Fine-Tuned Convex Approximations of Probabilistic Reachable Sets under Data-driven Uncertainties

Pengcheng Wu, Sonia Martinez, Fellow, IEEE, Jun Chen, Member, IEEE

Abstract—This paper proposes a mechanism to fine-tune convex approximations of probabilistic reachable sets (PRS) of uncertain dynamic systems. We consider the case of unbounded uncertainties, for which it may be impossible to find a bounded reachable set of the system. Instead, we turn to find a PRS that bounds system states with high confidence. Our data-driven approach builds on a kernel density estimator (KDE) accelerated by a fast Fourier transform (FFT), which is customized to model the uncertainties and obtain the PRS efficiently. However, the non-convex shape of the PRS can make it impractical for subsequent optimal designs. Motivated by this, we formulate a mixed integer nonlinear programming (MINLP) problem whose solution result is an optimal $n$ sided convex polygon that approximates the PRS. Leveraging this formulation, we propose a heuristic algorithm to find this convex set efficiently while ensuring accuracy. The algorithm is tested on comprehensive case studies that demonstrate its near-optimality, accuracy, efficiency, and robustness. The benefits of this work pave the way for promising applications to safety-critical, real-time motion planning of uncertain dynamic systems.

Note to Practitioners—This study is motivated by the realization of safety-critical real-time motion planning for a dynamic system under uncertainties. A popular method used to guarantee the safe operation of uncertain dynamic systems is reachability analysis. However, this method may not work well in the face of the following challenges: unbounded uncertainties, unknown distributions, generality, convexity, and efficiency. To address these issues, we first present a data-driven approach to model arbitrary unknown uncertainties and obtain a set encompassing the system states with high confidence; Then we propose an algorithm to efficiently find a tight convex polygon approximation for the set. This clearly benefits real motion planning. When considering collision avoidance in motion planning, a tight convex approximation allows a larger feasible search area which may provide a better-planned trajectory. Also, the efficiency of the algorithm ensures that the motion planning can be realized in real-time.

Index Terms—Probabilistic Reachable Set, Convex Approximation, Uncertain Dynamic System, Kernel Density Estimator, Mixed Integer Nonlinear Programming

I. INTRODUCTION

The deployment of autonomous systems in urban ground and air traffic environments requires the development of new real-time collision detection and resolution algorithms [1]–[3]. However, widespread uncertainties in such systems raise important concerns about their safety [4]. Such uncertainties may have various causes, such as epistemic, given a lack of understanding of the underlying mechanism, the gap between simple assumptions and complex reality, or aleatoric, given the inherent randomness in the systems, or the errors of sensor measurements [5]–[8]. A popular method that is used to guarantee the safe operation of uncertain dynamic systems is based on reachability analysis [9]. This set-based method computes the reachable set of states that are accessible by a stochastic dynamic system from all initial conditions and all admissible inputs and parameters [10]. Reachability analysis has increasingly attracted significant attention from researchers in the past decades [11]. In prior literature, accounting for the knowledge of uncertainties, many algorithms have been proposed to utilize detailed system information to figure out a reachable set. However, existing algorithms may not work well or even fail in the face of the following challenges:

1) Unboundedness: When uncertainties are unbounded, it may be impossible to figure out a bounded reachable set for an uncertain dynamic system. However, most real-world applications need a bounded reachable set due to physical limits.

2) Unknown Distributions: Uncertainties are often assumed to obey Gaussian distributions in most literature. However, in practice, the uncertainties may be non-Gaussian or even empirical due to unknown system interconnections and nonlinearity [12], [13].

3) Generality: To compute reachable sets, many algorithms require detailed system information given a priori. However, in some applications like complex cyber-physical systems that are only accessible through experiments or simulations, this detailed information is unavailable. Many algorithms also rely on strong assumptions about the dynamics or uncertainties, which are often unrealistic.

4) Convexity: The reachable set computed by most algorithms is irregular or non-convex, which is hard to handle when it comes to the resolution of collision avoidance in practice.

5) Efficiency: In order to realize real-time motion planning and control of uncertain dynamic systems, the reachable set must be computed efficiently. However, many algorithms are computationally expensive.

To address the challenges, in what follows, we will first review some related literature, and then introduce the contributions of our paper.
A. Literature Review

A classical way to compute reachable sets for bounded uncertainties is given by performing a Hamilton-Jacobi (HJ) reachability analysis. It formulates the problem as a partial differential equation which can be solved by numerical methods. While computationally tractable in lower dimensions, the method scales poorly to higher dimensions [1], [14]. Also, HJ reachability analysis requires detailed information on dynamics or uncertainties and relies on strong assumptions like there are no time correlations of parameter uncertainties, which makes it unavailable in some realistic scenarios [1].

When uncertainties are unbounded, it is often impossible to guarantee collision avoidance with 100% certainty as the reachable set becomes unbounded. Instead, researchers seek to identify a trajectory with a probability of collision bounded by a certain risk bound. One direct way to realize this goal is to evaluate the probability of collision of a generated candidate trajectory. The candidate trajectory will not be adopted until the probability of collision is below the risk bound. However, there is no closed-form expression to evaluate the probability of collision, which poses difficulties to this approach [15], [16]. Another indirect but effective method consists of computing a probabilistic reachable set (PRS) instead of an exact one [17], [18]. A PRS is a set of possible states that a system can reach with a certain probability. By means of this, the stochastic constraint on the system, which sets a limit on the collision probability, can be converted to a deterministic constraint by which the generated trajectory does not intersect with the PRS [19]. An advantage of this method is that it is not only applicable for unbounded uncertainties but also for bounded ones. As the risk bound approaches 0%, the PRS becomes a superset of the reachable set [17]. In this paper, we adopt the PRS approach.

Many existing works assume a Gaussian distribution of the uncertainty [20], [21]. Blackmore et al. [19] use convex polygons (or polytopes) determined by the mean and covariance of the Gaussian distribution as a proxy of PRS. Wu and Chen et al. [4], [6] directly use the level sets of the Gaussian distribution to serve as PRS. However, in many scenarios uncertainties are of non-Gaussian type [12]. Even in that case, some researchers still make use of Gaussian distributions to approximate non-Gaussian ones [22]. Unfortunately, this approach can not guarantee that the risk of a generated trajectory is still within a given bound [13]. Instead, Aoude et al. [23] apply a particle method to evaluate the probability of collision in the process of motion planning. Calafiore and Campi [24] use a scenario approach by which stochastic constraints are sampled to obtain a convex optimization problem whose solution is feasible for the original stochastic constraints with high confidence. However, all of the above typically require a large number of samples for probabilistic assurance, which is neither feasible nor computationally efficient. Alternatively, Han and Jasour et al. [13] employ a moment-based method to characterize non-Gaussian PRS. However, moment information is assumed to be known a priori in their works, which is not realistic as this information can only be learned online from sensor measurements [25]. Moreover, the PRS is often irregular, which is challenging to handle in practice.

Such drawbacks motivate the study of data-driven reachability analysis, using data obtained from experiments or simulations [25]–[27]. Devonport et al. [11] use the level sets of a Christoffel function to estimate the reachable sets. The Christoffel function is obtained from the samples of the arbitrary unknown uncertainties, without requiring any detailed information a priori. However, the level sets estimating the reachable sets may be irregular or non-convex, which is hard to deal with in practice. Motivated by this, Lew et al. [1] employ a conservative convex hull of data samples for reachable set approximation. The number of sides of the convex hull is unspecified, leading to an arbitrary number of constraints in subsequent design problems. A bounding box can enclose data samples with a fixed number of sides but is more conservative [28]. All the works above are intended to estimate reachable sets given bounded uncertainties and are not accurate approximations of the reachable sets because lacking uncertainty quantification. For arbitrary probability distributions, including unbounded ones, a natural way of approximating reachable sets is via empirical or data-fitted probability density functions (PDF), whose level sets serve as the reachable sets. In this paper, we employ this approach considering kernel density estimators (KDE) [29]. Our goal is to address important limitations of adopting such a method such as the lack of closed-form expressions for KDEs, the irregular shapes of their level sets, and the conservativeness and inaccuracy of taking convex hulls or bounding boxes of the level sets. Some researchers have considered dynamic systems with time delays [30], which is not done in this manuscript. The extension of the proposed methodology to scenarios where time delay matters is left for future work.

B. Contributions and Organization

This paper proposes an efficient convex approximation of PRS of uncertain dynamic systems while ensuring their accuracy. Our approach consists of two parts: (i) A data-driven uncertainty quantification; (ii) An optimization problem formulation to find the convex approximation.

The major contributions of this paper are the following:

1) We present an algorithm that efficiently finds the PRS via KDE accelerated by a fast Fourier transform (FFT). FFT-based KDE is customized to estimate the PDF under arbitrary unknown uncertainties and therefore its level sets can serve as a proxy for the PRS of the uncertainties. As an efficient data-driven approach, FFT-based KDE can learn from data online, and thus it neither requires a priori information nor relies on strong assumptions;

2) We formulate an optimization problem of mixed integer nonlinear programming (MINLP), resulting in an optimal $n$ sided convex polygon that approximates the PRS. As opposed to a convex hull, users can arbitrarily customize the number of sides. The optimization problem is established not by using data samples directly but by weighted grid points from KDE, which makes it tractable. Compared with bounding boxes, the result obtained by this approach is more tight and accurate;

3) We develop a heuristic algorithm to efficiently solve the formulated MINLP problem. This algorithm performs
weighted sampling to select the representative grid points from all the grid points from KDE, which significantly reduces computational complexity. Comprehensive case studies demonstrate that this algorithm enjoys the benefits of accuracy, efficiency, near-optimality, and robustness while providing a convex approximation for the PRS.

The rest of this paper is organized as follows. In Section II, we formally define the concept of PRS and state the problem as providing a convex approximation for it. In Section III, we use KDE accelerated by FFT to model arbitrary probability distributions and present an algorithm to find the PRS. In Section IV, we formulate an MINLP optimization framework, the solution of which leads to a convex polygon approximation of the PRS. In Section V, we develop a heuristic algorithm to efficiently solve the formulated MINLP. In Section VI, we conduct comprehensive case studies to demonstrate the performance of our proposed algorithm. In Section VII, we draw conclusions and make suggestions for future research. In addition, rigorous proofs of some propositions in this paper are given in Section VIII Appendix.

II. Problem Statement

In this section, we take into account an uncertain discrete-time dynamic system and introduce the concept of reachable set and PRS of the system. Then, we present the goal of this paper.

Consider a general discrete-time dynamic system of the form

$$ x_{k+1} = f(x_k, u_k, \theta_k, w_k), $$

where $x_k \in \mathbb{R}^n$ is the system state, $u_k \in \mathcal{U}_k$ is an uncertain control input, $\theta_k \in \Theta$ is an uncertain parameter, $w_k \in \mathcal{W}$ is an uncertain disturbance, and $x_0 \in \mathcal{X}_0$ is an initial state. Unlike most prior works [1], in this paper, the sets $\mathcal{U}_k \subset \mathbb{R}^m, \Theta \subset \mathbb{R}^p, \mathcal{W} \subset \mathbb{R}^q$, and $\mathcal{X}_0 \subset \mathbb{R}^n$ can be unbounded. Given the uncertainties of $x_k, u_k, \theta_k, w_k$, then $x_{k+1}$ is a random vector which obeys an unknown probability distribution.

Formally, the concept of reachable set is introduced as follows.

**Definition 1** (Reachable Set [31]). At time $T$, the reachable set $\mathcal{X}_T$ of the dynamic system in Eq. (1) is defined to be

$$ \mathcal{X}_T := \{ x_T \in \mathbb{R}^n : x_k = f(x_{k-1}, u_{k-1}, \theta_{k-1}, w_{k-1}), \forall k \in \{1, \ldots, T\} \}, $$

where $x_0$ is the initial state and $\mathcal{X}_0$ is the initial set.

Observe that, since the sets $\mathcal{U}_{k-1}, \Theta, \mathcal{W}$, and $\mathcal{X}_0$ are not limited to bounded sets, the reachable set $\mathcal{X}_T$ may be unbounded. Hence, it may be infeasible to find a bounded set in which the state is guaranteed to lie. Instead, we expect to find a bounded set such that the probability of the state lying in the bounded set is greater than a confidence level. To this end, we formally introduce the concept of the PRS as follows.

**Definition 2** (Probabilistic Reachable Set (PRS) [18]). At time $k$, a bounded set $\tilde{\mathcal{X}}_k$ is defined to be a probabilistic reachable set (PRS) of the dynamic system in Eq. (1) at confidence level $\alpha$ if and only if

$$ \Pr(x_k \in \tilde{\mathcal{X}}_k) \geq \alpha. $$

A PRS is a set of possible states that a system can reach with a certain probability. Note that $\tilde{\mathcal{X}}_k$ may not exist. When $\alpha = 100\%$, if there exists a PRS $\tilde{\mathcal{X}}_k$, then $\tilde{\mathcal{X}}_k$ is a superset of the reachable set.

In this paper, we assume the dynamic system in Eq. (1) is two-dimensional, and leave the general dimension case for future work. The goal of this paper is twofold: 1) We aim to put forward a data-driven approach to model the arbitrary unknown uncertainties and obtain a PRS $\tilde{\mathcal{X}}_k$ of the dynamic system; 2) Motivated by the realization of safety-critical real-time motion planning for uncertain systems, we aim to approximate the PRS $\tilde{\mathcal{X}}_k$ by means of a bounded set that satisfies: i) Convexity: The boundary of the set is an $n$-sided convex polygon; ii) Efficiency: The computation of this set is tractable; iii) Accuracy: The probability of the state $x_k$ lying in the set is close to the prescribed confidence level $\alpha$; iv) Optimality: The area of the set is as small as possible (not too conservative) while ensuring accuracy.

III. PRS Identification

In this section, we model an arbitrary unknown probability distribution through FFT-based KDE, and develop an online algorithm to find a PRS at confidence level $\alpha$.

**A. FFT-Based KDE**

We first briefly review FFT-based KDE. The concept of KDE is introduced below.

**Definition 3** (Kernel Density Estimator (KDE) [29]). Let $x_k \in \mathbb{R}^2, k \in \{1, \ldots, M\}$ be $M$ data samples drawn from a 2-variate probability distribution given by a PDF $f: \mathbb{R}^2 \mapsto \mathbb{R}$. The kernel density estimator (KDE) $\hat{f}: \mathbb{R}^2 \mapsto \mathbb{R}$ to approximate the PDF $f$ is defined to be

$$ \hat{f}(x) = \frac{1}{M} \sum_{k=1}^{M} K(x - x_k), $$

where the function $K: \mathbb{R}^2 \mapsto \mathbb{R}$ is the kernel function which is a symmetric 2-variate density function.

The choice of the kernel function $K$ is not critical to the accuracy of KDE, so we use the standard 2-variate Gaussian kernel in this paper

$$ K(x) = (2\pi)^{-1/2} \det(H)^{-1/2} \exp(-\frac{1}{2}x^\top H^{-1}x), $$

where $H$ is a symmetric and positive definite bandwidth matrix. The choice of bandwidth matrix is crucial to the accuracy of KDE approximating PDF. In this paper, we choose the bandwidth matrix using Silverman’s rule [32].

Consider a mesh of $N^2$ grid points $\{(x_i, y_j) \in \mathbb{R}^2 : i, j \in \{1, \ldots, N\}\}$ equally spaced on the plane $\mathbb{R}^2$, the boundary of which encompasses all $M$ data samples $x_k \in \mathbb{R}^2, k \in \{1, \ldots, M\}$. The mesh of grid points can be viewed as a map
\( g : \{(i, j) : i, j \in \{1, \ldots, N\}\} \mapsto \{(x_i, y_j) \in \mathbb{R}^2 : i, j \in \{1, \ldots, N\}\} \) given by
\[
(x_i, y_j) = g(i, j), \ i, j \in \{1, \ldots, N\},
\]
where \((i, j)\) is the index of a grid point whose coordinate is \((x_i, y_j)\). Throughout the rest of this paper, a coordinate \( g(i, j) \) is abbreviated as \( g_{ij} \).

The KDE \( \hat{f} \) given in Definition 3 can be evaluated on the mesh of \( N^2 \) grid points \( g \) in Eq. (3), which yields
\[
\hat{f}_{ij} := \hat{f}(g_{ij}) = \frac{1}{M} \sum_{k=1}^{M} K(g_{ij} - x_k), \ i, j \in \{1, \ldots, N\}.
\]

One way to significantly accelerate the evaluation of KDE is linear binning. Instead of placing the kernel functions \( K \) on \( M \) data samples, we can place them on \( N^2 \) grid points \( g \) weighted by grid counts \( c_{uv} \), where \( u, v \in \{1, \ldots, N\} \). A grid count \( c_{uv} \) is a weight chosen to represent the amount of data samples near a grid point \( g_{uv} \). To obtain \( c_{uv} \), we can go through every data sample and assign weights to its neighbouring grid points, as shown in Fig. 1.

![Graphical representation of 2-variate linear binning](image)

Fig. 1. Graphical representation of 2-variate linear binning

Using the strategy of linear binning, the binned KDE \( \tilde{f} : \mathbb{R}^2 \mapsto \mathbb{R} \) to approximate the KDE \( \hat{f} \) in Definition 3 is given by
\[
\tilde{f}(x) = \frac{1}{M} \sum_{u=1}^{N} \sum_{v=1}^{N} K(x - g_{uv})c_{uv},
\]
where \( c_{uv} \) is the grid count of the grid point \( g_{uv} \) indexed by \((u, v)\).

Evaluating the binned KDE \( \tilde{f} \) in Eq. (4) on the mesh of \( N^2 \) grid points \( g \) yields
\[
\tilde{f}_{ij} := \tilde{f}(g_{ij}) = \frac{1}{M} \sum_{u=1}^{N} \sum_{v=1}^{N} K(g_{ij} - g_{uv})c_{uv},
\]
\[i, j \in \{1, \ldots, N\},
\]
which shows that the matrix \( \tilde{f} := [\tilde{f}_{ij}] \) is the discrete convolution of \( c := [c_{uv}] \) and \( k := [K(g_{uv})] \).

The discrete convolution can be computed by FFT, which significantly saves computational time. Let \( C \) and \( K \) be FFT of \( c \) and \( k \), and \( \tilde{F} \) be the element-wise product of \( C \) and \( K \). Then \( \tilde{f} \) can be extracted from the inverse FFT of \( \tilde{F} \). By doing so, we can obtain the binned KDE \( \tilde{f}_{ij}, i, j \in \{1, \ldots, N\} \) in Eq. (5) to approximate the PDF \( f \) of an arbitrary unknown probability distribution in real-time [15], [33].

### B. Online Computation of PRS

For real-time motion planning of uncertain dynamic systems, we propose an efficient data-driven algorithm using FFT-based KDE to capture PRS of an unknown static uncertainty whose data samples are gradually observed. See Algorithm 1.

In the pseudo-code of Algorithm 1, the symbol \( \epsilon \) represents the tolerance of confidence error. From Line 10 through 17, FFT is employed to accelerate the computation of the discrete convolution of \( c \) and \( k \) to obtain the binned KDE \( \tilde{f} \) evaluated on the mesh of \( N^2 \) grid points \( g \). From Line 18 through 33, a bisection search is implemented to find the critical binned KDE value \( C^{kde} \) at which the confidence level \( \alpha \) is achieved. The level set of the KDE corresponding to \( C^{kde} \) is the PRS that we aim to find.

As the number of the collected data samples increases, KDE captured from those data samples will be convergent to the true PDF of the uncertain uncertainties. However, considering more data samples requires a longer evaluation time. Fig. 2 shows how increasing data samples influence the evaluation time of identifying a PRS at confidence level 95\% on a mesh of \( 128^2 \) grid points through implementing Algorithm 1. As the number of data samples increases, the proposed Algorithm 1 can rapidly recompute. For example, when the number of data samples is within the range of \( 10^3 \) to \( 10^4 \), the evaluation time of implementing Algorithm 1 is always under 0.01 s. This result demonstrates the computational power of our proposed Algorithm 1 and thus it can handle streaming data in a reasonable way. However, it cannot guarantee to deal with uncertainties that can evolve with time, and this will be part of our future work.

Moreover, note that the shape of a PRS obtained in this way is often irregular, which precludes its use in subsequent applications requiring an optimal design. This will be solved in the next section.

![Evaluation time of Algorithm 1 with respect to the increasing number of data samples](image)

Fig. 2. Evaluation time of Algorithm 1 with respect to the increasing number of data samples

### IV. MINLP FORMULATION FOR PRS APPROXIMATION

In this section, we formulate the convex approximation problem stated in Section II as an MINLP optimization problem for the PRS of a two-dimensional system subject to arbitrary uncertainties.
Algorithm 1 PRS Identification Algorithm

1: function GENDS($M$)
2: Generate $M$ data samples $x_k, k \in \{1, \ldots, M\}$
3: return $x_k$
4: function MESHGRID($x_k, N$)
5: Get $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ from data samples $x_k$
6: $h = \text{linspace}(x_{\min}, x_{\max}, N)$
7: $v = \text{linspace}(y_{\min}, y_{\max}, N)$
8: $g = \text{Cartesian product of } h \text{ and } v$
9: return $g$
10: function FFTKDE($x_k, g$)
11: Obtain grid counts $c$ for all grid points $g$ using $x_k$
12: Evaluate kernel functions $K$ on all grid points $g$
13: $C = \text{FFT}(c)$
14: $K = \text{FFT}(k)$
15: $F = \text{the element-wise product of } C \text{ and } K$
16: $f = \text{iFFT}(F)$
17: return $f$
18: function BISECSEARCH($\tilde{f}, \alpha, \epsilon$)
19: $\text{low} = \text{min}(\tilde{f})$
20: $\text{up} = \text{max}(\tilde{f})$
21: while $\text{low} < \text{up}$ do
22:  $\text{mid} = (\text{low} + \text{up})/2$
23:  $z_{\text{bin}} = \{f \geq \text{mid}\}$
24:  $z_{\text{min}} = \text{element-wise product of } z_{\text{bin}} \text{ and } \tilde{f}$
25:  $pr = \text{sum}(z_{\text{min}}) / \text{sum}(\tilde{f})$
26:  if $\text{abs}(pr - \alpha) \leq \epsilon$ then
27:      $C_{\text{kde}} = \text{mid}$
28:      return $z_{\text{bin}}, C_{\text{kde}}$
29:  else if $pr < \alpha$ then
30:      $\text{up} = \text{mid}$
31:  else
32:      $\text{low} = \text{mid}$
33: return Failure
34: function FINDPRS($M, N, \alpha, \epsilon$)
35: $x_k = \text{GENDS}(M)$
36: $g = \text{MESHGRID}(x_k, N)$
37: $f = \text{FFTKDE}(x_k, g)$
38: $z_{\text{bin}}, C_{\text{kde}} = \text{BISECSEARCH}(\tilde{f}, \alpha, \epsilon)$
39: return $g, f, z_{\text{bin}}, C_{\text{kde}}$
40: FINDPRS($M, N, \alpha, \epsilon$)

Recall that, after evaluating Eq. (5) through the implementation of Algorithm 1, we have obtained that each grid point $g_{ij}, i, j \in \{1, \ldots, N\}$ has a binned KDE value $\tilde{f}_{ij}, i, j \in \{1, \ldots, N\}$, respectively. We define a normalized weight matrix $w := [w_{ij}]$ whose element is

$$w_{ij} := \frac{\tilde{f}_{ij}}{\sum_{i=1}^{N} \sum_{j=1}^{N} \tilde{f}_{ij}}, \quad i, j \in \{1, \ldots, N\}. \quad (6)$$

The MINLP will be established using these weights, and not with the original data samples. This makes it more tractable because the number of grid points is far less than the number of data samples, significantly reducing the number of decision variables and constraints.

A. Problem Description and Objective Function Formulation

The optimization goal is to find a minimum $n$ sided convex polygon to approximate the PRS efficiently and accurately, by determining which weighted grid points should lie inside the polygon.

To this end, we introduce three types of decision variables to formulate the MINLP optimization problem:

- $2n$ continuous variables $a_k, b_k \in \mathbb{R}, k \in \{1, \ldots, n\}$;
- $N^2 n$ binary variables $l_{ij}^k \in \{0, 1\}, i, j \in \{1, \ldots, N\}, k \in \{1, \ldots, n\}$;
- $N^2$ binary variables $z_{ij} \in \{0, 1\}, i, j \in \{1, \ldots, N\}$.

Here the integer $n \geq 3$ is the number of convex polygon edges; The integer $N$ is the number of grid points in each dimension; An ordered pair of real numbers $(a_k, b_k)$ indicates the coefficients of a line $l^k$ which is the extension of an edge of the convex polygon; The binary variable $l_{ij}^k = 1$ if and only if the grid point $g_{ij}$, indexed by $(i, j)$, lies on the specified side of the line $l^k$; The binary variable $z_{ij} = 1$ if and only if the grid point $g_{ij}$ lies inside the polygon. We will fully discuss the roles of these decision variables in what follows.

Our goal is to make the convex polygon as small as possible without violating the constraints. This can be realized by minimizing the number of grid points lying inside the polygon. Formally speaking, our optimization objective function can be formulated as

$$\text{min} \sum_{i=1}^{N} \sum_{j=1}^{N} z_{ij}, \quad (7)$$

where $\sum_{i=1}^{N} \sum_{j=1}^{N} z_{ij}$ is a linear function of binary variables $z_{ij}$.

B. Constraints on Continuous Variables

All the planar lines not passing through the origin can be represented by

$$\{ax + by - 1 = 0 : a, b \in \mathbb{R} \cap a^2 + b^2 \neq 0\},$$

and different pairs of coefficients $(a, b)$ determine different lines.

A line $l^k := a_k x + b_k y - 1 = 0$ always divides the entire plane into two half-planes. We refer to the half plane $\{(x, y) : a_k x + b_k y - 1 < 0\}$ where the origin $(0, 0)$ lies as the strict inner side of the line, and $\{(x, y) : a_k x + b_k y - 1 \leq 0\}$ as the inner side of the line. A strict inner side is a proper subset of the inner side of the same line.

For any $n \geq 3$ planar lines not passing through the origin, the region $S$ bounded by these lines is defined as

$$S := \{(x, y) : \bigwedge_{k=1}^{n} a_k x + b_k y - 1 \leq 0\}, \quad (8)$$
where $a_k, b_k \in \mathbb{R}$, $a_k^2 + b_k^2 \neq 0$, and $n \geq 3$. On the one hand, since $S$ is the intersection of $n$ inner sides, then $S$ is a convex set and the origin $(0, 0) \in S$; On the other hand, the region $S$ may be not a bounded set (In other words, the boundary of $S$ may be not enclosing). Even though bounded, the boundary of $S$ may be a polygon with the number of edges less than $n$, or even not a polygon but an object consisting of line segments or a point only.

We look for conditions that ensure $S$ is a bounded $n$ sided convex polygon with the origin in its interior. To do this, we first introduce the following concepts.

**Definition 4 (Digraph [34]).** A finite directed graph (in short, a digraph) is a 2-tuple $(V, E)$, where $V \subseteq \mathbb{R}^2$ is a finite set of planar points called nodes and $E \subseteq V \times V$ is a set of ordered pairs of nodes called (directed) edges. Self-loops are allowed in a digraph. For $u, v \in V$, if the ordered pair $(u, v) \in E$, it denotes an edge from $u$ to $v$.

**Definition 5 (Path [34]).** A finite directed path (in short, a path) given by a finite sequence of planar points $S_g := (S_g[0], \ldots, S_g[m])$ where $S_g[i] \in \mathbb{R}^2, i \in \{0, \ldots, m\}$, is a digraph whose set of nodes $V = \{S_g[i] : i \in \{0, \ldots, m\}\}$ and set of edges $E = \{(S_g[i], S_g[i+1]) : i \in \{0, \ldots, m-1\}\}$. In a path $G$, one node $S_g[i]$ immediately precedes another node $S_g[j]$ if and only if there is an edge $(S_g[i], S_g[j])$; Two nodes are consecutive if and only if one immediately precedes the other; Three nodes are consecutive if and only if the first one immediately precedes the second and the second immediately precedes the third.

Note that if $S_g[i] = S_g[j], i \neq j$, then $\{S_g[i], S_g[j]\} = \{S_g[i]\}$, according to the Axiom of Extensionality in axiomatic set theory [35]. For a node $S_g[i]$ in $S_g$, the node which $S_g[i]$ immediately precedes (resp. which immediately precedes $S_g[i]$) may not exist, and even if it exists, it may be not unique. If there is an edge $(S_g[i], S_g[i])$, then $S_g[i]$ immediately precedes itself. It is possible that $S_g[i]$ immediately precedes $S_g[j]$ and $S_g[j]$ immediately precedes $S_g[i]$ at the same time if $(S_g[i], S_g[j])$ and $(S_g[j], S_g[i])$ are both edges in $S_g$.

**Definition 6 (Graph and Undirected Version of Digraph [34]).** A finite undirected graph (in short, a graph) is a 2-tuple $(V, E)$, where $V \subseteq \mathbb{R}^2$ is a finite set of planar points called nodes and $E$ is a set of unordered pairs of nodes called (undirected) edges. Self-loops are not allowed in a graph. For $u, v \in V$, $u \neq v$, if the set $\{u, v\} \in E$, it denotes an edge between $u$ and $v$. A graph $(V, E)$ is the undirected version of a digraph $(V, E')$ if and only if $(u, v) \in E' \iff \{u, v\} \in E$ for $u, v \in V, u \neq v$.

In this paper, these definitions emphasize the planar geometric properties. For example, the positions of nodes are 2D coordinates; The shape of an edge connecting two distinct nodes is a straight line segment and the shape of a self-loop is a point. The operation of taking undirected version preserves geometric properties. For example, for any three nodes collinear in a digraph, these nodes must also be collinear in the graph which is the undirected version of the digraph.

**Definition 7 (Enclosing Path).** A path $S_g$ is enclosing if and only if: 1) There are $n \geq 3$ different nodes; 2) No nodes appear more than once in the sequence $S_g$ except for the case of the initial and final node, which should be identical; 3) For any edge connecting two nodes, no other nodes lie on that edge.

For example, all the paths in Fig. 3 are not enclosing. The path $(1, 2, 3, 4, 2, 1)$ in Fig. 3a violates Condition 2) in Definition 7 because Node 2 appears more than once; The path $(1, 2, 3, 4)$ in Fig. 3b also violates Condition 2) because the first node 1 and last node 4 are not identical; The path $(1, 2, 3, 4, 1)$ in Fig. 3c satisfies first two conditions, but violates Condition 3) because Node 4 is on the edge $(1, 2)$.

Through the definition of an enclosing path, we can introduce a definition of an enclosing graph.

**Definition 8 (Enclosing Graph).** A graph $G$ is enclosing if and only if there is an enclosing path $S_g$ whose undirected version is $G$.

For example, the graph $G$ in Fig. 4 is not enclosing because any path whose undirected version is $G$ is not an enclosing path.

In this paper, these definitions emphasize the planar geometric properties. For example, the positions of nodes are 2D coordinates; The shape of an edge connecting two distinct nodes is a straight line segment and the shape of a self-loop is a point. The operation of taking undirected version preserves geometric properties. For example, for any three nodes collinear in a digraph, these nodes must also be collinear in the graph which is the undirected version of the digraph.

It is straightforward to see that if a path $S_g$ is enclosing, then its undirected version $G$ is also enclosing. Conversely, if a path is not enclosing, its undirected version may still be enclosing.

**Definition 9 (Formally Enclosing Sequence of Symbols).** Let $S_s$ be a finite sequence of symbols. The sequence $S_s$ is formally enclosing if and only if: 1) There are $n \geq 3$ formally different symbols; 2) No symbols appear more than once except for the case in which the initial and final symbol are formally identical.

In a finite sequence of symbols $S_s$, one symbol immediately precedes another symbol if and only if there is an appearance $S_s[i]$ of the first one and an appearance $S_s[j]$ of the second such that $j = i + 1$; Two symbols are consecutive if and only if one immediately precedes the other; Three symbols are...
Definition 10 (Isomorphism Type A). Let \( f_{\text{tag}} \) be a map from a finite sequence of symbols \( S_s \) to a path \( S_g \). The map \( f_{\text{tag}} \) is an isomorphism type A if and only if the following properties hold for \( f_{\text{tag}} \): 1) **Injective**: formally different symbols in \( S_s \) refer to different nodes in \( S_g \); 2) **Surjective**: every node in \( S_g \) is the image of a symbol in \( S_s \); 3) **Order-Preserving**: for any two symbols \( S_s[i], S_s[j] \) in \( S_s \), the symbol \( S_s[i] \) immediately precedes \( S_s[j] \) if and only if \( (f_{\text{tag}}(S_s[i]), f_{\text{tag}}(S_s[j])) \) is an edge in \( S_g \); 4) **Not on the edge**: for any two consecutive symbols \( S_s[i], S_s[i+1] \) and any third symbol \( S_s[k] \) in \( S_s \), if \( S_s[k] \) is formally different from \( S_s[i] \) and \( S_s[i+1] \), then \( f_{\text{tag}}(S_s[k]) \) is not on the edge \( (f_{\text{tag}}(S_s[i]), f_{\text{tag}}(S_s[i+1])) \) in \( S_g \).

Remark 1. Observe that if there is an isomorphism type A \( f_{\text{tag}} \) from a finite sequence of symbols \( S_s \) to a path \( S_g \), then \( S_s \) is formally enclosing if and only if \( S_g \) is enclosing. We have seen that if \( S_g \) is enclosing, then its undirected version \( G \) is also enclosing. Therefore, we conclude that if \( f_{\text{tag}} \) is an isomorphism type A from \( S_s \) to \( S_g \), and \( S_s \) is formally enclosing, then the graph \( G \), as the undirected version of \( S_g \), is also enclosing.

In addition to the discussions of enclosing paths and graphs above, we would also like to explore under which conditions it suffices to guarantee non-degenerate paths and graphs.

Definition 11 (Non-Degenerate Path and Graph). Let \( S_g \) (resp. \( G \)) be a path (resp. a graph) with nodes given by a set of \( n \geq 3 \) points in \( \mathbb{R}^2 \). The path \( S_g \) is a non-degenerate path if and only if any three consecutive nodes in the sequence \( S_g \) are not collinear. The graph \( G \) is a non-degenerate graph if and only if there is a non-degenerate path \( S_g \) whose undirected version is \( G \). A path \( S_g \) (resp. a graph \( G \)) is degenerate if and only if it is not non-degenerate.

Requiring that the number of nodes is \( n = 4 \), the following Fig. 5 gives two degenerate paths. In Fig. 5a, the path \((1, 2, 2, 3, 1)\) is degenerate, because the number of nodes in the path is three less than \( n = 4 \) and there are three consecutive nodes \((1, 2, 2)\) that are collinear. The path \((1, 2, 3, 4, 1)\) in Fig. 5b is also degenerate, because there are three consecutive nodes \((2, 3, 4)\) that are collinear.

Requiring that the number of nodes is \( n = 4 \), the following Fig. 6 gives two degenerate graphs, which are the undirected version of the two paths shown in Fig. 5, respectively.

Note that non-degenerate graphs (resp. paths) and enclosing graphs (resp. paths) are two independent concepts. The following Fig. 7 gives examples to illustrate the difference between those two concepts.

Definition 12 (Isomorphism Type B). Let \( f_{\text{tag}} \) be a map from a finite sequence of symbols \( S_s \) to a path \( S_g \). The map \( f_{\text{tag}} \) is an isomorphism type B if and only if the following properties hold for \( f_{\text{tag}} \): 1) **Injective**: formally different symbols in \( S_s \) refer to different nodes in \( S_g \); 2) **Surjective**: every node in \( S_g \) is the image of a symbol in \( S_s \); 3) **Order-Preserving**: for any two symbols \( S_s[i], S_s[j] \) in \( S_s \), the symbol \( S_s[i] \) immediately precedes \( S_s[j] \) if and only if \( (f_{\text{tag}}(S_s[i]), f_{\text{tag}}(S_s[j])) \) is an edge in \( S_g \); 4) **Noncollinearity**: for any three consecutive symbols in \( S_s \), their images are not collinear in \( S_g \).

Remark 2. Observe that if there is an isomorphism type B \( f_{\text{tag}} \) from a finite sequence of symbols \( S_s \) to a path \( S_g \), and there are \( n \geq 3 \) formally different symbols in \( S_s \), then \( S_g \) and its undirected version \( G \) are both non-degenerate.

Note that Condition 4) in Definition 12 of isomorphism type B is different from Condition 4) in Definition 10 of isomorphism type A. For example, the label shown in Fig. 8a gives a map \( f_{\text{tag}} \) from the sequence of symbols \("a", "b", "c", "d", "b"\) to the path \((a, b, c, d, b)\). The map \( f_{\text{tag}} \) is an isomorphism type A but not an isomorphism type B, because there are three consecutive symbols \("a", "c", "d"\) that are collinear in the path. This violates Condition 4) in Definition 12. In contrast, the label shown in
Fig. 8b gives another map $f'_{tag}$ from the sequence of symbols (“a”, “b”, “c”, “d”) to the path $(a, b, c, d)$. The map $f'_{tag}$ is an isomorphism type B but not an isomorphism type A, since although the symbol “d” is formally different from each of the two consecutive symbols “a”, “b”, Node d is on the edge $(a, b)$ in the path. This violates Condition 4) in Definition 10.

We go back to look for the conditions that ensure $S$ in Eq. (8) is a bounded $n$ sided convex polygon with the origin in its interior.

For any two lines $l^i := a_i x + b_i y - 1 = 0$ and $l^j := a_j x + b_j y - 1 = 0$, if $D_{ij} := a_i b_j - b_i a_j \neq 0$, then there is a unique intersection point between those two lines, which is

$$V_{ij} := l^i \cap l^j = \left(-\frac{b_j - b_i}{D_{ij}}, a_i - a_j \frac{D_{ij}}{D_{ij}}\right).$$

Note that $V_{ij} = V_{ji}$. The fact $D_{ij} \neq 0$ implies that $a_i^2 + b_i^2 \neq 0 \land a_j^2 + b_j^2 \neq 0$, and also implies that $a_i \neq a_j \lor b_i \neq b_j$, namely, $V_{ij} \neq (0, 0)$. This makes sense since neither lines pass through the origin and therefore the origin is definitely not the intersection point. The sign of $D_{ij}$ depends on the order of $i$ and $j$, namely, $D_{ij} = -D_{ji}$. For two lines $l^i, l^j$ sharing a unique intersection point, if we can rotate $l^i$’s normal vector $(a_i, b_i)$ to $l^j$’s normal vector $(a_j, b_j)$ counterclockwise about the origin by an angle $\theta \in (0, \pi)$, then $D_{ij} > 0$; otherwise, $D_{ij} < 0$. When $D_{ij} < 0$, we can swap the value of $(a_i, b_i)$ with that of $(a_j, b_j)$ so that $D_{ij} > 0$. Hence, without loss of generality, we can assume that $D_{ij} > 0$.

For simplicity of notation, we define a unary operation $i^\oplus$ for $i \in \{1, \ldots, n\}$

$$i^\oplus := \begin{cases} i + 1, & i \in \{1, \ldots, (n-1)\}, \\ 1, & i = n, \end{cases}$$

where $i + 1$ is the ordinary arithmetic operation of addition. Inversely, we can define another unary operation $i^{\ominus}$ for $i \in \{1, \ldots, n\}$, that is, $j = i^{\ominus}$ if and only if $j^\oplus = i$.

Consider $n \geq 3$ planar lines not passing through the origin, denoted by $l^k$, $k \in \{1, \ldots, n\}$. We hope the boundary of the region $S$, generated by these lines in the way Eq. (8) gives, is an enclosing $n$ sided convex polygon with the origin in its interior. A necessary condition to make it is that there are at least $n$ different line intersection points serving as $n$ nodes of the polygon. Without loss of generality, we require the existence of such $n$ formally different intersection points $V_{ij}$, $i \in \{1, \ldots, n\}, j = i^{\oplus}$ (which also implicitly requires $l^i \neq l^j, i \in \{1, \ldots, n\}, j = i^\ominus$). To enforce this, we formulate the following constraints

$$D_{ij} := a_i b_j - b_i a_j > 0, \quad i \in \{1, \ldots, n\}, \quad j = i^{\oplus}. \quad (9)$$

Nevertheless, satisfying Eq. (9) can not ensure that there are indeed $n$ different intersection (9) can not ensure that there are indeed $n$ different intersection points (resp. lines), because two formally different representations of the intersection points (resp. lines) may refer to an identical intersection point (resp. line), as illustrated in Fig. 9.
As discussed above, the constraints Eq. (9) are not sufficient to guarantee that the graph \( G \) formed by \( S_g \) is an enclosing \( n \) sided convex polygon with the origin in its interior. To this end, we provide additional constraints in Theorem 1.

Before proceeding to Theorem 1, we first introduce Lemma 1. This lemma is useful in the sense that once constraints Eq. (9) and Eq. (10) are satisfied, we no longer need to explicitly enforce that all \( n \) intersection points (resp. lines) are indeed different from each other.

**Lemma 1.** Assume that there are \( n \geq 3 \) planar lines not passing through the origin \((0,0)\), denoted by \( l^k := a_k x + b_k y - 1 = 0, k \in \{1, \ldots, n\} \). If there are \( n \) intersection points \( V_{ij} := l^i \cap l^j, i \in \{1, \ldots, n\}, j = i^\ominus \) and the coefficients of these lines satisfy the following inequalities

\[
- a_k (b_i - b_j) + b_k (a_i - a_j) - (a_i b_j - b_i a_j) < 0, \quad i \in \{1, \ldots, n\}, j = i^\ominus, k \in \{1, \ldots, n\} - \{i, j\},
\]

then, any two formally different representations of intersection points (resp. lines) refer to different intersection points (resp. lines), and therefore all \( n \) intersection points (resp. lines) are different from each other.

**Proof.** See Section VIII-A.

Next, the following Theorem 1 gives the conditions which suffice to ensure the boundary of the region \( S \) in Eq. (8) is an enclosing \( n \) sided convex polygon with the origin in its interior. Intuitively, constraints Eq. (10) check each intersection point \( V_{ij}, i \in \{1, \ldots, n\}, j = i^\ominus \) against all its opposite \( (n-2) \) lines \( l^k, k \in \{1, \ldots, n\} - \{i, j\} \).

**Theorem 1.** Assume that there are \( n \geq 3 \) planar lines not passing through the origin \((0,0)\), denoted by \( l^k := a_k x + b_k y - 1 = 0, k \in \{1, \ldots, n\} \). If there are \( n \) intersection points \( V_{ij} := l^i \cap l^j, i \in \{1, \ldots, n\}, j = i^\ominus \), and the coefficients of these lines satisfy Eq. (10), then the boundary of the region

\[
S := \{(x,y) : \bigwedge_{k=1}^n a_k x + b_k y - 1 \leq 0\}
\]
generated by these lines is the graph \( G \) formed by the sequence of intersection points \( S_g := (V_{12}, V_{23}, \ldots, V_{(n-1)n}, V_{n1}, V_{12}) \), which defines an enclosing \( n \) sided convex polygon with the origin in its interior.

**Proof.** See Section VIII-B.

So far, we have just derived constraints Eq. (9) and Eq. (10) to ensure that the polygon of interest contains the origin \((0,0)\). The following result extends the applicability of these conditions to contain an arbitrary point.

**Corollary 1.** Assume that there are \( n \geq 3 \) planar lines not passing through the point \((\bar{x}, \bar{y})\), denoted by \( l^k := a_k (x - \bar{x}) + b_k (y - \bar{y}) - 1 = 0 \). If constraints Eq. (9) and Eq. (10) hold, then the boundary of the region \( S := \{(x,y) : \bigwedge_{k=1}^n a_k (x - \bar{x}) + b_k (y - \bar{y}) - 1 \leq 0\} \) generated by these lines is the graph \( G \) formed by the sequence of intersection points \( S_g := (V_{12}, V_{23}, \ldots, V_{(n-1)n}, V_{n1}, V_{12}) \), which defines an enclosing \( n \) sided convex polygon with the point \((\bar{x}, \bar{y})\) inside.

**Proof.** See Section VIII-C.

We finally conclude that, according to Corollary 1, Eq. (9) and Eq. (10) are the constraints that we formulate for \( 2n \) continuous variables \( a_k, b_k \in \mathbb{R}, k \in \{1, \ldots, n\} \). For any \( n \geq 3 \) planar lines not passing through an arbitrary point, if satisfying Eq. (9) and Eq. (10), we can guarantee that the boundary of the region

\[
S := \{(x,y) : \bigwedge_{k=1}^n a_k (x - \bar{x}) + b_k (y - \bar{y}) - 1 \leq 0\}
\]
generated by these lines is an enclosing \( n \) sided convex polygon with that point in its interior.

**C. Constraints Relating Continuous and Discrete Variables**

Recall that, a mesh of grid points is a map \( g \) given by \((x_i,y_j) = g_{ij}, i,j \in \{1,\ldots,N\}\) as defined in Eq. (3), and we have obtained a normalized weight matrix \( w := [w_{ij}] \) such that every grid point \( g_{ij}, i,j \in \{1,\ldots,N\} \) has a normalized weight \( w_{ij} \), respectively, in Eq. (6). The index of the grid point whose normalized weight is the greatest is denoted by \((\hat{i},\hat{j}) := \arg \max (w)\), and its coordinate is \((x_{\hat{i}}, y_{\hat{j}}) := g_{\hat{i}\hat{j}}, \) which plays the role of \((\bar{x},\bar{y})\) in Section IV-B.

The following binary variables \( z_{ij} \in \{0,1\} \) are defined so that \( z_{ij} = 1 \) if and only if the grid point \( g_{ij} \) lies inside the polygon obtained in Section IV-B. In other words,

\[
z_{ij} = 1 \implies \bigwedge_{k=1}^n a_k (x_i - x_{\hat{i}}) + b_k (y_j - y_{\hat{j}}) - 1 \leq 0, \quad i,j \in \{1,\ldots,N\},
\]

\[
z_{ij} = 0 \implies \bigvee_{k=1}^n a_k (x_i - x_{\hat{i}}) + b_k (y_j - y_{\hat{j}}) - 1 > 0, \quad i,j \in \{1,\ldots,N\}.
\]

To convert the logical constraints above into equivalent algebraic constraints, we introduce new binary variables...
where \( l_{ij}^k \in \{0, 1\} \) such that \( l_{ij}^k = 1 \) if and only if the grid point \( g_{ij} \) lies on the inner side of the line \( l^k \). That is,
\[
l_{ij}^k = 1 \implies a_k(x_i - x_j) + b_k(y_j - y_j) - 1 \leq 0, \quad i, j \in \{1, \ldots, N\}, \quad k \in \{1, \ldots, n\},
\]
\[
l_{ij}^k = 0 \implies a_k(x_i - x_j) + b_k(y_j - y_j) - 1 > 0, \quad i, j \in \{1, \ldots, N\}, \quad k \in \{1, \ldots, n\},
\]
where \( n \in \{x \in \mathbb{N} : x \geq 3\} \). These can be formulated as algebraic constraints via big-M representation \([36]\) as
\[
a_k(x_i - x_j) + b_k(y_j - y_j) - 1 \leq M_{ijk}^1 (1 - l_{ij}^k) \quad i, j \in \{1, \ldots, N\}, \quad k \in \{1, \ldots, n\},
\]
\[
- a_k(x_i - x_j) - b_k(y_j - y_j) + 1 \leq M_{ijk}^2 l_{ij}^k \quad i, j \in \{1, \ldots, N\}, \quad k \in \{1, \ldots, n\},
\]
where the parameters \( M_{ijk}^1, M_{ijk}^2, i, j \in \{1, \ldots, N\}, \quad k \in \{1, \ldots, n\} \) are sufficiently large constants. They are often set specifically across different constraints, but in this paper, we just need to choose a common constant for them.

The logical relationship between \( z_{ij} \) and \( l_{ij}^k \) is
\[
\begin{align*}
z_{ij} = 1 & \implies \sum_{k=1}^n l_{ij}^k = n, \quad i, j \in \{1, \ldots, N\}, \\
z_{ij} = 0 & \implies \sum_{k=1}^n l_{ij}^k \leq (n - 1), \quad i, j \in \{1, \ldots, N\},
\end{align*}
\]
and these can be formulated as algebraic constraints
\[
\begin{align*}
\sum_{k=1}^n l_{ij}^k & \geq nz_{ij} \quad i, j \in \{1, \ldots, N\}, \\
\sum_{k=1}^n l_{ij}^k & \leq (n - 1) + z_{ij} \quad i, j \in \{1, \ldots, N\},
\end{align*}
\]
where \( n \in \{x \in \mathbb{N} : x \geq 3\} \).

In this way, we have converted the original logical constraints Eq. (11) and Eq. (12), into equivalent algebraic inequalities Eq. (13), Eq. (14), Eq. (15) and Eq. (16). These constraints give the relation between \( 2n \) continuous variables \( a_k, b_k \in \mathbb{R}, \quad k \in \{1, \ldots, n\} \) and \( N^2 \) binary variables \( l_{ij}^k \in \{0, 1\}, \quad i, j \in \{1, \ldots, N\}, \quad k \in \{1, \ldots, n\} \), \( N^2 \) binary variables \( z_{ij} \in \{0, 1\}, \quad i, j \in \{1, \ldots, N\} \).

D. Constraints on Discrete Variables

We would like to enforce that the grid point \( g_{ij} := (x_i, y_j) \) lies inside the polygon obtained in Section IV-B. Recall that, the binary variable \( z_{ij} = 1 \) if and only if the grid point \( g_{ij} \) lies inside the polygon. Therefore, we impose that
\[
z_{ij} = 1. \quad (17)
\]
In fact, Eq. (17) is unnecessary as \( (x_i, y_j) \in S := \{(x, y) : \bigwedge_{k=1}^n a_k(x - x_j) + b_k(y_j - y_j) - 1 \leq 0\} \) given by Corollary 1 in Section IV-B, which means this grid point must lie inside the polygon. However, we explicitly list it here for better clarity.

The convex polygon is intended to approximate the PRS. To guarantee a level of confidence of the PRS, we impose that the sum of the normalized weights \( w_{ij} \) of the grid points which lie inside the polygon should be greater than this confidence level \( \alpha \). That is,
\[
\sum_{i=1}^N \sum_{j=1}^N w_{ij}z_{ij} \geq \alpha. \quad (18)
\]

Eq. (17) and Eq. (18) lead to two constraints that we formulate for \( N^2 \) binary variables \( z_{ij} \in \{0, 1\}, \quad i, j \in \{1, \ldots, N\} \).

E. Formulation of MINLP Optimization Framework

To solve the problem described in Section IV-A, we have chosen the objective function \( \min \sum_{i=1}^N \sum_{j=1}^N z_{ij} \) in Eq. (7), and introduced the following three types of decision variables with \( n \in \{x \in \mathbb{N} : x \geq 3\} \) in Section IV-A:

- \( 2n \) continuous variables \( a_k, b_k \in \mathbb{R}, \quad k \in \{1, \ldots, n\} \);
- \( N^2 \) binary variables \( l_{ij}^k \in \{0, 1\}, \quad i, j \in \{1, \ldots, N\}, \quad k \in \{1, \ldots, n\} \);
- \( N^2 \) binary variables \( z_{ij} \in \{0, 1\}, \quad i, j \in \{1, \ldots, N\} \).

Collect the previous constraints Eq. (9), Eq. (10) from Section IV-B; Eq. (13), Eq. (14), Eq. (15), Eq. (16) from Section IV-C; Eq. (17), Eq. (18) from Section IV-D.

We formally formulate the problem stated in Section II as an MINLP optimization framework below. All coordinates are expressed in a global coordinate system. If not specified, \( \forall i, j \in \{1, \ldots, N\}, \quad \forall k \in \{1, \ldots, n\} \).
variables. Gurobi is well-suited for addressing such constraints.

A benchmark that provides the optimal solution to Eq. (19) while ensuring accuracy. Algorithm 2 just serves as this issue, we develop Algorithm 3 to address scales poorly with a large number of grid points. To address this issue, we develop Algorithm 3 that can efficiently solve Eq. (19) while ensuring accuracy. Algorithm 2 just serves as a benchmark that provides the optimal solution to Eq. (19) which is used only for comparison purposes. As explained in Fig. 11, this algorithm performs weighted sampling based on the KDE values to select representative grid points \( g' \) from all grid points \( g \). Unlike the original MINLP problem, which uses all grid points \( g \), a new MINLP problem can be formulated using representative grid points \( g' \). Then, the cutting planes and branching algorithm built in Gurobi can be applied to solve this newly formulated MINLP problem. As illustrated in Fig. 12, the optimal solution to the new MINLP problem is an approximation to the solution of the original MINLP problem. By doing so, efficiency comes at the cost of optimality. Since the size of \( g' \) can be far less than \( g \), the new MINLP problem reduces the number of decision variables and constraints, greatly contributing to reducing computational time.

V. Solution Method

In this section, we develop a heuristic algorithm to efficiently solve the previous MINLP optimization.

A. MINLP Optimal Algorithm

Although the optimization framework Eq. (19) is an MINLP problem with nonlinear constraints, all constraints except for Eq. (9) and Eq. (10) are in fact linear ones. Fortunately, even the quadratic constraints Eq. (9) and Eq. (10) are bilinear constraints that only involve the product of disjoint pairs of variables. Gurobi is well-suited for addressing such constraints. It employs cutting planes and branching algorithm; see [37] for more information. Leveraging this solver, Algorithm 2 finds an optimal solution to Eq. (19), which results in a convex approximation of the PRS computed by Algorithm 1.

B. MINLP Heuristic Algorithm

However, Algorithm 2 is computationally expensive and scales poorly with a large number of grid points. To address this issue, we develop Algorithm 3 that can efficiently solve Eq. (19) while ensuring accuracy. Algorithm 2 just serves as a benchmark that provides the optimal solution to Eq. (19)

In the pseudo-code of Algorithm 3, we use \( N' \) to represent the number of representative grid points \( g' \). From Lines 1 through 6, we use the AES algorithm proposed by [38] to randomly select \( N' \) representative grid points \( g' \) from all \( N^2 \) grid points \( g \) without replacement, according to their normalized weights \( w \). Recall that \( g \) is the output of Algorithm 1 and \( w := [w_{ij}] \) is obtained through Eq. (6). From Line 7 through 9, the original weights \( w' \) of representative grid points \( g' \) are re-normalized to \( \tilde{w'} \), so that their sum is equal to one. The process of re-normalization is crucial to guarantee the accuracy of the solution obtained from Algorithm 3, when

Algorithm 2 MINLP Optimal Algorithm

1: function MINLPSOLVER(\( \alpha, g, w \))
2: \( \hat{i}, \hat{j} = \operatorname{arg\ max}(w) \)
3: Determine \( x_{\hat{i}}, y_{\hat{j}} \) according to \( \hat{i}, \hat{j} \) and \( g \)
4: Formulate Eq. (19) for grid points \( g \) with weights \( w \) given confidence level \( \alpha \)
5: Implement cutting planes and branching algorithm to solve Eq. (19) for \( a_k, b_k \)
6: return \( a_k, b_k, x_{\hat{i}}, y_{\hat{j}} \)

Algorithm 3 MINLP Heuristic Algorithm

1: function FINDPOLYGON(\( g, w, \alpha \))
2: \( a_k, b_k, x_{\hat{i}}, y_{\hat{j}} = \text{MINLPSOLVER}(\alpha, g, w) \)
3: return \( a_k, b_k, x_{\hat{i}}, y_{\hat{j}} \)
Algorithm 3 MINLP Heuristic Algorithm

1: function WGT Samp\(g, w, N_s\)
2: \(u_{ij} = \text{random}(0, 1)\)
3: \(k_{ij} = u_{ij}^{1/w_{ij}}\)
4: Take first \(N_s\) grids from \(g\) in descending order of \(k_{ij}\)
5: Take the weights of those \(N_s\) grids from \(w\)
6: return \(g', w'\)

7: function NORM WGT\(w', N_s\)
8: \(\dot{w}' = w' + \text{ones}(N_s) \cdot (1 - \text{sum}(w'))/N_s\)
9: return \(\dot{w}'\)

10: function MINLPSolver\(g', \dot{w}', \alpha, g\)
11: \(i, j = \arg \max(\dot{w}')\)
12: Determine \(x_i, y_j\) according to \(i, j\) and \(g\)
13: Formulate Eq. (19) for grid points \(g'\) with weights \(\dot{w}'\)
14: Implement cutting planes and branching algorithm to solve Eq. (19) for \(a_k, b_k\)
15: return \(a_k, b_k, x_i, y_j\)

16: function FINDPOLYGON\(g, w, N_s, \alpha\)
17: \(g', w' = \text{WGT Samp}(g, w, N_s)\)
18: \(\dot{w}' = \text{NORM WGT}(w', N_s)\)
19: \(a_k, b_k, x_i, y_j = \text{MINLPSolver}(g', \dot{w}', \alpha, g)\)
20: Determine the polygon \(S\) according to \(a_k, b_k, x_i, y_j\)
21: return \(S\)
22: FINDPOLYGON\(g, w, N_s, \alpha\)

VI. MAIN RESULTS

In this section, we conduct comprehensive case studies to compare the performance of the MINLP Optimal algorithm (Algorithm 2), and the MINLP Heuristic algorithm (Algorithm 3), with the Bounding Box algorithm introduced in [28], as well as the impact of different parameters on the performance of the algorithms. We also discuss the robustness of the MINLP Heuristic algorithm. The tests were implemented in Python.
$M_1^{i,j,k}, M_2^{i,j,k}$ in Eq. (13) and Eq. (14), and the number of data samples is $M = 1000$. The setting below is chosen as a baseline: The number of sides is $n = 4$, the confidence level is $\alpha = 90\%$, the number of grid points used by MINLP Optimal is $N^2 = 20^2$, and the number of grid points used by MINLP Heuristic is $N^i = 70$ for Case I and $N^i = 60$ for Case II. In the following figures and tables, the baseline is marked with the symbol “*”.

**B. Different Numbers of Sides**

In this part, we compare the performance of different algorithms for different numbers of polygon sides: $n = 3, 4, 5$. The other parameters are the same as in the baseline.

The results are shown in Fig. 15 and Table I. Note that changing the number of sides $n$ is only applicable for MINLP Optimal and MINLP Heuristic. For MINLP Optimal or MINLP Heuristic, as $n$ increases from 3 to 5, the area of the polygon becomes smaller, while the ratio is just slightly different and always closer to the confidence level $\alpha$. This suggests that the results of both algorithms become less conservative while ensuring accuracy. However, the associated computational time increases as the number of decision variables and constraints in Eq. (19) increases with an increase on $n$. To achieve a balance between optimality and computational efficiency, we set $n = 4$.

When $n = 4$, the area and ratio of either MINLP Optimal or MINLP Heuristic are much smaller than those of the Bounding Box algorithm, which suggests that the polygons obtained by both MINLP Optimal and MINLP Heuristic algorithms are more tight and accurate than the Bounding Box. This clearly benefits real motion planning. When considering collision avoidance between the vehicle and an obstacle, a less conservative convex approximation of the PRS allows a larger feasible search area which may provide a better-planned trajectory. Plus, the computational time of MINLP Heuristic is far less than MINLP Optimal ($0.801s < 70.5s$ for Case I and $0.361s < 18.6s$ for Case II), while their results of area and ratio are still similar. This demonstrates the computational efficiency of MINLP Heuristic while ensuring accuracy, making online implementation feasible.

As stated in Section V, the polygon obtained by MINLP Optimal is optimal in the sense that it has a minimum area without violating constraints. However, for some cases in Table I, the area of MINLP Heuristic (e.g. 1807.9 m$^2$) may be slightly less than MINLP Optimal (e.g. 1827.0 m$^2$). The approximation error comes from the re-normalization with the newly selected $N^i$ grid points $g'$. The polygon obtained by MINLP Heuristic is an optimal solution to a new MINLP problem based on those selected $N^i$ grid points, which is also an approximation to the original MINLP problem involving $N^2$ grid points. Since the original and new MINLP problems have different grid settings, there is a chance that the polygon area of MINLP Heuristic is less than MINLP Optimal. However, those errors are relatively small according to our experiments.

**C. Different Numbers of Grid Points**

Here, we compare the performance of the algorithms when considering different numbers of grid points: $N^2 = 10^2, 20^2, 30^2, 40^2$. For MINLP Heuristic, $N^i = 20, 70, 160, 280$ grid points are randomly selected for Case I and $N^i = 15, 60, 135, 240$ grid points for Case II, respectively. The other parameters are the same as in the baseline.

From Fig. 16 and Table II, we can draw some conclusions as follows. The results obtained by the implementation of all three algorithms given in Fig. 16a and Fig. 16b are visually very different from the other subfigures in Fig. 16. This is because when the number of grid points is too small ($N^2 = 10^2$), KDE can not approximate PDF well. As indicated in Table II, for every algorithm, as $N^2$ ranges from $20^2$ to $40^2$, the performance in ratio and area slightly improves. However, as $N^2$ increases, the increase in the number of decision variables and constraints leads to a significant increase in computational time. As shown in Table II, the computational time of MINLP Heuristic goes from 0.801 s to 125.5 s for Case I and from 0.361 s to 23.8 s for Case II. Thus, there is a slight improvement in accuracy and optimality, but at huge cost of efficiency. To reach a compromise, the number of grid points can be chosen as $N^2 = 20^2$.

When $N^2$ is fixed to $20^2$, the area and ratio of MINLP Optimal and MINLP Heuristic are much smaller than those of the Bounding Box algorithm, indicating MINLP Optimal and MINLP Heuristic significantly outperform Bounding Box in terms of optimality and accuracy. Further, for MINLP Optimal and MINLP Heuristic, their results of ratio and area are quite similar. However, the computational time of MINLP Heuristic greatly outperforms MINLP Optimal, i.e., $0.801s < 70.5s$ for Case I and $0.361s < 18.6s$ for Case II, demonstrating the efficiency of MINLP Heuristic. Therefore, MINLP Heuristic can guarantee near-optimality, accuracy, and efficiency simultaneously, which makes it very well suited for real-time implementation.

**D. Different Confidence Levels**

In this part, we compare the performance of the algorithms relative to different confidence levels: $\alpha = 90\%, 95\%$, and $99\%$. The other parameters are the same as in the baseline.

In Fig. 17 and Table III, for both Case I and II, as $\alpha$ increases, the area of the polygon obtained by MINLP Optimal, MINLP Heuristic or Bounding Box respectively increases while the computational time of each algorithm slightly fluctuates.
Fig. 15. Comparisons of different numbers of sides $n$ (For Bounding Box $n \equiv 4$)

### TABLE I

<table>
<thead>
<tr>
<th># Sides $n$</th>
<th>Algorithm</th>
<th>Case I</th>
<th>Case II</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Ratio</td>
<td>Area (m$^2$)</td>
</tr>
<tr>
<td>3 (Triangle)</td>
<td>MINLP Optimal</td>
<td>90.1%</td>
<td>1835.9</td>
</tr>
<tr>
<td></td>
<td>MINLP Heuristic</td>
<td>90.2%</td>
<td>1911.9</td>
</tr>
<tr>
<td></td>
<td>Bounding Box</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>4 (Quadrilateral) *</td>
<td>MINLP Optimal</td>
<td>91.5%</td>
<td>1827.0</td>
</tr>
<tr>
<td></td>
<td>MINLP Heuristic</td>
<td>90.7%</td>
<td>1807.9</td>
</tr>
<tr>
<td></td>
<td>Bounding Box</td>
<td>97.9%</td>
<td>3235.1</td>
</tr>
<tr>
<td>5 (Pentagon)</td>
<td>MINLP Optimal</td>
<td>89.7%</td>
<td>1667.9</td>
</tr>
<tr>
<td></td>
<td>MINLP Heuristic</td>
<td>90.2%</td>
<td>1721.9</td>
</tr>
<tr>
<td></td>
<td>Bounding Box</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Fig. 16. Comparisons of different numbers of grid points $N^2$
TABLE II
COMPARISONS OF DIFFERENT NUMBERS OF GRID POINTS

<table>
<thead>
<tr>
<th># Grid Points $N^2$</th>
<th>Algorithm</th>
<th>Ratio</th>
<th>Area (m$^2$)</th>
<th>Time (s)</th>
<th>Ratio</th>
<th>Area (m$^2$)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Case I</td>
<td></td>
<td></td>
<td></td>
<td>Case II</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MINLP Optimal</td>
<td>87.2%</td>
<td>1703.6</td>
<td>0.321</td>
<td>82.2%</td>
<td>3784.7</td>
<td>0.082</td>
</tr>
<tr>
<td></td>
<td>MINLP Heuristic</td>
<td>90.2%</td>
<td>1908.7</td>
<td>0.064</td>
<td>92.6%</td>
<td>4110.5</td>
<td>0.015</td>
</tr>
<tr>
<td></td>
<td>Bounding Box</td>
<td>97.9%</td>
<td>3257.2</td>
<td>0.209</td>
<td>99.6%</td>
<td>6318.8</td>
<td>0.184</td>
</tr>
<tr>
<td>20$^2$ *</td>
<td>MINLP Optimal</td>
<td>91.5%</td>
<td>1872.0</td>
<td>70.5</td>
<td>91.4%</td>
<td>3570.5</td>
<td>18.6</td>
</tr>
<tr>
<td></td>
<td>MINLP Heuristic</td>
<td>90.7%</td>
<td>1807.9</td>
<td>0.801</td>
<td>90.6%</td>
<td>3457.1</td>
<td>0.361</td>
</tr>
<tr>
<td></td>
<td>Bounding Box</td>
<td>97.9%</td>
<td>3235.1</td>
<td>0.216</td>
<td>97.7%</td>
<td>4987.2</td>
<td>0.189</td>
</tr>
<tr>
<td>30$^2$</td>
<td>MINLP Optimal</td>
<td>91.4%</td>
<td>1829.1</td>
<td>1150.2</td>
<td>92.0%</td>
<td>3578.3</td>
<td>362.9</td>
</tr>
<tr>
<td></td>
<td>MINLP Heuristic</td>
<td>90.4%</td>
<td>1755.9</td>
<td>8.9</td>
<td>91.8%</td>
<td>3443.1</td>
<td>6.9</td>
</tr>
<tr>
<td></td>
<td>Bounding Box</td>
<td>97.7%</td>
<td>3157.7</td>
<td>0.225</td>
<td>97.6%</td>
<td>4893.5</td>
<td>0.193</td>
</tr>
<tr>
<td>40$^2$</td>
<td>MINLP Optimal</td>
<td>91.3%</td>
<td>1827.0</td>
<td>2360.1</td>
<td>91.4%</td>
<td>3554.8</td>
<td>1237.4</td>
</tr>
<tr>
<td></td>
<td>MINLP Heuristic</td>
<td>90.1%</td>
<td>1721.1</td>
<td>125.5</td>
<td>90.9%</td>
<td>3425.0</td>
<td>23.8</td>
</tr>
<tr>
<td></td>
<td>Bounding Box</td>
<td>97.1%</td>
<td>3030.2</td>
<td>0.239</td>
<td>97.3%</td>
<td>4722.6</td>
<td>0.211</td>
</tr>
</tbody>
</table>

Fig. 17. Comparisons of different confidence levels $\alpha$
we run MINLP Optimal once to obtain a constant optimal
A major difference would mean that MINLP Heuristic is not
weighted sampling, the selected grid points
In addition, we evaluate the computational time of running
and the optimal polygon and obtain ten distances eventually.
the Jaccard distance between every approximate polygon
approximate polygons which may differ every time. We evaluate
polygon and run MINLP Heuristic ten times to obtain ten
constant and the same as in the respective baselines.
N
to the varying number
N
to each other.
smaller Jaccard distance means the two sets are more similar
[33], where the operation
Card
B
and
Card
A
B
means of the
Jaccard distance
robust enough for applications in real scenarios.
the computational time spent significantly differs every time.
Time: this refers to whether
Space: this refers to whether the convex polygon obtained
changes every time it is applied. This leads to the following
concerns regarding the robustness of MINLP Heuristic. 1)
Space: this refers to whether the convex polygon obtained
significantly differs every time; 2) Time: this refers to whether
the computational time spent significantly differs every time.
A major difference would mean that MINLP Heuristic is not
robust enough for applications in real scenarios.

E. Robustness of the MINLP Heuristic Algorithm
As indicated in Algorithm 3, every time we run MINLP Heuristic, we apply a weighted sampling to select representative
grid points \( g' \) out of the grid points \( g \). Due to the randomness of weighted sampling, the selected grid points \( g' \) typically differ
every time, and the polygon obtained by MINLP Heuristic changes every time it is applied. This leads to the following
concerns regarding the robustness of MINLP Heuristic. 1) Space: this refers to whether the convex polygon obtained
significantly differs every time; 2) Time: this refers to whether the computational time spent significantly differs every time.
A major difference would mean that MINLP Heuristic is not robust enough for applications in real scenarios.

The similarity between two polygons can be quantified by
means of the Jaccard distance. Formally, for any two sets \( A \) and \( B \), Jaccard distance is defined as
\[ d(A, B) = \frac{\text{Card}(A \cup B)}{\text{Card}(A \cap B)} \]
[33], where the operation Card finds the cardinality of a set. A smaller Jaccard distance means the two sets are more similar
to each other.

We analyze the robustness of MINLP Heuristic with respect to
the varying number \( N_s \) of grid points \( g' \) used by MINLP Heuristic for Cases I and II mentioned above. For either Case I or Case II, all the parameters, except for the number \( N_s \), are constant and the same as in the respective baselines.

For either Case I or Case II, given a fixed number \( N_s \), we run MINLP Optimal once to obtain a constant optimal
polygon and run MINLP Heuristic ten times to obtain ten approximate polygons which may differ every time. We evaluate
the Jaccard distance between every approximate polygon and the optimal polygon and obtain ten distances eventually.
In addition, we evaluate the computational time of running
MINLP Heuristic every time. We repeat the operation above for
\( N_s = 50, 60, 70, 80, 90 \), respectively. The results are depicted
in Fig. 18. As illustrated in Fig. 18a, for both cases, when \( N_s \) is too small, like \( N_s = 50 \), different Jaccard distances fluctuate
significantly around the average value, suggesting that MINLP Heuristic has poor robustness under this circumstance. As \( N_s \) increases from 50 to 90, the average Jaccard distance decreases
from 0.28, 0.21 to 0.12, 0.10, respectively. This means the
approximate polygon obtained by MINLP Heuristic becomes
more near-optimal with respect to an increasing \( N_s \). Moreover,
the variance of ten Jaccard distances also decreases as \( N_s \) grows. That is, MINLP Heuristic becomes more robust with
respect to the increase of \( N_s \). As suggested in Fig. 18b, as \( N_s \) increases, the average computational time increases from
0.09 s to 3.1 s for Case I, and from 0.16 s to 2.81 s for Case II. Additionally, as \( N_s \) grows, the variance of computational
time increases.

In summary, the robustness of MINLP Heuristic comes at
the cost of losing computational efficiency. For the case studies
under consideration, a balance is achieved for \( N_s = 70 \) for Case I or \( N_s = 60 \) for Case II. This makes the MINLP Heuristic

<table>
<thead>
<tr>
<th>Confidence Level ( \alpha )</th>
<th>Algorithm</th>
<th>Case I</th>
<th></th>
<th>Case II</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ratio</td>
<td>Area (m²)</td>
<td>Time (s)</td>
<td>Ratio</td>
<td>Area (m²)</td>
</tr>
<tr>
<td>90% *</td>
<td>MINLP Optimal</td>
<td>91.5%</td>
<td>1872.0</td>
<td>70.5</td>
<td>91.4%</td>
</tr>
<tr>
<td></td>
<td>MINLP Heuristic</td>
<td>90.7%</td>
<td>1807.9</td>
<td>0.801</td>
<td>90.6%</td>
</tr>
<tr>
<td></td>
<td>Bounding Box</td>
<td>97.9%</td>
<td>3235.1</td>
<td>0.216</td>
<td>97.7%</td>
</tr>
<tr>
<td>95%</td>
<td>MINLP Optimal</td>
<td>95.7%</td>
<td>2596.1</td>
<td>53.5</td>
<td>96.1%</td>
</tr>
<tr>
<td></td>
<td>MINLP Heuristic</td>
<td>96.1%</td>
<td>2424.7</td>
<td>0.417</td>
<td>96.2%</td>
</tr>
<tr>
<td></td>
<td>Bounding Box</td>
<td>99.4%</td>
<td>3709.1</td>
<td>0.243</td>
<td>99.0%</td>
</tr>
<tr>
<td>99%</td>
<td>MINLP Optimal</td>
<td>99.9%</td>
<td>4530.6</td>
<td>50.6</td>
<td>99.9%</td>
</tr>
<tr>
<td></td>
<td>MINLP Heuristic</td>
<td>99.9%</td>
<td>4443.5</td>
<td>0.377</td>
<td>99.9%</td>
</tr>
<tr>
<td></td>
<td>Bounding Box</td>
<td>100%</td>
<td>6314.8</td>
<td>0.303</td>
<td>100%</td>
</tr>
</tbody>
</table>

**Table III**
Comparisons of different confidence levels

![Fig. 18. Robustness analysis of MINLP Heuristic algorithm](image-url)
robust enough and its applications promising.

VII. CONCLUSION

This paper proposes an efficient convex approximation of the PRS of an uncertain dynamic system. For arbitrary unknown uncertainties, especially unbounded ones, the notion of the PRS is introduced as an extension of the traditional reachable set, connecting chance constraint formulation and reachability analysis. In this paper, the PRS is obtained through a data-driven approach employing a KDE method. FFT is then customized to accelerate the computation of KDE to make it computationally attractive. The irregularity or non-convexity of the PRS refrains its use in optimal design applications. To address this issue, we formulate a MINLP problem whose solution accounts for a convex approximation for the PRS. We then develop a MINLP Heuristic algorithm to solve it. Comprehensive case studies demonstrate that our proposed algorithm enjoys the benefits of accuracy, efficiency, near-optimality, as well as robustness while providing a convex approximation for the PRS.

This work is limited to a 2D system. Future work will be devoted to solving higher dimensional problems, and applying the results to the design of safety-critical real-time motion planning algorithms for uncertain robotic systems.

VIII. APPENDIX

A. Proof of Lemma 1

I) First, we prove that any two formally different representations of lines refer to different lines.

Given $n \geq 3$ planar lines not passing through the origin $l^k := a_k x + b_k y - 1 = 0, k \in \{1, \ldots, n\}$, we can construct a sequence of lines $(l^1, l^2, \ldots, l^n, l^1)$. Two formally different lines $l^i, l^j$ in the sequence are defined to be consecutive if and only if $j = i \oplus v \land j = i \ominus w$. Since there are $n$ intersection points $V_{ij} := l^i \cap l^j, i \in \{1, \ldots, n\}, j = i \oplus v, j = i \ominus w$, it follows that $l^i \neq l^j, i \in \{1, \ldots, n\}, j = i \oplus v, j = i \ominus w$. Thus, any two formally different lines consecutive in the sequence are indeed different from each other.

For $n = 3$, the sequence of lines is $(l^1, l^2, l^3, l^1)$. Since any two formally different lines in the sequence $(l^1, l^2, l^3, l^1)$ are consecutive, thus $l^1, l^2, l^3$ are indeed different from each other. For $n \geq 4$, there are at least one pair of formally different lines not consecutive in the sequence. We have already shown that any two formally different lines consecutive in the sequence are indeed different from each other, and therefore it remains to show that any two formally different lines not consecutive in the sequence are also indeed different from each other, given the conditions of this lemma. We prove this by contradiction. Suppose that $l^u, l^v, u \in \{1, \ldots, n\}, v \in \{1, \ldots, n\} - \{u^\ominus, u^\ominus, u^\ominus, u^\ominus\}$ are two formally different lines not consecutive in the sequence but are indeed an identical line. Then $V_{u \oplus v}$ and $V_{u \ominus v}$ are both on the line $l^w$, which means

\begin{align*}
-a_u (b_v - b_u) + b_u (a_v - a_u) - (a_v) b_u - b_v a_u & = 0, \\
-a_u (b_v - b_u) + b_u (a_v - a_u) - (a_v) b_u - b_v a_u & = 0.
\end{align*}

However, since $v \notin \{u^\ominus, u^\ominus, u^\ominus, u^\ominus\}$, it follows that the subscript of $V_{u \ominus v}$ is also formally different from $V_{u \ominus u}$ and $V_{u u \ominus}$. Thus, since the constraints Eq. (10) are satisfied, we obtain two instances

\begin{align*}
-a_u (b_v - b_u) + b_u (a_v - a_u) - (a_v) b_u - b_v a_u & < 0, \\
-a_u (b_v - b_u) + b_u (a_v - a_u) - (a_v) b_u - b_v a_u & < 0,
\end{align*}

which contradicts Eq. (20).

Thus, if constraints Eq. (9) and Eq. (10) are satisfied, any two formally different lines are indeed different from each other.

II) Next, we show that if there are $n$ intersection points $V_{ij} := l^i \cap l^j, i \in \{1, \ldots, n\}, j = i \ominus v$ (namely, $D_{ij} := a_i b_j - b_i a_j > 0, i \in \{1, \ldots, n\}, j = i \ominus v$ according to Eq. (9)), and the constraints Eq. (10) are satisfied, then any two formally different intersection points are indeed different from each other.

We can prove this by contradiction. Assume that there are two formally different intersection points $V_{ij}, V_{uv}, i \in \{1, \ldots, n\}, j = i \ominus v, u \in \{1, \ldots, n\} - \{i\}, v = u \ominus w$ which are indeed an identical intersection point. Since $n \geq 3$, it follows that $(u \neq i \land u \neq j) \lor (v \neq i \land v \neq j)$, and without loss of generality, we assume $v \neq i \land v \neq j$. Thus, the constraints Eq. (10) apply to $V_{ij}$ and $l^v$, which yields an instance

\begin{align}
-a_v (b_i - b_j) + b_v (a_i - a_j) - (a_i b_i - b_i a_j) & < 0. \tag{21}
\end{align}

However, since $V_{ij}, V_{uv}$ are indeed an identical intersection point, $V_{ij}$ is on the line $l^v$, meaning that

\begin{align*}
-a_v (b_i - b_j) + b_v (a_i - a_j) - (a_i b_i - b_i a_j) & = 0,
\end{align*}

which contradicts Eq. (21).

Combining I) and II), we conclude that if constraints Eq. (9) and Eq. (10) are satisfied, any two formally different representations of intersection points (resp. lines) refer to different intersection points (resp. lines), and therefore all $n$ intersection points (resp. lines) are indeed different from each other.

B. Proof of Theorem 1

I) Since there are $n \geq 3$ intersection points $V_{ij} := l^i \cap l^j, i \in \{1, \ldots, n\}, j = i \ominus v$, we can enumerate them in a sequence of intersection points $S_g := (V_{12}, V_{23}, \ldots, V_{(n-1)n}, V_{n1}, V_{12})$. According to Lemma 1, since the coefficients of these lines satisfy constraints Eq. (9) and Eq. (10), then any two formally different intersection points (resp. lines) are indeed different from each other.

II) We can also construct a finite sequence of symbols $S_s := (V_{12}^{*}, V_{23}^{*}, V_{(n-1)n}^{*}, V_{n1}^{*}, V_{12}^{*})$. In $S_s$, I) since $n \geq 3$, there are at least three formally different symbols $V_{12}^{*}$, $V_{23}^{*}$, and $V_{31}^{*}$; 2) No symbols appear more than once except for the case in which only the first symbol “V_{12}” and last symbol “V_{12}’” are formally identical. Therefore, Definition 9 applies and $S_s$ is formally enclosing.

Consider a map $f_{tag}$ from $S_s$ to $S_g$ given by the operation of labelling the intersection points, i.e., $V_{ij} := l^i \cap l^j, i \in \{1, \ldots, n\}, j = i \ominus v$. It can be verified that $f_{tag}$ is injective, surjective and order-preserving. The map $f_{tag}$ also satisfies Condition 4) in Definition 10. This is because in $S_s$, for any
two consecutive symbols $V_{i\oplus j}$ and $V_{i\ominus j}$, and any third symbol $V_{uv}$ which is formally different from each of $V_{i\oplus j}$ and $V_{i\ominus j}$, according to Eq. (10) we obtain

$$-a_i(b_u-b_v)+b_i(a_u-a_v)-(a_u b_v-b_u a_v) \neq 0,$$

$$u \notin \{i\oplus j\}, v = u\ominus,$$

which means that the intersection point $V_{uv}$ is not on the line $l^i$. According to I), since formally different symbols $V_{i\oplus j}$ and $V_{i\ominus j}$ refer to different intersection points $V_{i\oplus j}$ and $V_{i\ominus j}$, thus these two intersection points determine a non-degenerate line segment. Further, the line segment determined by these two points is a segment of the line $l^i$. Hence, the intersection point $V_{uv}$ that the third symbol $V_{uv}$ refers to is not on the edge $(V_{i\ominus j}, V_{i\oplus j})$. In summary, the map $f_{\text{tag}}$ satisfies all the conditions in Definition 12 and therefore is an isomorphism type A.

According to Remark 1, since $S_a$ is formally enclosing, and $f_{\text{tag}}$ is an isomorphism type A, it follows that the graph $G$ formed by $S_g$ is enclosing.

III) Since $n \geq 3$, the number of formally different symbols in $S_a$ is $n \geq 3$.

As concluded in II), the map $f_{\text{tag}}$ is injective, surjective, and order-preserving. The map $f_{\text{tag}}$ also satisfies Condition 4 in Definition 12. This is because, for any three consecutive symbols $V_{i\ominus j}$, $V_{i\oplus j}$, $V_{i\ominus k}$ in $S_a$, applying Eq. (10) yields

$$-a_i(b_{i\oplus j}-b_{i\ominus j})+b_i(a_{i\ominus j}-a_{i\oplus j})-(a_{i\ominus j} b_{i\oplus j}-b_{i\ominus j} a_{i\oplus j}) \neq 0,$$

which means that the intersection point $V_{i\ominus j}$ is not on the line $l^i$. From I), since formally different symbols $V_{i\oplus j}$ and $V_{i\ominus j}$ refer to different intersection points $V_{i\oplus j}$ and $V_{i\ominus j}$, then $V_{i\ominus j}$ and $V_{i\oplus j}$ are different from each other. Further, the intersection point $V_{i\ominus j}$ and $V_{i\oplus j}$ are on the line $l^i$. Hence, the three intersection points $V_{i\ominus j}$, $V_{i\oplus j}$, $V_{i\ominus k}$ are not collinear. In summary, the map $f_{\text{tag}}$ satisfies all the conditions in Definition 12 and therefore is an isomorphism type B.

According to Remark 2, since $f_{\text{tag}}$ is an isomorphism type B, and there are $n \geq 3$ formally different symbols in $S_a$, it follows that the graph formed by $S_g$ is non-degenerate.

IV) We have shown that the graph $G$ formed by $S_g$ is an enclosing n sided polygon. For a line $l^k$, $k \in \{1, \ldots, n\}$, once the constraints Eq. (10) are satisfied, it follows that

$$-a_k (b_i - b_j) + b_k (a_i - a_j) - (a_i b_j - b_i a_j) < 0,$$

$$i \in \{1, \ldots, n\} - \{k\ominus, k\}, j = i\ominus,$$

$$-a_k (b_i - b_j) + b_k (a_i - a_j) - (a_i b_j - b_i a_j) = 0,$$

$$i \in \{k\ominus, k\}, j = i\ominus.$$  \hspace{5cm} (22)

Since there are $n$ intersection points $V_{ij}$, $i \in \{1, \ldots, n\}, j = i\ominus$, according to Eq. (9), we have $D_{ij} := a_i b_j - b_i a_j > 0$, $i \in \{1, \ldots, n\}$. Thus, the inequalities Eq. (22) can be transformed to

$$a_k \frac{(b_i - b_j)}{D_{ij}} + b_k \frac{(a_i - a_j)}{D_{ij}} - 1 < 0,$$

$$i \in \{1, \ldots, n\} - \{k\ominus, k\}, j = i\ominus,$$

$$a_k \frac{(b_i - b_j)}{D_{ij}} + b_k \frac{(a_i - a_j)}{D_{ij}} = 1,$$

$$i \in \{k\ominus, k\}, j = i\ominus,$$  \hspace{5cm} (23)

which means that all $n$ intersection points $V_{ij}$, $i \in \{1, \ldots, n\}, j = i\ominus$ are on the same side of the line $l^k$. Therefore, in $S_g$, for any two consecutive intersection points $V_{ij}, V_{uv}$ where $i \in \{1, \ldots, n\} - \{k\ominus, k\}, j = i\ominus$, $u = i\ominus$, $v = i\ominus$, we can define a set constructed by the convex combination of $V_{ij}$ and $V_{uv}$ excluding these two boundary points, which is

$$\left\{t \frac{(b_i - b_j)}{D_{ij}} + (1-t) \frac{(b_u - b_v)}{D_{uv}} \mid t \in \mathbb{R} \land 0 < t < 1\right\}.$$ 

It is a subset of the half plane $\{(x, y) : a_k x + b_k y - 1 < 0\}$. In other words, the edge $(V_{i\ominus j}, V_{k\ominus ij})$, $i \in \{1, \ldots, n\} - \{k\ominus\}$, is on the strict inner side of the line $l^k$. This is because

$$a_k \left(1 - \frac{(b_i - b_j)}{D_{ij}} \right) + \frac{(b_u - b_v)}{D_{uv}} +$$

$$b_k \left(1 - \frac{(a_i - a_j)}{D_{ij}} \right) - \frac{(a_u - a_v)}{D_{uv}} -$$

$$1 < 0,$$  \hspace{5cm} (24)

according to Eq. (23). For the polygon $G$ formed by $S_g$, according to Eq. (24), we have shown that every edge $(V_{i\ominus j}, V_{i\oplus j})$, $i \in \{1, \ldots, n\} - \{k\ominus\}$ of the polygon is on the same side of the line $l^k$, $k \in \{1, \ldots, n\}$ that the edge $(V_{k\ominus j}, V_{k\ominus ij})$, $k \in \{1, \ldots, n\}$ defines. Recall the definition of a convex polygon: For each edge of the polygon, all the other edges are on the same side of the line that the edge defines. Therefore, the polygon $G$ formed by $S_g$ is a convex polygon.

V) Recall that for an arbitrary point $(u, v)$ and an enclosing $n$ sided convex polygon determined by $n$ lines $l^k := a_k x + b_k y - 1 = 0, k \in \{1, \ldots, n\}$, if $\bigwedge_{k=1}^n a_k u + b_k v - 1 < 0$ holds, then the point $(u, v)$ lies inside the polygon. We have shown above that if constraints Eq. (9) and Eq. (10) are satisfied, then the graph $G$ formed by $S_g$ is an enclosing $n$ sided convex polygon. Since $a_k \cdot 0 + b_k \cdot 0 - 1 \equiv -1 < 0, k \in \{1, \ldots, n\}$, the origin $(0, 0)$ lies inside the polygon $G$ formed by $S_g$.

From the above, we conclude that if constraints Eq. (9) and Eq. (10) are satisfied, then the graph $G$ formed by $S_g$ is an enclosing $n$ sided convex polygon with the origin $(0, 0)$ inside. For such $n \geq 3$ planar lines $l^k := a_k x + b_k y - 1 = 0, k \in \{1, \ldots, n\}$ not passing through the origin satisfying constraints Eq. (9) and Eq. (10), the boundary of the region $S$ generated by these lines, as defined in Eq. (8), is exactly the graph $G$ formed by $S_g$. Since that graph $G$ is an enclosing $n$
sided convex polygon with the origin in its interior under the conditions of Eq. (9) and Eq. (10), it follows that the boundary of the region $S$ generated by these lines is an enclosing $n$ sided convex polygon with the origin in its interior.

C. Proof of Corollary 1

This follows from the application of an affine coordinate transformation using $(\bar{x}, \bar{y})$. More precisely, given coordinates $x, y$, consider the transformation $(x^{\text{new}}, y^{\text{new}}) = (x - \bar{x}, y - \bar{y})$. In the new coordinates, the origin $(0^{\text{new}}, 0^{\text{new}})$ is located at $(\bar{x}, \bar{y})$. In addition, the representation of lines changes to

$$\left\{a(x - \bar{x}) + b(y - \bar{y}) - 1 = 0 : a, b \in \mathbb{R} \land a^2 + b^2 \neq 0 \right\}.$$ 

From Theorem 1 it follows that Eq. (9) and Eq. (10) are sufficient to guarantee that the boundary of the region $S := \left\{(x^{\text{new}}, y^{\text{new}}) : \sum_{k=1}^{n} a_k x^{\text{new}} + b_k y^{\text{new}} - 1 \leq 0 \right\}$ is an enclosing $n$ sided convex polygon with the origin $(0^{\text{new}}, 0^{\text{new}})$ in its interior. By changing back to the original coordinate system, the result follows.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation under Grants CMMI-21238612. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not reflect the views of NSF. The authors would like to thank Lin Li, Paul Lathrop from UC San Diego and Jun Xiang from San Diego State University for their valuable comments and discussions.

REFERENCES


