is used to track the discrete path which is initially planned as a prior. Note that these methods are purely model-based, that is, they depend on accurate system modeling, which may be however hard in practice. A major difference with these work is that in this paper, we consider dynamic systems with unknown models and we are only accessible to input/output data.

Machine learning-based methods. Many machine learning-based methods have been proposed for robotic motion planning. In particular, there have been many works on (deep) reinforcement learning (RL)-based motion planning, see e.g. [23, 24]. These methods significantly extend traditional motion planning to more complex scenarios, e.g. vision-based planning [25]. We refer interested readers to [26] for a more comprehensive survey. We remark that the RL-based methods highly rely on the rewarding structure as well as the training quality if the deep neural network is involved. Although these methods are model-free, it is very hard to provide a provable safety guarantee.

In the field of systems and control, data-driven control [27] has attracted much attention, in particular thanks to the behavioral systems theory [28]. It is shown that a persistently exciting trajectory of a linear time-invariant system suffices to reproduce all system trajectories, without explicitly identifying the system, which we call the fundamental lemma. Based on this, a DeePC method was proposed in the context of MPC framework [8]. Later, this method was extended to a class of nonlinear systems [29]. In [30], a tracking controller was proposed to track a sequence of set points based on a behavior model. Different from these work, we apply the DeePC framework to the motion planning in nonconvex scenarios. More specifically, by constructing convex safe sets in a fine-grained way, we extend the DeePC to track a sequence of convex safe sets, rather than a sequence of set points (see Sec. 5).

This paper is structured as follows: Sec. 2 defines the motion planning problem with collision avoidance. Section 3 explains the overall scheme of the proposed method. Section 4 describes the main steps for generating the optimal trajectory based on safe sets sequence and Sec. 5 describes the data-driven tracking control. The proposed algorithm is evaluated by numerical and practical experiments, as detailed in Secs. 6 and 7. Section 8 concludes this paper.

Notations: Let $\mathbb{N}$ and $\mathbb{Z}_{>0}$ be the set of nonnegative integers and the set of positive integers, respectively. For $a, b \in \mathbb{N}$ and $a \leq b$, let $\mathbb{N}_{[a,b]} = \{z \in \mathbb{N} | a \leq z \leq b\}$. The Minkowski sum is denoted by $A \oplus B = a + b | a \in A, b \in B$. The inequality $>$ is element-wise for vectors.

2. Preliminaries and Problem Statement

In this section, we first introduce the data-driven behavior model and then formulate the problem to be studied in this paper.

2.1. *Data-driven behavior model*

Consider a controllable and observable discrete-time linear time-invariant system given by

$$x_{t+1} = Ax_t + Bu_t, \tag{1a}$$
$$y_t = Cx_t + Du_t, \tag{1b}$$

where $x_t \in \mathbb{R}^{n_x}$, $u_t \in \mathbb{R}^{n_u}$, $y_t \in \mathbb{R}^{n_y}$ are the state, control input, and output of the system at time $t \in \mathbb{N}$, respectively. We assume that the matrices $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, $C \in \mathbb{R}^{n_y \times n_x}$, and $D \in \mathbb{R}^{n_y \times n_u}$ are unknown.

Instead of identifying the system parameters, one can use the collected input and output data to characterize the dynamics of the system through the behavioral model. Given a horizon $T \in \mathbb{Z}_{>0}$, let $\hat{u}_{[1,T]} = [\hat{u}_1^\top, \hat{u}_2^\top, \ldots, \hat{u}_T^\top]^\top \in \mathbb{R}^{n_u T}$ and $\hat{y}_{[1,T]} = [\hat{y}_1^\top, \hat{y}_2^\top, \ldots, \hat{y}_T^\top]^\top \in \mathbb{R}^{n_y T}$ be an input trajectory and the corresponding output trajectory, which are collected from the unknown system (1) during an offline procedure. Given $L \in \mathbb{Z}_{>0}$ with $L \leq T$, define Hankel matrices $\mathcal{H}_L(\hat{u}_{[1,T]})$ and $\mathcal{H}_L(\hat{y}_{[1,T]})$ associated with the input and output data as

$$\left[\begin{array}{c} \mathcal{H}_L(\hat{u}_{[1,T]}) \\ \hline \mathcal{H}_L(\hat{y}_{[1,T]}) \end{array}\right] = \left[\begin{array}{cccc} \hat{u}_1 & \hat{u}_2 & \cdots & \hat{u}_{T-L+1} \\ \hat{u}_2 & \hat{u}_3 & \cdots & \hat{u}_{T-L+2} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{u}_L & \hat{u}_{L+1} & \cdots & \hat{u}_T \\ \hline \hat{y}_1 & \hat{y}_2 & \cdots & \hat{y}_{T-L+1} \\ \hat{y}_2 & \hat{y}_3 & \cdots & \hat{y}_{T-L+2} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{y}_L & \hat{y}_{L+1} & \cdots & \hat{y}_T \end{array}\right].$$

Each column of the Hankel matrices contains a sequence of consecutive input or output data points of length L.

The data used to characterize the system model should satisfy the condition of persistent excitation, as defined below.

Definition 2.1. Given $L, T \in \mathbb{Z}_{>0}$ with $L \leq T$, the input trajectory $\hat{u}_{[1,T]}$ is persistently exciting of order L if the Hankel matrix $\mathcal{H}_L(\hat{u}_{[1,T]})$ is of full row rank.

Willems *et al.*'s fundamental lemma [28] implies that if the input trajectory is persistently exciting, the output trajectory of the system (1) is predictable for any input. Here we recall the lemma in a state-space system context.

Lemma 2.2. *Consider system (1) and let $\hat{u}_{[1,T]}$ and $\hat{y}_{[1,T]}$ be input and output trajectories of (1), respectively. If the pair (A, B) is controllable and the trajectory $\hat{u}_{[1,T]}$ is persistently exciting of order $n_x + L$, then every length L input–output trajectory $(u_{[1,L]}, y_{[1,L]})$ is a trajectory of system (1) if and only if there exists $g \in \mathbb{R}^{T-L+1}$ such that*

$$\begin{bmatrix} \mathcal{H}_L(\hat{u}_{[1,T]}) \\ \mathcal{H}_L(\hat{y}_{[1,T]}) \end{bmatrix} g = \begin{bmatrix} u_{[1,L]} \\ y_{[1,L]} \end{bmatrix}.$$

2.2. *Problem formulation*

In this section, we will formulate the motion planning problem of the system (1) with unknown model information in a nonconvex environment. Let

$$u_t \in \mathcal{U} = \{z \in \mathbb{R}^{n_u} | F_u z \leq f_u\}, \tag{2}$$
$$y_t \in \mathcal{Y} = \{z \in \mathbb{R}^{n_y} | F_y z \leq f_y\} \tag{3}$$

be the input and output constraints and let

$$\bar{\mathcal{Y}}_{\text{target}} = \{z \in \mathbb{R}^{n_y} | F_{\text{target}} z \leq f_{\text{target}}\}$$

with $\bar{\mathcal{Y}}_{\text{target}} \subset \mathcal{Y}$ be the target set. In the set $\mathcal{Y}$, there are m polytopic obstacles, each of which is expressed as

$$\mathcal{O}^{(i)} = \{y \in \mathbb{R}^{n_y} | O^{(i)} y \leq o^{(i)}\},$$

where $O^{(i)} \in \mathbb{R}^{\ell_i \times n_y}, o^{(i)} \in \mathbb{R}^{\ell_i}$, ℓ_i is the number of constraints for the ith obstacle, and $i = 1, 2, \ldots, m$. Denote by $\mathcal{O}$ the set occupied by the obstacles, i.e. $\mathcal{O} = \mathcal{O}^{(1)} \cup \ldots \cup \mathcal{O}^{(m)}$. Without loss of generality, we assume that the target set $\bar{\mathcal{Y}}_{\text{target}}$ is safe, i.e. $\bar{\mathcal{Y}}_{\text{target}} \cap \mathcal{O} = \emptyset$.

The objective is to design a safe and efficient motion planning algorithm for the system (1) with unknown model information such that

(i) the system reaches the target set $\bar{\mathcal{Y}}_{\text{target}}$ from the initial point y_{start};
(ii) the input and output constraints (2)–(3) are satisfied;
(iii) there is no collision with the obstacle region $\mathcal{O}$.

Towards this objective, we will develop a data-driven predictive control framework in the next section. Before that, we introduce a running example to elaborate the above notations.

Example 2.3. Consider a mobile robot moving in an environment with six obstacles as shown in Fig. 2. The obstacles $\mathcal{O}^{(i)}$, $i = 1, 2, \ldots, 6$ are represented by light gray areas. The working space $\mathcal{Y}$ defined in (3) is enclosed by a thick solid gray line. The control task is to steer the robot from the initial blue circle point $[0.25, 0.25]^\top$ to a red square target set $\bar{\mathcal{Y}}_{\text{target}}$ centered at $(2.5, 2)$ with the side length 0.02 m. The minimal distance between the robot and the obstacles is 0.02 m. We assume that the system model is unknown and only input and output data are available, and its control input is the velocity of the x and y axes. The control input constraint set $\mathcal{U}$ in (2) is therefore given by $\mathcal{U} = \{u = [v_x, v_y]^\top | |v_x| + |v_y| \leq 1\}$.

3. Data-Driven Predictive Control Framework

In this section, we will provide an overview of the sample-based DeePC framework. This framework consists of the offline planning phase and the online tracking phase. In the offline phase, we employ the sampled-based algorithm to produce a sequence of safe sets in an alternating way. These safe sets satisfy a group of properties that facilitate the online tracking phase. In the online phase, leveraging the behavioral systems theory and a set contraction method, we solve a sequence of DeePC problems in real time to sequentially track the safe sets that are generated from the offline phase.

Before providing the details, we define the distance between sets.

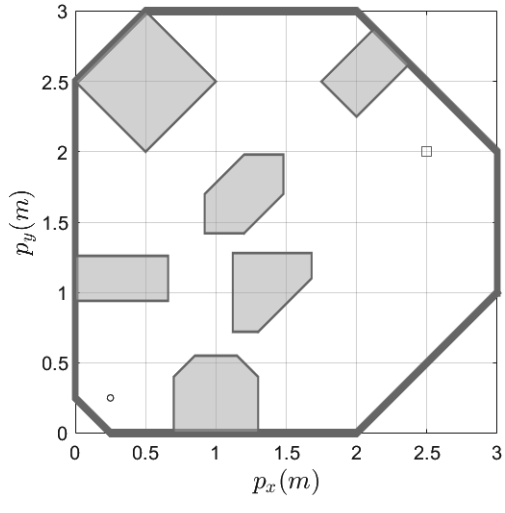


Fig. 2. Control task in the environment with six obstacles.

Definition 3.1 (Set distance). The distance between two sets $\mathbb{X}_1$ and $\mathbb{X}_2$, where $\mathbb{X}_1, \mathbb{X}_2 \subseteq \mathbb{R}^n$, is defined as

$$\text{dist}(\mathbb{X}_1, \mathbb{X}_2) = \inf\{\|x_1 - x_2\| | x_1 \in \mathbb{X}_1, x_2 \in \mathbb{X}_2\}. \quad (4)$$

Now let us elaborate the offline planning phase. The problem to be solved in this phase is as follows.

Problem 3.2. Given the information y_{start}, $\bar{\mathcal{Y}}_{\text{target}}$, $\mathcal{Y}$, and $\mathcal{O}$, design an algorithm that automatically generates a sequence of safe sets $\mathbb{Y} = \{\mathcal{Y}_0^{\text{safe}}, \mathcal{Y}_1^{\text{safe}}, \ldots, \mathcal{Y}_{N_{\text{set}}-1}^{\text{safe}}, \mathcal{Y}_{N_{\text{set}}}^{\text{safe}}\}$ with $\mathcal{Y}_{N_{\text{set}}}^{\text{safe}} = \bar{\mathcal{Y}}_{\text{target}}$ such that

(i) $y_{\text{start}} \in \mathcal{Y}_0^{\text{safe}}$;
(ii) $\mathcal{Y}_i^{\text{safe}} \subset \mathcal{Y}$, $\forall i \in \mathbb{N}_{[0,N_{\text{set}}]}$;
(iii) $\text{dist}(\mathcal{Y}_i^{\text{safe}}, \mathcal{O}) \geq d_{\min}$, $\forall i \in \mathbb{N}_{[0,N_{\text{set}}]}$;
(iv) $\mathcal{Y}_i^{\text{safe}} \cap \mathcal{Y}_{i+1}^{\text{safe}} \neq \emptyset$, $\forall i \in \mathbb{N}_{[0,N_{\text{set}}-1]}$,

where $d_{\min}$ is a strictly positive constant.

In order to efficiently solve this problem, we parameterize each safe set $\mathcal{Y}_i^{\text{safe}}$ to be in the form of

$$\mathcal{Y}_i^{\text{safe}} = \bar{\mathcal{Y}}(y_i^s, \alpha_i) = \alpha_i \mathcal{Y} \oplus (1 - \alpha_i) y_i^s, \quad (5)$$

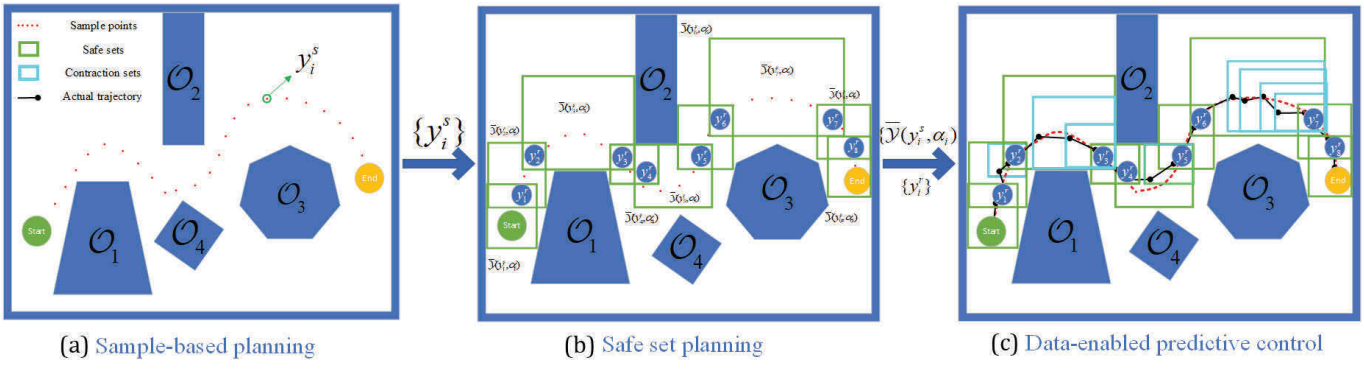
where $y_i^s \in \mathcal{Y}$ and $\alpha_i \in [0, 1]$. The key idea is to employ the sample-based heuristic planning algorithms, e.g. A^* algorithm [14], to generate y_i^s, and then find the optimal scaling coefficient $\alpha_i = \max_{\alpha \in [0,1]} \alpha$ such that $\text{dist}(\bar{\mathcal{Y}}(y_i^s, \alpha), \mathcal{O}) \geq d_{\min}$. Note that the randomness of the sample-based planning algorithms may result in a sequence of samples that are not sufficiently dense such that condition (iv) is not satisfied. In this case, there is a need to interpolate new samples. On the other hand, if the generated safe sets are too nested, the online computation cost will be high. Thus, removing redundant safe sets is necessary. A detailed solution to Problem 3.2 is given in Sec. 4, where we propose a new algorithm that automatically returns a sequence of safe sets $\mathbb{Y} = \{\bar{\mathcal{Y}}(y_{\text{start}}, \alpha_0), \bar{\mathcal{Y}}(y_1^s, \alpha_1), \ldots, \bar{\mathcal{Y}}(y_{N_{\text{set}}-1}^s, \alpha_{N_{\text{set}}-1}), \bar{\mathcal{Y}}_{\text{target}}\}$ that fulfills the conditions (i)–(iv). An illustration is shown in Figs. 3(a) and 3(b). In Fig. 3(a), the red dots are the outcomes from the sample-based planning algorithm. These samples are collision-free and connect the initial point y_{start} and the target region $\bar{\mathcal{Y}}_{\text{target}}$. Given these samples, Fig. 3(b) shows the constructed safe sets (the green rectangles). We can see that these safe sets are nested with each other and satisfy the conditions (i)–(iv).

The safe sets obtained by solving Problem 3.2 are then used for the online tracking problem.

Problem 3.3. Given a sequence of safe sets $\mathbb{Y}$ that satisfy the conditions (i)–(iv), design a data-driven tracking controller such that the system with unknown dynamics can sequentially move along the safe sets and satisfy the input constraints.

In order to solve Problem 3.3, we leverage the historical data of the system (1) and the behavioral systems theory to describe the unknown dynamics. We reformulate the popular DeePC to track the safe sets in a set-contraction manner. The structure of the resulting optimization problem is parameterized by the safe set $\bar{\mathcal{Y}}(y_i^s, \alpha_i)$, more precisely, by the vector y_i^s and the scalar α_i. This parameterization offers us a more efficient way to solve the problem using the off-the-shelf solver. As shown in Fig. 3(c), the blue rectangles are the contraction sets while the black dotted line is the actual trajectory under our tracking controller. The solution to Problem 3.3 will be detailed in Sec. 5.

The algorithm proposed in this paper mainly addresses the above two problems by providing path planning and tracking control algorithms. In the path planning part, an initial path is obtained using heuristic control algorithm based on the given environmental information. Subsequently, the safety set sequence generation method designed in this paper is used to determine the tracking target points and corresponding safety set control sequences. A tracking controller is then designed to track the target points by applying contraction constraints. When the set conditions are met, the tracking target is switched to achieve tracking control of target points and safety set sequences, thereby implementing path planning and



(a) Sample-based planning (b) Safe set planning (c) Data-enabled predictive control

Fig. 3. Sketch of sample-based DeePC framework.

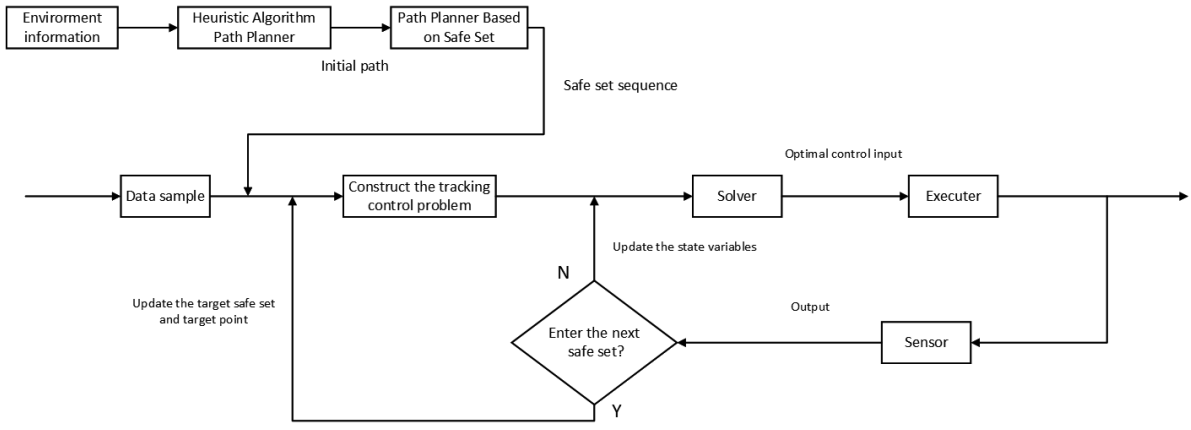


Fig. 4. Data-driven tracking control block diagram.

tracking control under a data-driven framework. The block diagram is shown in Fig. 4.

4. Sample-Based Safe Set Planning

In this section, we first show how to construct safe set $\bar{\mathcal{Y}}(y^s,\alpha)$ in (5) satisfying $\text{dist}(\bar{\mathcal{Y}}(y^s,\alpha),\mathcal{O}) \geq d_{\min}$. Then, we design an algorithm that is able to automatically generate a sequence of safe sets satisfying the conditions in Problem 3.2.

4.1. *Constructing a single safe set* $\bar{\mathcal{Y}}(y^s,\alpha)$

Following [31], let us first recall some basic properties of the set $\bar{\mathcal{Y}}(y^s,\alpha)$ defined in (5). It is easy to show that

$$\begin{aligned}\bar{\mathcal{Y}}(y^s,\alpha) &= \{(1-\alpha)y^s\} \oplus \alpha\mathcal{Y} \\ &= \{y \in \mathbb{R}^{n_y} | F_y y \leq \alpha f_y + F_y(1-\alpha)y^s\}.\end{aligned}$$

For any $y^s \in \mathbb{R}^{n_y}$, the set $\bar{\mathcal{Y}}(y^s,\alpha)$ is monotonically increasing in the sense of set inclusion with α, i.e. $\bar{\mathcal{Y}}(y^s,\alpha_1) \subseteq \bar{\mathcal{Y}}(y^s,\alpha_2)$ for any $0 \leq \alpha_1 \leq \alpha_2 \leq 1$. Recall that $\mathcal{O} = \bigcup_{i=1}^m \mathcal{O}^{(i)}$. Thanks to the monotonicity, we are interested in solving the following problem:

$$\begin{cases} \max\limits_{\alpha\in[0,1]} & \alpha \\ \text{s.t.} & \text{dist}(\bar{\mathcal{Y}}(y^s,\alpha),\mathcal{O}^{(i)}) \geq d_{\min}, \forall\, i \in \mathbb{N}_{[1,m]}, \end{cases} \tag{6}$$

where $y^s \in \mathbb{R}^{n_y}$ and $d_{\min} > 0$ are known.

Note that even through the sets $\bar{\mathcal{Y}}(y^s,\alpha)$ and $\mathcal{O}^{(i)}$ are convex polytopes, the constraint $\text{dist}(\bar{\mathcal{Y}}(y^s,\alpha),\mathcal{O}^{(i)}) \geq d_{\min}$ is nonconvex with respect to α. Leveraging the results in [32], we reformulate the safety constraints that only involve continuous decision variables, which is provided in the following lemma.

Lemma 4.1. *The safety constraint (6) holds if and only if there exist* $\{\mu_i\}_{i=1}^m$, λ, *and* z *with appropriate dimensions such that*

$$\begin{cases} \forall\, i \in \mathbb{N}_{[1,m]}, \\ \quad \begin{cases} -{o^{(i)}}^\top \mu_i - (\alpha f_y + F_y(1-\alpha)y^s)^\top \lambda \geq d_{\min}, \\ {O^{(i)}}^\top \mu_i - z = 0, \quad \mu_i > 0, \end{cases} \\ F_y^\top \lambda + z = 0, \\ \|z\| \leq 1, \quad \lambda > 0. \end{cases} \tag{7}$$

Proof. Please refer to [32] for a detailed proof. □

To this end, the problem (6) can be rewritten as

$$\begin{cases} \max\limits_{\alpha,\{\mu_i\}_{i=1}^m,\lambda,z} & \alpha \\ \text{s.t.} & \alpha \in [0,1] \text{ and } (7). \end{cases} \tag{8}$$

The problem (8) is nonlinear since the constraints involve the multiplication of α and λ. Let us denote by $\Pi(\alpha)$ the set of $\{\mu_i\}_{i=1}^m$, λ, and z such that (7) holds. Note that when α is fixed, the set $\Pi(\alpha)$ consists of a group of convex inequalities and equalities, and thus one can check if $\Pi(\alpha)$ is empty by solving a convex optimization problem.

In order to efficiently solve this problem, we propose to use a bisection method, as detailed in Algorithm 1. We initialize $\underline{\alpha}$, $\bar{\alpha}$, and α_{mid} to be 0, 1, and $1/2$, respectively. In the while-loop, we repeat to check if the set $\Pi(\alpha_{\text{mid}})$ is empty and update $\underline{\alpha}$ and $\bar{\alpha}$ accordingly, until the difference between $\underline{\alpha}$ and $\bar{\alpha}$ converges to a certain tolerance ε.

Proposition 4.2. *If the problem (8) is feasible, Algorithm 3 terminates in* $\lfloor \log_2(1/\varepsilon) \rfloor$ *steps and* $\alpha^* - \hat{\alpha}^* \leq \varepsilon$, *where* α^* *is the optimal solution to the problem (8) and* $\hat{\alpha}^*$ *is the output from Algorithm 1.*

Proof. If the problem (8) is feasible, it is easy to see that $\Pi(0)$ is nonempty. From Algorithm 1, we have that $\Pi(\underline{\alpha})$ is always nonempty, that is, the problem (8) is feasible if α is fixed to be $\underline{\alpha}$. The bisection method yields that $\bar{\alpha} - \underline{\alpha} = (1/2)^k$, where k is the iteration step. Let

Algorithm 1. Bisection Method for Solving (8)
Input: $0 < \varepsilon < 1$
Output: $\hat{\alpha}^*$
1: $\underline{\alpha} \leftarrow 0$, $\bar{\alpha} \leftarrow 1$, $\alpha_{\text{mid}} \leftarrow 1/2$, $k \leftarrow 0$
2: **while** $\bar{\alpha} - \underline{\alpha} > \varepsilon$ **do**
3: **if** $\Pi(\alpha_{\text{mid}})$ is empty **then**
4: $\bar{\alpha} \leftarrow \alpha_{\text{mid}}$
5: $\alpha_{\text{mid}} \leftarrow (\underline{\alpha} + \bar{\alpha})/2$
6: **else**
7: $\underline{\alpha} \leftarrow \alpha_{\text{mid}}$
8: $\alpha_{\text{mid}} \leftarrow (\underline{\alpha} + \bar{\alpha})/2$
9: **end if**
10: $k \leftarrow k + 1$
11: **end while**
12: $\hat{\alpha}^* = \underline{\alpha}$

$(1/2)^k \leq \varepsilon$, which gives that the maximal iteration step satisfies $k \geq \lfloor \log_2(1/\varepsilon) \rfloor$. Since the optimal solution α^* to the problem (8) lies in the interval $[\underline{\alpha}, \bar{\alpha}]$, we have that $\alpha^* - \hat{\alpha}^* \leq \varepsilon$. □

4.2. *Sample-based safe set planning*

Based on the construction of a single safe set in Sec. 4.1, this subsection will show how to construct a sequence of safe sets that fulfil the requirements in Problem 3.2. The detailed procedures are outlined in Algorithm 2. In the following, we will elaborate this algorithm.

Input: The inputs consist of the initial point y_{start}, the target zone $\bar{\mathcal{Y}}_{\text{target}}$, the set $\mathcal{O}$ occupied by obstacles, the working space $\mathcal{Y}$ and the safe distance $d_{\min}$. A sequence of samples in the working space $\mathcal{Y}$, denoted by $\mathcal{P} = \{y_0^s, y_1^s, \ldots, y_{N_{\text{sample}}-1}^s, y_{N_{\text{sample}}}^s\}$, is generated by leveraging the heuristic algorithm, e.g. A^* algorithm. Here N_{sample} is the number of samples. This sequence starts from the initial point $y_0^s = y_{\text{start}}$ and ends within $\bar{\mathcal{Y}}_{\text{target}}$, i.e. $y_{N_{\text{sample}}}^s \in \bar{\mathcal{Y}}_{\text{target}}$. Moreover, all the samples are collision-free in the sense of $\text{dist}(y_j^s, \mathcal{O}) > d_{\min}$, for all $j \in \mathbb{N}_{[1,N_{\text{sample}}]}$.

In order to illustrate the above algorithm, we continue to use Example 2.3 for discussion.

Example 4.3. Let us consider the planning task as shown in Example 2.3. For the input step of Algorithm 2, we run A^* algorithm to plan a safe path, as shown in Fig. 5(a), where colored line segments represent different exploration directions. Based on this, we distill a safe path that starts from the initial point y_{start} and reaches the target set $\bar{\mathcal{Y}}_{\text{target}}$ while avoiding collision with the obstacles, which is shown in Fig. 5(b).

Initialization (line 1): We initialize the safe set sequence to be $\mathbb{Y} = \{\bar{\mathcal{Y}}_{\text{target}}\}$ and the sample sequence to be

Algorithm 2. Sample-based Safe Set Planning (8)
Input:
Initial point y_{start}, target zone $\bar{\mathcal{Y}}_{\text{target}}$, obstacles $\mathcal{O}$, working space $\mathcal{Y}$, and safe distance $d_{\min}$. A sequence of samples $\mathcal{P} = \{y_0^s, y_1^s, \ldots, y_{N_{\text{sample}}-1}^s, y_{N_{\text{sample}}}^s\}$ with the following properties:

$$\begin{cases} y_j^s \in \mathcal{Y},\ \forall j \in \mathbb{N}_{[1,N_{\text{sample}}]}, \\ y_0^s = y_{\text{start}},\ y_{N_{\text{sample}}}^s \in \bar{\mathcal{Y}}_{\text{target}} \\ \text{dist}(y_j^s, \mathcal{O}) > d_{\min},\ \forall j \in \mathbb{N}_{[1,N_{\text{sample}}]} \end{cases}$$

Output: A sequence of safe sets $\mathbb{Y} = \{\bar{\mathcal{Y}}(y_0^s, \alpha_0), \bar{\mathcal{Y}}(y_1^s, \alpha_1), \ldots, \bar{\mathcal{Y}}(y_{N_{\text{set}}-1}^s, \alpha_{N_{\text{set}}-1}), \bar{\mathcal{Y}}_{\text{target}}\}$
1: Initialize $\mathcal{P} \leftarrow \mathcal{P}[1 : N_{\text{sample}}]$, $\mathbb{Y} \leftarrow \{\bar{\mathcal{Y}}_{\text{target}}\}$
2: **while** $\mathcal{P}$ is not empty **do**
3: **if** $\nexists i \in \mathbb{N}_{[0,\text{Length}(\mathcal{P})-1]}$ such that $y_i^s \in \mathbb{Y}[1]$ **then**
4: Interpolate a new sample y_{new}^s between $\mathcal{P}[\text{End}]$ and $\mathbb{Y}[1]$ such that $y_{\text{new}}^s \in \mathbb{Y}[1]$ and $\text{dist}(y_{\text{new}}^s, \mathcal{O}) > d_{\min}$
5: Let

$$\begin{cases} y_{\text{Length}(\mathcal{P})}^s \leftarrow y_{\text{new}}^s \\ \mathcal{P} \leftarrow \{\mathcal{P},\ y_{\text{Length}(\mathcal{P})}^s\} \end{cases}$$

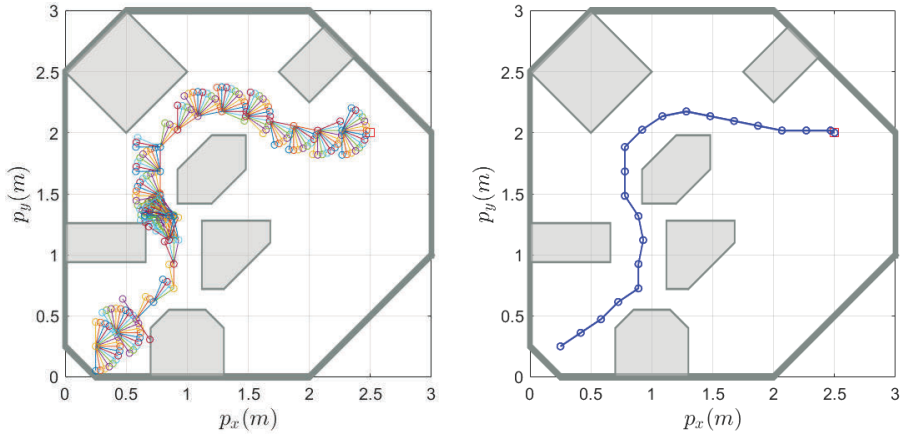
6: **end if**
7: Let

$$\begin{cases} i \leftarrow \min\{j \mid y_j^s \in \mathbb{Y}[1]\} \\ y^s \leftarrow y_i^s \\ \mathcal{P} \leftarrow \mathcal{P}[1 : i] \end{cases}$$

8: Solve the optimization problem (11) and let α^* be the optimal solution
9: Let $\mathbb{Y} \leftarrow \{\bar{\mathcal{Y}}(y^s, \alpha^*),\ \mathbb{Y}\}$
10: **end while**
11: ▷ For a sequence $\mathcal{P}$, $\mathcal{P}[1 : i]$ denotes a sequence generated from the first i elements from $\mathcal{P}$, $\mathcal{P}[1]$ denotes the first element of $\mathcal{P}$, and $\mathcal{P}[\text{End}]$ denotes the last element of $\mathcal{P}$
12: Rename and re-index $\mathbb{Y}$ to be $\mathbb{Y} = \{\bar{\mathcal{Y}}(y_0^s, \alpha_0), \bar{\mathcal{Y}}(y_1^s, \alpha_1), \ldots, \bar{\mathcal{Y}}(y_{N_{\text{set}}-1}^s, \alpha_{N_{\text{set}}-1}), \bar{\mathcal{Y}}_{\text{target}}\}$
▷ N_{set} denotes the number of the safe sets
13: **Return** $\mathbb{Y}$

$\mathcal{P} = \mathcal{P}[1 : N_{\text{sample}}]$ (i.e. removing the last sample $y_{N_{\text{sample}}}^s$), where $\mathcal{P}[1 : i]$ denotes a sequence of the first i elements from $\mathcal{P}$.

While loop (lines 2–10): In the while-loop, we recursively select the active sample y^s from the sequence $\mathcal{P}$ and construct the safe set accordingly by solving the optimization problem (8). More specifically, we first check if there exists some sample in $\mathcal{P}$ that lies in the set $\mathbb{Y}[1]$, where $\mathbb{Y}[1]$ denotes the first element of $\mathbb{Y}$. If not, we interpolate a new sample y_{new}^s between $\mathcal{P}[\text{End}]$ and $\mathbb{Y}[1]$ such that $y_{\text{new}}^s \in \mathbb{Y}[1]$ and stack this new sample to the end of the sequence of $\mathcal{P}$



(a) (b)

Fig. 5. Initial planning in Algorithm 2 by A^* algorithm: (a) running process and (b) distilled path.

(lines 3–6). Otherwise, we select $i = \min\{j | y_j^s \in \mathbb{Y}[1]\}$ and update the sequence to be $\mathcal{P} \leftarrow \mathcal{P}[1:i]$ (line 8). By setting $y^s = y_i^s$, we solve the optimization problem (8) and obtain the optimal scaling coefficient α^*. To this end, we get the new safe set $\bar{\mathcal{Y}}(y^s, \alpha^*)$ and stack this set to be the first element of the sequence $\mathbb{Y}$ (lines 8–9).

Re-index and output (lines 12–13): We re-index the obtained sequence of safe sets and return it in the form of $\mathbb{Y} = \{\bar{\mathcal{Y}}(y_{\text{start}}, \alpha_0), \bar{\mathcal{Y}}(y_1^s, \alpha_1), \ldots, \bar{\mathcal{Y}}(y_{N_{\text{set}}-1}^s, \alpha_{N_{\text{set}}-1}), \bar{\mathcal{Y}}_{\text{target}}\}$.

The following proposition states that the output of Algorithm 4.2 is a solution to Problem 3.2.

Proposition 4.4. *The sequence* $\mathbb{Y} = \{\bar{\mathcal{Y}}(y_0^s, \alpha_0), \bar{\mathcal{Y}}(y_1^s, \alpha_1), \ldots, \bar{\mathcal{Y}}(y_{N_{\text{set}}-1}^s, \alpha_{N_{\text{set}}-1}), \bar{\mathcal{Y}}_{\text{target}}\}$ *from Algorithm* 2 *satisfies the conditions (i)–(iv) in Problem* 3.2.

Proof. The condition (i) holds since the sample starts from the initial point $y_0^s = y_{\text{start}}$. The condition (ii) $\bar{\mathcal{Y}}(y_i^s, \alpha_i) \subseteq \mathcal{Y}$ naturally holds since the safe set $\bar{\mathcal{Y}}(y_i^s, \alpha_i)$ is a scaled set of $\mathcal{Y}$ with respect to the sample y_i^s. The condition (iii) is true since each sample satisfies $\text{dist}(y_i^s, \mathcal{O}) > d_{\min}$ and the scaling coefficient is obtained by solving the problem (8), which incorporates the safety constraint. Finally, thanks to the point inclusion check and the interpolation in lines 4–8, it is easy to see that any two adjacent safe sets have a nonempty intersection, which validates the condition (iv). □

If the sequence of safe sets obtained from Algorithm 4.2 is too nested, the computation for online tracking will be high. Thus, we expect a sequence of safe sets that are as spare as possible while fulfilling conditions (i)–(iv) in Problem 3.2. Algorithm 2 is proposed to simplify the output from Algorithm 2.

In detail, we start from the first element of the set sequence by setting $\mathbb{Y} = \mathbb{Y}[1]$ and store the remaining sets as $\hat{\mathbb{Y}} = \mathbb{Y}[2:\text{End}]$ (line 1). In the while loop, we recursively update $\hat{\mathbb{Y}}$ and $\mathbb{Y}$. That is, we select $i = \max\{j | \hat{\mathbb{Y}}[j] \cap \mathbb{Y}[\text{End}] \neq \emptyset\}$, stack $\hat{\mathbb{Y}}[i]$ to the end of $\mathbb{Y}$ (i.e. $\mathbb{Y} = \{\mathbb{Y}, \hat{\mathbb{Y}}[i]\}$), and accordingly update $\hat{\mathbb{Y}} = \hat{\mathbb{Y}}[i+1:\text{End}]$ (lines 2–4). A byproduct of Algorithm 3 is a sequence of reference points $\mathcal{T} = \{y_0^r, y_1^r, \ldots, y_{N_{\text{set}}-1}^r\}$ that will be used for online tracking in Sec. 5 (line 3). These reference points lie in the intersection of two adjacent safe sets. Finally, we re-index the sequences $\mathbb{Y}$ and $\mathcal{T}$, and return them as outputs (lines 5–7).

Corollary 4.5 implies that after removing the redundant safe sets, the output of Algorithm 3 is still a solution to Problem 3.2.

Algorithm 3. Simplification of Safe Set Sequence

Input: The sequence of safe sets $\mathbb{Y} = \{\bar{\mathcal{Y}}(y_0^s, \alpha_0), \bar{\mathcal{Y}}(y_1^s, \alpha_1), \ldots, \bar{\mathcal{Y}}(y_{N_{\text{set}}-1}^s, \alpha_{N_{\text{set}}-1}), \bar{\mathcal{Y}}_{\text{target}}\}$ obtained from Algorithm 2

Output: A sequence of safe sets $\mathbb{Y} = \{\bar{\mathcal{Y}}(y_0^s, \alpha_0), \bar{\mathcal{Y}}(y_1^s, \alpha_1), \ldots, \bar{\mathcal{Y}}(y_{N_{\text{set}}-1}^s, \alpha_{N_{\text{set}}-1}), \bar{\mathcal{Y}}_{\text{target}}\}$ and a sequence of reference points $\mathcal{T} = \{y_0^r, y_1^r, \ldots, y_{N_{\text{set}}-1}^r\}$

1: Initialize $\mathbb{Y} \leftarrow \mathbb{Y}[1], \hat{\mathbb{Y}} \leftarrow \mathbb{Y}[2:\text{End}], \mathcal{T} \leftarrow \emptyset$

2: **while** $\hat{\mathbb{Y}}$ is not empty **do**

3: Let

$$\begin{cases} i \leftarrow \max\{j \mid \hat{\mathbb{Y}}[j] \cap \mathbb{Y}[\text{End}] \neq \emptyset\} \\ \text{select } y^r \in \mathbb{Y}[\text{End}] \cap \hat{\mathbb{Y}}[i] \\ \mathbb{Y} \leftarrow \{\mathbb{Y}, \hat{\mathbb{Y}}[i]\} \\ \hat{\mathbb{Y}} \leftarrow \hat{\mathbb{Y}}[i+1:\text{End}] \\ \mathcal{T} \leftarrow \{\mathcal{T},\ y^r\} \end{cases}$$

4: **end while**

5: Let $N_{\text{set}} = \text{Length}(\mathbb{Y})$

6: Rename and re-index $\mathbb{Y}$ and $\mathcal{T}$ to be $\mathbb{Y} = \{\bar{\mathcal{Y}}(y_0^s, \alpha_0), \bar{\mathcal{Y}}(y_1^s, \alpha_1), \ldots, \bar{\mathcal{Y}}(y_{N_{\text{set}}-1}^s, \alpha_{N_{\text{set}}-1}), \bar{\mathcal{Y}}_{\text{target}}\}$ and $\mathcal{T} = \{y_0^r, y_1^r, \ldots, y_{N_{\text{set}}-1}^r\}$, respectively

7: **Return** $\mathbb{Y}$ and $\mathcal{T}$

Corollary 4.5. The sequence $\mathbb{Y} = \{\bar{\mathcal{Y}}(y_0^s, \alpha_0), \bar{\mathcal{Y}}(y_1^s, \alpha_1), \ldots, \bar{\mathcal{Y}}(y_{N_{\text{set}}-1}^s, \alpha_{N_{\text{set}}-1}), \bar{\mathcal{Y}}_{\text{target}}\}$ from Algorithm 3 satisfies the conditions (i)–(iv) in Problem 3.2.

Proof. The conditions (i)–(iii) hold since the input to Algorithm 2 comes from Algorithm 2, which satisfies the conditions (i)–(iv) in Problem 3.2, as shown in Proposition 4.4. It is easy to verify the last condition (iv) due to the operations in line 3. □

Let us continue Example 4.3 for illustrating how to compute the safe sets from Algorithms 2 and 3.

Example 4.6. Based on the path from A^* algorithm, the implementation of Algorithm 2 returns a sequence of safe sets, which are the sets circled by the blue line in Fig. 6(a). We can see that the safe sets are nested densely. We further implement Algorithm 3 to remove redundant sets, which leads to a new sequence of safe sets shown in Fig. 6(b). The number of sets in this new sequence is 16, much smaller than 56, the number of sets in the sequence in Fig. 6(a). The sequences in both Figs. 6(a) and 6(b) fulfill the requirements as stated in Problem 3.2. As shown in Fig. 6(b), the intersection area of two adjacent safe sets is colored in light blue, and the reference point sequence is marked as colored asterisks.

5. Data-Enabled Predictive Tracking Controller

In this section, we will first briefly review DeePC framework [8], and then will adapt DeePC framework to design a data-driven tracking control method based on the safe set sequence for solving Problem 3.3 in Sec. 3.

5.1. *Review of DeePC*

Consider $T_{\text{ini}}, N, T \in \mathbb{N}$ such that $T \geq (n_u + 1)(T_{\text{ini}} + N + n_x) - 1$. Let $\hat{u}_{[1,T]}$ and $\hat{y}_{[1,T]}$ be the input trajectory and the corresponding output trajectory measured offline from system (1). We organize the input–output data into the Hankel matrices $\mathcal{H}_{T_{\text{ini}}+N}(\hat{u}_{[1,T]})$ and $\mathcal{H}_{T_{\text{ini}}+N}(\hat{y}_{[1,T]})$, which are further partitioned into two parts

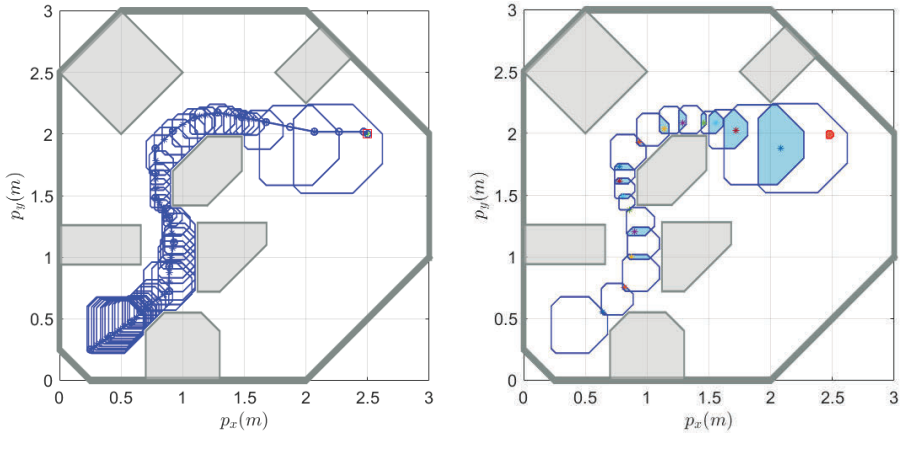
$$\begin{bmatrix} U_p \\ U_f \end{bmatrix} = \mathcal{H}_{T_{\text{ini}}+N}(\hat{u}_{[1,T]}), \quad \begin{bmatrix} Y_p \\ Y_f \end{bmatrix} = \mathcal{H}_{T_{\text{ini}}+N}(\hat{y}_{[1,T]}),$$

where $U_p \in \mathbb{R}^{n_u T_{\text{ini}} \times \kappa}$ and $Y_p \in \mathbb{R}^{n_y T_{\text{ini}} \times \kappa}$ are the first T_{ini} block rows of $\mathcal{H}_{T_{\text{ini}}+N}(\hat{u}_{[1,T]})$ and $\mathcal{H}_{T_{\text{ini}}+N}(\hat{y}_{[1,T]})$, respectively; $U_f \in \mathbb{R}^{n_u N \times \kappa}$ and $Y_f \in \mathbb{R}^{n_y N \times \kappa}$ are the last N block rows of $\mathcal{H}_{T_{\text{ini}}+N}(\hat{u}_{[1,T]})$ and $\mathcal{H}_{T_{\text{ini}}+N}(\hat{y}_{[1,T]})$, respectively; and $\kappa = T - T_{\text{ini}} - N + 1$ is introduced for brevity.

Assumption 5.1. Assume that the offline input trajectory $\hat{u}_{[1,T]}$ is persistently exciting of order $T_{\text{ini}} + N + n_x$.

At each time instant $t \in \mathbb{N}$, let $\mathbf{u}_{\text{ini}} = [u_{t-T_{\text{ini}}}^\top, u_{t-T_{\text{ini}}-1}^\top, \ldots, u_{t-1}^\top]^\top \in \mathbb{R}^{n_u T_{\text{ini}}}$ and $\mathbf{y}_{\text{ini}} = [y_{t-T_{\text{ini}}}^\top, y_{t-T_{\text{ini}}-1}^\top, \ldots, y_{t-1}^\top]^\top \in \mathbb{R}^{n_y T_{\text{ini}}}$ be initial input and output trajectories with length T_{ini} and $\mathbf{u} = [u_{0|t}^\top, u_{1|t}^\top, \ldots, u_{N-1|t}^\top]^\top \in \mathbb{R}^{n_u N}$ and $\mathbf{y} = [y_{0|t}^\top, y_{1|t}^\top, \ldots, y_{N-1|t}^\top]^\top \in \mathbb{R}^{n_y N}$ be the predicted trajectories starting from the initial trajectories with length N. From Lemma 2.2, the entire input–output trajectory $[\mathbf{u}_{\text{ini}}^\top, \mathbf{u}^\top, \mathbf{y}_{\text{ini}}^\top, \mathbf{y}^\top]^\top$ is a trajectory of (1) if and only if there exists $g \in \mathbb{R}^\kappa$ such that

$$\begin{bmatrix} U_p \\ U_f \\ Y_p \\ Y_f \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{ini}} \\ \mathbf{u} \\ \mathbf{y}_{\text{ini}} \\ \mathbf{y} \end{bmatrix}. \tag{9}$$



(a) (b)

Fig. 6. (a) A sequence of safe sets generated by Algorithm 2; (b) a simplified sequence of safe sets generated by Algorithm 2.

The length T_{ini} of the initial trajectory should be chosen no smaller than the *lag* of system (1), such that the predicted output trajectory $\mathbf{y}$ is uniquely determined through (9) for any given predicted input trajectory $\mathbf{u}$. The *lag* is defined as the smallest integer $p \in \mathbb{N}$ such that the matrix $[C^\top, A^\top C^\top, \ldots, (A^\top)^{p-1} C^\top]$ has rank n_x.

By using (9) as an alternative model of (1) to predict the future system behavior, the DeePC algorithm solves the following optimization problem to obtain the optimal predicted input trajectory $\mathbf{u}$ at every sampling instant t.

$$\begin{cases} \min\limits_{g,\mathbf{u},\mathbf{y}} \sum\limits_{k=0}^{N-1} [\|y_{k|t}\|_Q^2 + \|u_{k|t}\|_R^2] + \|y_{N|t}\|_P^2 \\ \text{s.t.} \begin{bmatrix} U_p \\ U_f \\ Y_p \\ Y_f \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{ini}} \\ \mathbf{u} \\ \mathbf{y}_{\text{ini}} \\ \mathbf{y} \end{bmatrix}, \\ u_{k|t} \in \mathcal{U}, \quad \forall k \in \mathbb{N}_{[0,N-1]}, \\ y_{k|t} \in \mathcal{Y}, \quad \forall k \in \mathbb{N}_{[0,N]}, \end{cases} \tag{10}$$

where $Q \in \mathbb{R}^{n_y \times n_y}$, $R \in \mathbb{R}^{n_u \times n_u}$, and $P \in \mathbb{R}^{n_y \times n_y}$ are positive definite. The general process of DeePC algorithm at each time t is summarized below.

5.2. *Data-enabled predictive tracking control*

This section will show how to leverage the DeePC and the safe set sequence obtained in Sec. 4 to design a new data-enabled predictive tracking controller.

Consider the output from Algorithm 3, i.e. the sequence of safe sets $\mathbb{Y} = \{\bar{\mathcal{Y}}(y_0^s, \alpha_0), \bar{\mathcal{Y}}(y_1^s, \alpha_1), \ldots, \bar{\mathcal{Y}}(y_{N_{\text{set}}-1}^s, \alpha_{N_{\text{set}}-1}), \bar{\mathcal{Y}}_{\text{target}}\}$ and the sequence of reference points $\mathcal{T} = \{y_0^r, y_1^r, \ldots, y_{N_{\text{set}}-1}^r\}$. Given the output y_t at time t, if $y_t \notin \bar{\mathcal{Y}}_{\text{target}}$, the active safe set $\bar{\mathcal{Y}}(y_i^s, \alpha_i)$ is defined by selecting $i = \max\{j | y_t \in \bar{\mathcal{Y}}(y_j^s, \alpha_j)\}$. The new optimization problem to be solved at time step t is parameterized by the current output y_t, the active safe set $\bar{\mathcal{Y}}(y_i^s, \alpha_i)$, and the active reference point y_i^r, denoted by $\text{OPT}(y_t, \bar{\mathcal{Y}}(y_i^s, \alpha_i), y_i^r)$:

$$\begin{cases} \min\limits_{g,\mathbf{u},\{\beta_{k|t}\}_{k=0}^N} \beta_{N|t} \\ \text{s.t.} \begin{bmatrix} U_p \\ U_f \\ Y_p \\ Y_f \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{ini}} \\ \mathbf{u} \\ \mathbf{y}_{\text{ini}} \\ \mathbf{y} \end{bmatrix}, \\ u_{k|t} \in \mathcal{U}, \quad \forall k \in \mathbb{N}_{[0,N-1]}, \\ y_{k|t} \in (1-\beta_{k|t}) y_i^r \oplus \beta_{k|t} \bar{\mathcal{Y}}(y_i^s, \alpha_i), \quad \forall k \in \mathbb{N}_{[0,N]}, \\ 1 \geq \beta_{0|t} \geq \beta_{1|t} \geq \ldots \geq \beta_{N|t} \geq 0. \end{cases} \tag{11}$$

The only difference between the problem (11) and the problem (10) for the DeePC is the objective function and the last two constraints. We introduce a new sequence of scalar decision variables $\{\beta_{k|t}\}_{k=0}^N$. Similar to the scaling coefficient α_i to define the safe set $\bar{\mathcal{Y}}(y_i^s, \alpha_i)$, $\beta_{k|t}$ plays the same role to scale the active safe set $\bar{\mathcal{Y}}(y_i^s, \alpha_i)$ with respect to the active reference point y_i^r (which lies in $\bar{\mathcal{Y}}(y_i^s, \alpha_i)$ based on Algorithm 3).

More specifically, the problem (11) aims at driving the predicted output trajectory $y_{k|t}$ to the active reference point y_i^r asymptotically. To achieve this, the last two constraints establish a sequence of shrinking active safe sets in the sense that $(1-\beta_{k+1|t}) y_i^r \oplus \beta_{k+1|t} \bar{\mathcal{Y}}(y_i^s, \alpha_i) \subseteq (1-\beta_{k|t}) y_i^r \oplus \beta_{k|t} \bar{\mathcal{Y}}(y_i^s, \alpha_i)$ and enforce the predicted output trajectory $y_{k|t}$ to track these shrinking sets. Given the last monotonicity constraint on $\beta_{k|t}$, the objective function of (11) is to find the optimal scaling coefficients $\beta_{k|t}^*$ such that the last predicted output $y_{N|t}$ is as close to the reference point y_i^r as possible. We remark that the predicted trajectory $y_{k|t}$ is safe in the sense that $\text{dist}(y_{k|t}, \mathcal{O}) > d_{\min}$, in virtue of the property of safe sets we construct in Sec. 4.

Remark 5.2. Note that the quadratic cost function in (10) can be incorporated into the problem (11) to achieve a trade-off between control performance and convergence to the active reference point. For example, the objective function can be replaced by $\min_{g,\mathbf{u},\{\beta_{k|t}\}_{k=0}^N} \rho \beta_{N|t} + \sum_{k=0}^{N-1} [\|y_{k|t} - y_i^r\|_Q^2 + \|u_{k|t}\|_R^2]$ where $\rho > 0$ is the weighting parameter.

5.3. *Robust data-enabled predictive tracking control through regularization*

In practice, it is inevitable to encounter various uncertainties, for example, numerical errors in data collection or disturbances from the environment. Thus, it is of great interest to explore a robust variant of the DeePC that is able to tolerate these uncertainties to some extent. A popular way to achieve this is to incorporate regularization into the optimization problem, see for example [8, 9]. Following the same line, we extend the problem (11) to a new formulation for robust data-enabled predictive tracking control. The new optimization problem, denoted by $\text{ROPT}(y_t, \bar{\mathcal{Y}}(y_i^s, \alpha_i), y_i^r)$, is as follows:

$$\begin{cases} \min\limits_{g,\mathbf{u},\{\beta_{k|t}\}_{k=0}^N,\sigma} \beta_{N|t} + \lambda_g \|g\|_2^2 + \lambda_\sigma \|\sigma\|_2^2 \\ \text{s.t.} \begin{bmatrix} U_p \\ U_f \\ Y_p \\ Y_f \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{ini}} \\ \mathbf{u} \\ \mathbf{y}_{\text{ini}} + \sigma \\ \mathbf{y} \end{bmatrix}, \\ u_{k|t} \in \mathcal{U}, \quad \forall k \in \mathbb{N}_{[0,N-1]}, \\ y_{k|t} \in (1-\beta_{k|t}) y_i^r \oplus \beta_{k|t} \bar{\mathcal{Y}}(y_i^s, \alpha_i), \quad \forall k \in \mathbb{N}_{[0,N]}, \\ 1 \geq \beta_{0|t} \geq \beta_{1|t} \geq \ldots \geq \beta_{N|t} \geq 0. \end{cases} \tag{12}$$

In this problem, the decision variable σ is introduced to ensure that the problem is feasible despite the data noises. Different from the problem (11), the objective function of the problem (12) consists of the new regularization on g and σ. The parameters λ_g and λ_σ are weights. Like the regularization in linear regression, the term $\lambda_g \|g\|_2^2$ enables a trade-off between tracking performance and preventing overfitting [30].

5.4. *Online data-enabled predictive tracking control algorithm*

Based on the new optimization problem (11) for the data-enabled predictive tracking control, we are ready to design an online implementation algorithm, as detailed in Algorithm 4. Algorithm 4 starts from the offline phase which consists of safe set planning as done in Sec. 4 and the data collection. For the online phase, we initialize the active safe set to be $\bar{\mathcal{Y}}(y_0^s, \alpha_0)$ since the initial output is in this set. The while loop terminates until the output y_t reaches the target set $\bar{\mathcal{Y}}_{\text{target}}$. Otherwise, we will check if $y_t \in \bar{\mathcal{Y}}(y_i^s, \alpha_i^s) \cap \bar{\mathcal{Y}}(y_{i+1}^s, \alpha_{i+1}^s)$, which is used to update the active safe set (lines 4–6). Then, we solve the problem OPT$(y_t, \bar{\mathcal{Y}}(y_i^s, \alpha_i), y_i^r)$ in (11) (or ROPT$(y_t, \bar{\mathcal{Y}}(y_i^s, \alpha_i), y_i^r)$ in (12)), whose optimal solution gives the control input for implementation (lines 7–8). Using the online input–output data, we refresh the information for the data-based model (9) (line 9).

Next we apply the data-enabled tracking controller to Example 4.6.

Algorithm 4. Sample-based DeePC for Safe Motion Planning

Offline:

(a) Determine the sequence of safe sets $\mathbb{Y} = \{\bar{\mathcal{Y}}(y_0^s, \alpha_0), \bar{\mathcal{Y}}(y_1^s, \alpha_1), \ldots, \bar{\mathcal{Y}}(y_{N_{\text{set}}-1}^s, \alpha_{N_{\text{set}}-1}), \bar{\mathcal{Y}}_{\text{target}}\}$ and the sequence of reference points $\mathcal{T} = \{y_0^r, y_1^r, \ldots, y_{N_{\text{set}}-1}^r\}$ using Algorithms 2–3;

(b) Collect $\mathbf{u}_{\text{ini}}$, $\mathbf{y}_{\text{ini}}$, $(\hat{u}_{[1,T]}, \hat{y}_{[1,T]})$ with $\hat{u}_{[1,T]}$ persistently exciting of order $T_{\text{ini}} + N + n_x$.

Online:

1: Initialization $i = 0$ and $t = 0$ (note that $y_0 \in \bar{\mathcal{Y}}(y_0^s, \alpha_0^s))$ $y_t \notin \bar{\mathcal{Y}}_{\text{target}}$
2: Measure the output y_t, $y_t \in \bar{\mathcal{Y}}(y_i^s, \alpha_i^s) \cap \bar{\mathcal{Y}}(y_{i+1}^s, \alpha_{i+1}^s)$
3: Let $i \leftarrow i + 1$
4: Solve the problem OPT$(y_t, \bar{\mathcal{Y}}(y_i^s, \alpha_i), y_i^r)$ in (11) (or ROPT$(y_t, \bar{\mathcal{Y}}(y_i^s, \alpha_i), y_i^r)$ in (12)) to obtain the optimal g^* and calculate $\mathbf{u}^* = U_f g^*$
5: Implement the control input $u_t = u_{0|t}^*$
6: Update $(\mathbf{u}_{\text{ini}}, \mathbf{y}_{\text{ini}})$ and let $t \leftarrow t + 1$

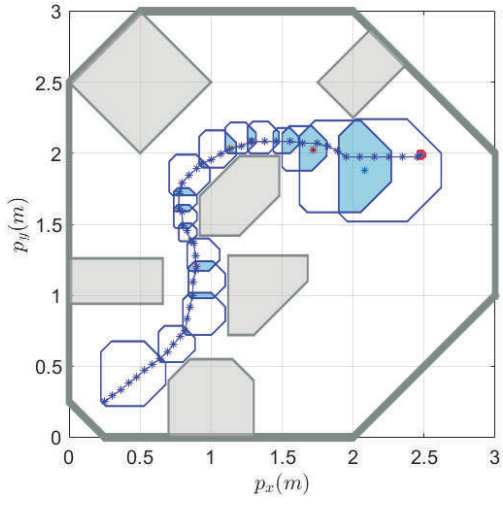


Fig. 7. Tracking control performance under Algorithm 4.

Example 5.3. We implement Algorithm 4. The output trajectory is represented by a sequence of blue asterisks as shown in Fig. 7. The reference points y_i^r are the colored asterisks. The actual position trajectory of the mobile robot under Algorithm 4 is the blue asterisked line. We can see that the mobile robot tracks the sequence of safe sets with respect to the reference points and safely reach the target set, which is the red square in Fig. 7.

6. Numerical Simulations

In this section, we provide several numerical simulations for validating our proposed algorithm. In particular, the advantages of the proposed algorithm in terms of computational efficiency and control performance are highlighted by comparing it with an MIP-based DeePC algorithm. Both simulations are solved by using YALMIP [33] with the solver CasADi [34] version 3.6.2 on an Intel(R) Core(TM) i7-10700kf 3.8 GHz Desktop PC with 64 GB RAM.

6.1. *Safe planning of aerial robot*

We first consider the planning of a quadcopter in a safety-critical environment. The states of the quadcopter are given by the three-dimensional space coordinates (p_x, p_y, p_z) and their velocities, and the three angular coordinates (α, β, γ) and their velocities, which can be represented as $(x, y, z, \dot{x}, \dot{y}, \dot{z}, \alpha, \beta, \gamma, \dot{\alpha}, \dot{\beta}, \dot{\gamma})$. The inputs are given by the thrusts from the four rotors (u_1, u_2, u_3, u_4). As shown in [35], the model of a quadcopter can be approximated by a

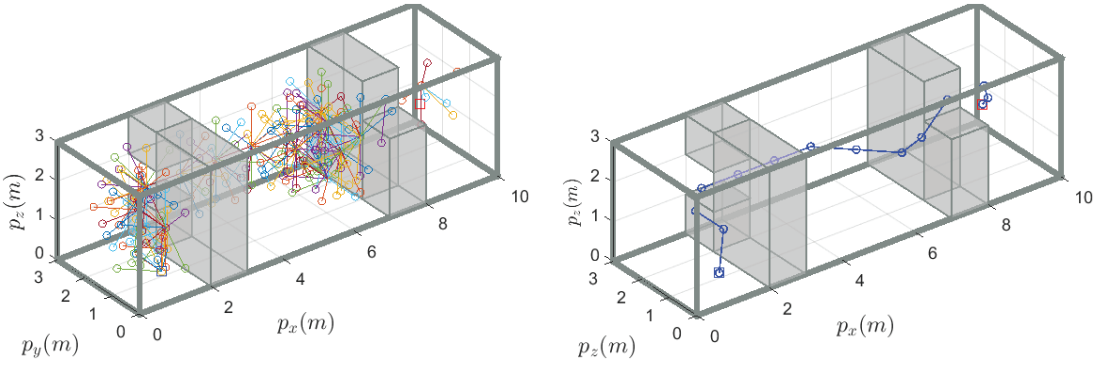
high-fidelity linear model, which motivates us to consider the data-driven planning framework. The control period is set to be 0.01 s. We assume that the system model is unknown and only input and output data are available.

Consider an environment as shown in Fig. 8. The working space $\mathcal{Y}$ is 10 m long, 3 m wide, and 3 m high, which is enclosed by a thick solid gray line. There are two walls, consisting of the obstacles $\mathcal{O}^{(i)}, i = 1, 2, \ldots, 5$, represented by light gray areas. The control task is to steer the robot from the initial point $[1, 0.5, 0.5]^\top$, through small safe holes on the two walls, to a square target set $\bar{\mathcal{Y}}_{\text{target}}$ centered at $(9, 1.5, 1.5)$ with the side length 0.1 m, while ensuring safety, by only using the data. The desired safe distance between the safe sets and the obstacles is set to be 0.02 m.

For the initial step of Algorithm 2, we use A^* algorithm to plan a safe path, which is shown in Fig. 8. The running process of A^* algorithm is shown in Fig. 8(a), where colored line segments represent different exploration directions. Based on this, we distill a safe path that starts from the initial point y_{start} and reaches the target set $\bar{\mathcal{Y}}_{\text{target}}$ while avoiding collision with the obstacles, which is shown in Fig. 8(b).

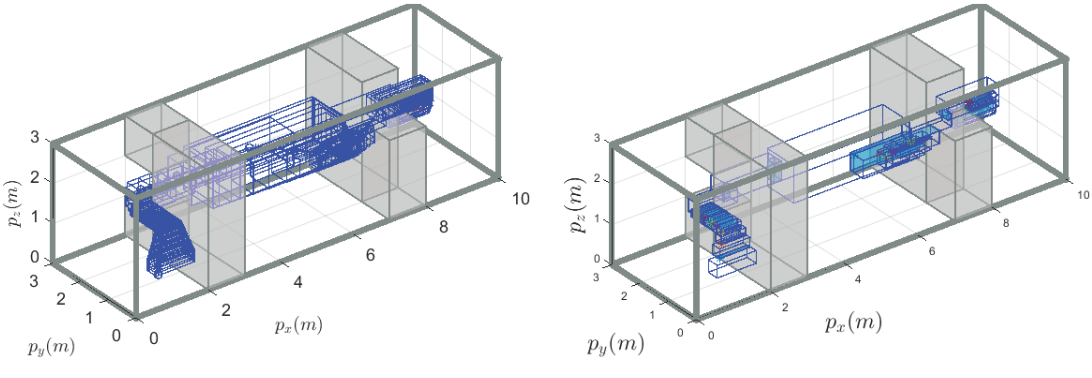
Based on the path from A^* algorithm, the implementation of Algorithm 2 returns a sequence of safe sets, which are the sets circled by the blue line in Fig. 6(a). We can see that the safe sets are nested densely. We further implement Algorithm 3 to remove redundant sets, which leads to a new sequence of safe sets shown in Fig. 9(b). The number of sets in this new sequence is 25, much smaller than 77, the number of sets in the sequence in Fig. 9(a). The sequences in both Figs. 9(a) and 9(b) fulfill the requirements as stated in Problem 3.2. As shown in Fig. 9(b), the intersection area of two adjacent safe sets is colored in light blue, and the reference point sequence is marked as colored asterisks. The total computation time for offline planning is 34.445 s.

For the historical data collection in the offline stage of Algorithm 4, the inputs are sampled uniformly from the input constraint set $\mathcal{U}$ and the output data refers to the system state data. The predictive model in (9) is formulated



(a) (b)

Fig. 8. Initial planning in Algorithm 2 by A^* algorithm: (a) running process and (b) distilled path.



(a) (b)

Fig. 9. (a) A sequence of safe sets generated by Algorithm 2; (b) a simplified sequence of safe sets generated by Algorithm 3.

by using the input and output data $(\hat{u}_{[1,T]}, \hat{y}_{[1,T]})$ of the controlled quadcopter via the Hankel matrix. The parameters of the behavior predictive system model are set as follows: $T_{\text{ini}} = 10$, $N = 10$, $T = 246$, $L = 20$, and $\kappa = 227$. We then implement Algorithm 4. The output trajectory is represented by a sequence of green asterisks as shown in Fig. 10. We can see that the quadcopter tracks the sequence of safe sets with respect to the reference points and safely reach the target set, which is the red square in Fig. 10. The total online calculation time is 0.47 s, the number of control steps is 100, and The average computation time during online tracking is 0.0047 s per time step.

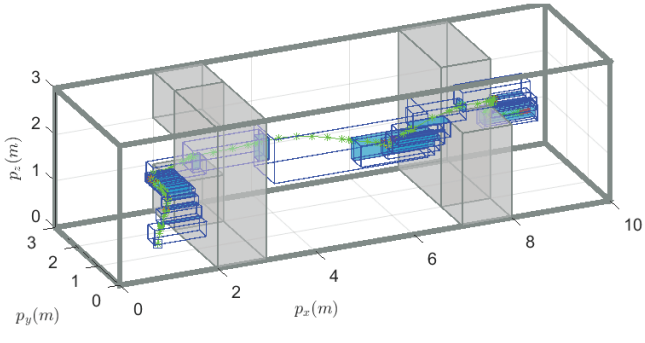
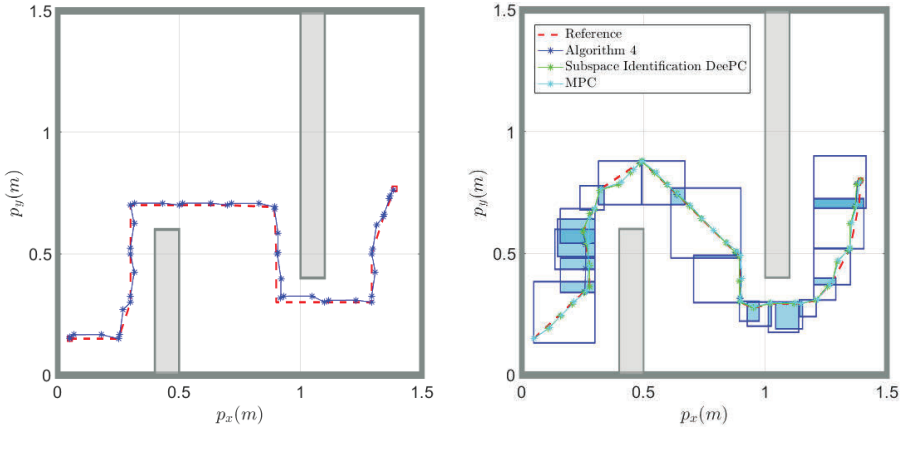


Fig. 10. The tracking performance of UAV under the control of Algorithm 4. The reference points y_i^r are the colored asterisks. The actual position trajectory of the UAV under Algorithm 4 is the green asterisked line.

6.2. *Comparison with MIP-based DeePC, subspace identification DeePC and MPC Algorithm*

In this subsection, we compare our method with the MIP-based DeePC, subspace identification DeePC and MPC Algorithm. In the DeePC algorithm based on MIP, as many literature has done e.g. [20], we use MIP [20] to offline plan a safe path, and then use DeePC for tracking control to compare the path planning capability of the algorithm proposed in this paper. In the DeePC algorithm based on subspace identification [36], the algorithm proposed in this paper is used for path planning, and then the system model is identified from historical data using subspace identification method. The identified model is used instead of the behavioral model combined with the set contraction constraints proposed in this paper to achieve tracking control, thus comparing the performance of the data-driven control algorithm. In the MPC [32] algorithm, the path planned by the algorithm proposed in this paper is used as the reference trajectory, the precise controlled object model is used instead of the behavioral model, without combining the set contraction constraints proposed in this paper for tracking control, thereby comparing the performance of the data-driven tracking control proposed in this paper. As highlighted in [20], the computational complexity of MIP-based method increases exponentially with respect to the number of obstacles. To this end, we consider the same system in the previous subsection but a simple scenario with two obstacles for comparison, which is shown in Fig. 11. The initial point y_{start} is $(0.05, 0.15)$, the target point y_{end} is $(1.4, 0.8)$ and the desired safe distance is 0.1 m.

The red dotted lines in Fig. 11 represent the offline path planning results of the MIP-based method and our method. The blue, green and cyan lines in Fig. 11 depict the online



(a) (b)

Fig. 11. Planning trajectories under (a) MIP-based DeePC and (b) Algorithm 4, subspace identification DeePC and MPC. The red lines are the offline planning paths and the blue, green and cyan lines are the online tracking trajectories under the Algorithm 4, subspace identification DeePC and MPC algorithm.

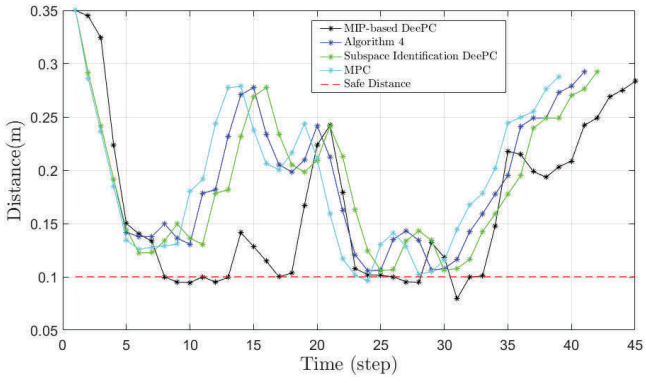


Fig. 12. The minimum distance between the mobile robot and the obstacles at each time step. Black dotted line: MIP-based DeePC algorithm; blue dotted line: Algorithm 4; green dotted line: subspace identification DeePC; cyan dotted line: MPC. The red dotted line denotes $d_{\min} = 0.1$ m.

tracking results using MIP-based DeePC, our method, subspace identification method and MPC method. As shown in Fig. 12, the trajectory using MIP-based DeePC and MPC violate the safe distance constraint, while the trajectory using our method and subspace identification DeePC satisfy the safe distance constraint. That is, the use of DeePC to track a point-based safe path may not come up with a safe trajectory. The safe sets in our method serve as safe barriers when implementing DeePC for tracking. The total computation time (including offline and online phases) of our method is 6.470 s, much shorter than 58.651 s for the MIP-based method and 12.438 s for the subspace identification DeePC, slightly longer than 6.421 s for the MPC algorithm.

To compare the performance of two methods, we introduce the running step t_{end} (i.e. the time of reaching the target set) and the quadratic cost, measured by $J_p = \sum_{t=1}^{t_{\text{end}}} [\|y_t - y_{\text{end}}\|_Q^2 + \|u_t\|_R^2]$, where the weighting matrices in J_p are set to be $Q = \text{diag}\{30, 30\}$ and $R = \text{diag}\{0.5, 0.5\}$. From Table 1, we can see that in comparison with the MIP-based DeePC, our method can complete the control task safely and computationally efficiently with less control cost and shorter running step. Compared with MPC algorithm, our costs and computation time are similar. Compared with the DeePC algorithm based on subspace identification, we have made significant improvements in the offline computation part, slightly lagging behind in online computation efficiency, but having a slight advantage in overall control costs.

7. Experiments

In this section, we apply our method to a four-wheel-mecanum mobile robot in a real-world environment. In this situation, it is inevitable to encounter noisy data collection and thus we implement the robust version of the algorithm.

7.1. *Experiment setup*

The experimental environment is a 3.6 m by 1.5 m rectangle, and there are five obstacles (see Fig. 13). Each obstacle is composed of one or two square blocks with a standard length, width and height of 30 cm. The initial position and the target set are marked by pink square areas.

7.2. *Mobile device*

The four-wheel-mecanum mobile robot with a length of 50 cm, a width of 40 cm and a height of 30 cm used in this experiment is shown in Fig. 13. Each wheel of the mobile robot can be assigned and operated separately, which guarantees the omni-directional movement ability. Four motors are installed and independently driven by IMDR4 motor driving board. The real-time position information is provided by VINS-Fusion method developed in [37]. A ZED2i camera is used to obtain the image of the environment which is essential to apply the positioning method. The image information is transmitted to a Nvidia Jetson TX2 via a USB hub. A VIO terminal is run on TX2 to process the image information and calculate the optimization problem for obtaining the real-time position of the mobile robot, and a control terminal provides the control input, and the experiment data is saved on TX2. STM32F407 is the control board of the low layer which is responsible for executing the control input transmitted from TX2 via RS232. Limited

Table 1. Comparison of computation time, running step and control cost.

Control scheme	MIP-based DeePC	Algorithm 4	Subspace identification DeePC	MPC
Offline computation time (s)	58.536	6.382	12.365	6.382
Online computation time (s)	0.115	0.088	0.073	0.049
Running step t_{end} (step)	44	41	42	39
Online computation time per step (s)	0.0026	0.0022	0.0017	0.0012
Control cost J_p	48.29	45.96	46.35	42.56

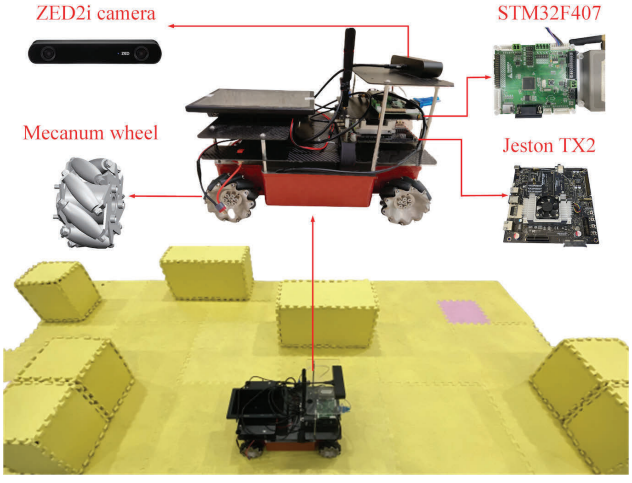


Fig. 13. Four-wheel-mecanum mobile robot in the experiment environment.

by the motor drive capacity, the combined control inputs of the wheeled robot in the x and y axes cannot exceed 1 m/s.

7.3. *Data collection*

Before the experiment, the mobile robot is controlled for 20 s from a randomly selected initial point, by implementing a pre-specified control input sequence which satisfies the persistent excitation condition. With a sampling period of 0.1 s, we obtain an output trajectory. Then the inputs and outputs of the robot are stored in pairs as the historical data. The parameters of the behavior predictive system model are set as follows: $T_{\text{ini}} = 20$, $N = 10$, $T = 200$, $L = 30$, and $\kappa = 171$.

7.4. *Experimental results*

Our objective is to control the robot from the initial point of $(0.1, 0.1)$ to a rectangle area with a length of 0.036 m and a width of 0.015 m centered on $(3.3, 1.2)$. When taking into account the space occupied by the robot, we require that the distance between the center of the robot and the obstacles is no less than 25 cm (half of the length). We implement our data-driven motion planning algorithm to this mobile robot. The results are reported in Fig. 14. The rectangles circled in blue lines are the sequence of safe sets by Algorithms 2 and 3. The blue asterisked line is the position trajectory under the data-enabled predictive tracking controller. We can see that the trajectory is safe in the sense that its distance from the obstacles is greater than 25 cm. The input trajectory is shown in Fig. 15. The first two subfigures plot the inputs v_x and v_y, while the last subfigure shows the evolution of $|v_x| + |v_y|$, where the green line represents the constraint boundary. It can be seen that at all time steps, $|v_x| + |v_y|$ never exceeds the constraint boundary 1. The experimental results verify the computational efficiency of Algorithm 4 and validate that the proposed data-based method is able to ensure the safety of the mobile robot in practice. A video of the experiment can be seen at ExpVideo.

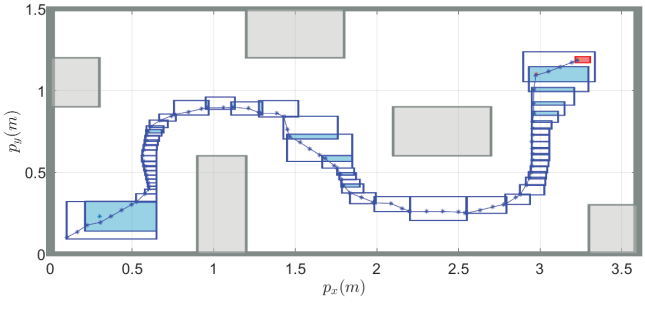


Fig. 14. Position trajectory of the mobile robot under Algorithm 4.

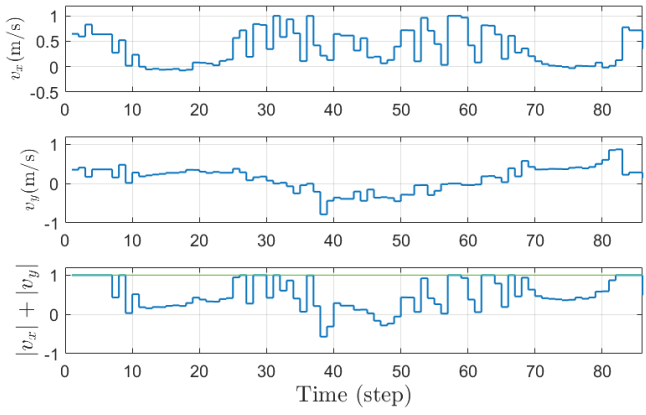


Fig. 15. Evolution of the control inputs and constraint under Algorithm 4.

8. Conclusions

In this paper, we have proposed a novel sample-based DeePC method for safe motion planning of unknown systems. We developed an efficient way to generate a sequence of convex and safe sets from the initial state to the target set and showed that each set can be computed by solving a bilinear optimization problem, which can be solved in a polynomial time complexity for any tolerance. Then we extended the DeePC to the set tracking of unknown systems. That is, by leveraging the behavioral systems theory and a set contraction method, we formulated a sequence of data-driven optimal control problems. Solving each

problem in a receding-horizon manner enables the system to move along the sequence of safe sets. We examined our results through both numerical simulations and practical experiments.

There are a few interesting future research directions. For example, it is of interest to incorporate the occupied space of the dynamical systems into the safety constraint, rather than a mass point as in this paper. Another important extension is safe planning in an environment with moving obstacles.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants 62122014 and 62173036.

ORCID

Li Dai https://orcid.org/0000-0002-7268-7548
Teng Huang https://orcid.org/0000-0002-2062-6137
Yulong Gao https://orcid.org/0000-0003-2338-5487
Sihang Li https://orcid.org/0000-0002-1325-3379
Yunshan Deng https://orcid.org/0000-0002-3311-6080
Yuanqing Xia https://orcid.org/0000-0002-5977-4911

References

[1] J.-C. Latombe, *Robot Motion Planning* (Springer, 2012).
[2] Y. Gao, Safe autonomy under uncertainty: Computation, control, and application, Ph.D. thesis, KTH Royal Institute of Technology (2020).
[3] Y. Gao, F. J. Jiang, L. Xie and K. H. Johansson, Risk-aware optimal control for automated overtaking with safety guarantees, *IEEE Trans. Control Syst. Technol.* **30**(4) (2021) 1460–1472.
[4] Y. Yan, L. Peng, J. Wang, H. Zhang, T. Shen and G. Yin, A hierarchical motion planning system for driving in changing environments: Framework, algorithms, and verifications, *IEEE/ASME Trans. Mechatron.* (2022), doi: 10.1109/TMECH.2022.3219617.
[5] L. Pallottino, E. M. Feron and A. Bicchi, Conflict resolution problems for air traffic management systems solved with mixed integer programming, *IEEE Trans. Intell. Transp. Syst.* **3**(1) (2002) 3–11.
[6] P. Yu and D. V. Dimarogonas, Distributed motion coordination for multirobot systems under LTL specifications, *IEEE Trans. Robot.* **38**(2) (2021) 1047–1062.
[7] A. N. Kopeikin, S. S. Ponda, L. B. Johnson and J. P. How, Dynamic mission planning for communication control in multiple unmanned aircraft teams, *Unmanned Syst.* **1**(1) (2013) 41–58.
[8] J. Coulson, J. Lygeros and F. Dörfler, Data-enabled predictive control: In the shallows of the DeePC, in *2019 18th European Control Conf.* (IEEE, 2019), pp. 307–312.
[9] J. Berberich, J. Köhler, M. A. Müller and F. Allgöwer, Data-driven model predictive control with stability and robustness guarantees, *IEEE Trans. Autom. Control* **66**(4) (2020) 1702–1717.
[10] L. Dai, T. Huang, R. Gao, Y. Zhang and Y. Xia, Cloud-based computational data-enabled predictive control, *IEEE Internet Things J.* **9**(24) (2022) 24949–24962.
[11] S.-F. Wu, J.-S. Mei and P.-Y. Niu, Path guidance and control of a guided wheeled mobile robot, *Control Eng. Pract.* **9**(1) (2001) 97–105.
[12] S. H. Kamil and W. Khaksar, A review on motion planning and obstacle avoidance approaches in dynamic environments, *Adv. Robot. Autom.* **04**(2) (2015) 134–142.
[13] S. J. Russell, *Artificial Intelligence: A Modern Approach* (Pearson Education, 2010).
[14] B.-b. Meng, UAV path planning based on bidirectional sparse A^* search algorithm, in *2010 Int. Conf. Intelligent Computation Technology and Automation*, Vol. 3 (IEEE, 2010), pp. 1106–1109.
[15] S. M. LaValle and J. J. Kuffner, Jr., Randomized kinodynamic planning, *Int. J. Robot. Res.* **20**(5) (2001) 378–400.
[16] L. Jiang, S. Liu, Y. Cui and H. Jiang, Path planning for robotic manipulator in complex multi-obstacle environment based on Improved_RRT, *IEEE/ASME Trans. Mechatron.* **27**(6) (2022) 4774–4785.
[17] S. Karaman and E. Frazzoli, Sampling-based algorithms for optimal motion planning, *Int. J. Robot. Res.* **30**(7) (2011) 846–894.
[18] J. Xu, C. Qian, J.-W. Park and K.-S. Park, Adaptive sampling-based moving obstacle avoidance for cable-driven parallel robots, *IEEE/ASME Trans. Mechatron.* **27**(6) (2022) 4983–4993.
[19] R. Deits and R. Tedrake, Efficient mixed-integer planning for UAVs in cluttered environments, in *2015 IEEE Int. Conf. Robotics and Automation* (IEEE, 2015), pp. 42–49.
[20] M. H. Maia and R. K. Galvão, On the use of mixed-integer linear programming for predictive control with avoidance constraints, *Int. J. Robust Nonlinear Control* **19**(7) (2009) 822–828.
[21] S. V. Raković, S. Zhang, Y. Hao, L. Dai and Y. Xia, Safe polyhedral tubes for locally convexified MPC, *Automatica* **132** (2021) 109791.
[22] A. Kovacs and I. Vajk, Optimization-based model predictive tube control for autonomous ground vehicles with minimal tuning parameters, *Unmanned Syst.* **11**(1) (2023) 93–108.
[23] M. Everett, F. C. Yu and J. P. How, Motion planning among dynamic, decision-making agents with deep reinforcement learning, in *2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (IEEE, 2018), pp. 3052–3059.
[24] Y. Zhu, Z. Wang, C. Chen and D. Dong, Rule-based reinforcement learning for efficient robot navigation with space reduction, *IEEE/ASME Trans. Mechatron.* **27**(2) (2021) 846–857.
[25] S. Veer and A. Majumdar, Probably approximately correct vision-based planning using motion primitives, in *Proc. 2020 Conf. Robot Learning*, Vol. 155 (PMLR, 2021), pp. 1001–1014.
[26] S. Aradi, Survey of deep reinforcement learning for motion planning of autonomous vehicles, *IEEE Trans. Intell. Transp. Syst.* **23**(2) (2022) 740–759.
[27] C. De Persis and P. Tesi, Formulas for data-driven control: Stabilization, optimality, and robustness, *IEEE Trans. Autom. Control* **65**(3) (2019) 909–924.
[28] J. C. Willems, P. Rapisarda, I. Markovsky and B. L. De Moor, A note on persistency of excitation, *Syst. Control Lett.* **54**(4) (2005) 325–329.
[29] J. Berberich and F. Allgöwer, A trajectory-based framework for data-driven system analysis and control, in *2020 European Control Conf.* (IEEE, 2020), pp. 1365–1370.
[30] J. Berberich, J. Köhler, M. A. Müller and F. Allgöwer, Data-driven tracking MPC for changing setpoints, *IFAC-PapersOnLine* **53**(2) (2020) 6923–6930.
[31] Y. Gao, M. Cannon, L. Xie and K. H. Johansson, Invariant cover: Existence, cardinality bounds, and computation, *Automatica* **129** (2021) 109588.
[32] X. Zhang, A. Liniger and F. Borrelli, Optimization-based collision avoidance, *IEEE Trans. Control Syst. Technol.* **29**(3) (2021) 972–983.

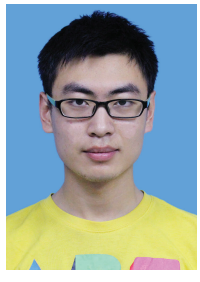
[33] J. Lofberg, YALMIP: A toolbox for modeling and optimization in MATLAB, in *2004 IEEE Int. Conf. Robotics and Automation* (IEEE, 2004), pp. 284–289.
[34] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings and M. Diehl, CasADi: A software framework for nonlinear optimization and optimal control, *Math. Program. Comput.* **11** (2019) 1–36.
[35] R. Mahony, V. Kumar and P. Corke, Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor, *IEEE Robot. Autom. Mag.* **19**(3) (2012) 20–32.
[36] R. Gao, Y. Xia, G. Wang, L. Yang and Y. Zhan, Fast subspace identification method based on containerised cloud workflow processing system, *IEEE Trans. Autom. Sci. Eng.* **33**(6) (2023) 1702–1715.
[37] T. Qin, P. Li and S. Shen, VINS-Mono: A robust and versatile monocular visual-inertial state estimator, *IEEE Trans. Robot.* **34**(4) (2018) 1004–1020.

Li Dai received her B.S. degree in Information and Computing Science in 2010 and her Ph.D. degree in Control Science and Engineering in 2016, both from the Beijing Institute of Technology, Beijing, China. She is currently an Professor in the School of Automation at the Beijing Institute of Technology. Her research interests include model predictive control, distributed control, data-driven control, stochastic systems, and cloud control systems.

Teng Huang received his B.S. degree in Automation in 2018 from the Beijing Institute of Technology, Beijing, China. He is currently a Ph.D. candidate in Control Science and Engineering at the Beijing Institute of Technology. His current research interests include model predictive control, stochastic systems control, motion planning for AGVs, and data-driven control.

Yulong Gao received his B.E. degree in Automation in 2013, his M.E. degree in Control Science and Engineering in 2016, both from Beijing Institute of Technology, and his joint Ph.D. degree in Electrical Engineering in 2021 from KTH Royal Institute of Technology and Nanyang Technological University. He is a Lecturer (Assistant Professor) at the Department of Electrical and Electronic Engineering of the Imperial College London. His research interests include automatic verification, stochastic control, and model predictive control with applications to safety-critical systems.

Sihang Li was born in Hebei, China, in 1996. He received his B.S. degree in Engineering from the Beijing Institute of Technology. He is currently a Ph.D. candidate in Control Science and Engineering at the Beijing Institute of Technology. His research interests include active disturbance rejection control, model predictive control, and motion planning for AGVs.



Yunshan Deng received his B.S. degree in Flight Vehicle Design and Engineering from the Beijing Institute of Technology, Beijing, China, in June 2019, where he is currently working toward a Ph.D. degree in Control Science and Engineering. His research interests include convex optimization and model predictive control.

Yuanqing Xia received his M.Sc. degree in Fundamental Mathematics from Anhui University, China, in 1998, and his Ph.D. degree in Control Theory and Control Engineering from Beijing University of Aeronautics and Astronautics, China, in 2001. He is currently a Professor at the School of Automation, Beijing Institute of Technology, Beijing. His research interests include cloud control systems, networked control systems, robust control and signal processing, active disturbance rejection control, unmanned system control, and flight control.