Fast Iterative Region Inflation for Computing Large 2-D/3-D Convex Regions of Obstacle-Free Space

Qianhao Wang[†], Zhepei Wang[†], Mingyang Wang, Jialin Ji,
Zhichao Han, Tianyue Wu, Rui Jin, Yuman Gao, Chao Xu, and Fei Gao[*]

***Abstract*—Convex polytopes have compact representations and exhibit convexity, which makes them suitable for abstracting obstacle-free spaces from various environments. Existing generation methods struggle with balancing high-quality output and efficiency. Moreover, another crucial requirement for convex polytopes to accurately contain certain seed point sets, such as a robot or a front-end path, is proposed in various tasks, which we refer to as manageability. In this paper, we propose Fast Iterative Regional Inflation (FIRI) to generate high-quality convex polytope while ensuring efficiency and manageability simultaneously. FIRI consists of two iteratively executed submodules: Restrictive Inflation (RsI) and Maximum Volume Inscribed Ellipsoid (MVIE) computation. By explicitly incorporating constraints that include the seed point set, RsI guarantees manageability. Meanwhile, iterative MVIE optimization ensures high-quality result through monotonic volume bound improvement. In terms of efficiency, we design methods tailored to the low-dimensional and multi-constrained nature of both modules, resulting in orders of magnitude improvement compared to generic solvers. Notably, in 2-D MVIE, we present the first linear-complexity analytical algorithm for maximum area inscribed ellipse, further enhancing the performance in 2-D cases. Extensive benchmarks conducted against state-of-the-art methods validate the superior performance of FIRI in terms of quality, manageability, and efficiency. Furthermore, various real-world applications showcase the generality and practicality of FIRI. The high-performance code of FIRI will be open-sourced.**

I. Introduction

In robotics, a key task is to navigate without collisions, which involves frequent interactions with environments abundant in vast amounts of discrete obstacle information. For this interaction requirement, convex polytopes provide a compact, structured geometric abstraction of feasible space, alleviating the burden of collision avoidance [1]–[3]. Furthermore, this abstraction facilitates large-scale space storage and enables topological analysis, including the construction of road maps [4] and convex covers [5]. Additionally, convex polytopes construct convex linear safety constraints from non-convex obstacles, which benefits problem formulation and solution. In fact, in several applications, this approach even transforms the problem into convex optimization, leading to the attainment of global optima [6], [7].

Although convex polytopes offer compact representation, generating satisfactory ones is far from a trivial task. First, a larger convex polytope extracts more information from the safety space, benefiting tasks that require spatial search within the safety space, such as trajectory planning. As illustrated in Fig. 1(a), larger convex polytopes facilitate a smoother trajectory. Therefore, we aim to maximize the polytope volume, which we define as its *quality* in this paper. Furthermore, maximizing computational *efficiency* enables applications in online high-speed tasks or with limited onboard resources. Despite extensive research [1], [2], [8], [9], balancing polytope quality and generation efficiency remains challenging. Existing approaches either generate high-quality regions but require substantial computational budget [1], [8], or achieve fast computation while yielding conservative results [2], [9].

In addition to quality and efficiency, several applications impose a crucial requirement for the generated convex polytope to accurately encompass a specified set of points, to which we refer as the *manageability*. For example, in trajectory planning shown in Fig. 1(b), the corridor-based approach [2], [10] uses blue path segments as seeds to generate convex hulls, forming a safety corridor. If the convex hulls fail to encompass the corresponding line segments, it may result in a discontinuity in the corridor, making it impossible to generate a continue trajectory within the corridor. Additionally, in whole-body planning [3], [11], insufficient coverage of the robot's shape can lead to similar planning failure due to the absence of feasible solution space, as illustrated in Fig.1(b). However, most existing algorithms prioritize optimizing the size of the region without adequately considering [1], [2] or ability to ensure manageability [9], [12].

To satisfy these requirements, we propose a novel algorithm, called **F**ast **I**terative **R**egion **I**nflation (FIRI) for computing free convex polytope, which simultaneously achieves high *quality*, high *efficiency*, and strong *manageability* for the first time. FIRI takes obstacles, a seed consisting of points that is required to be included, and an initial ellipsoid as inputs. It iteratively proceeds with two modules (detailed in Fig. 2):

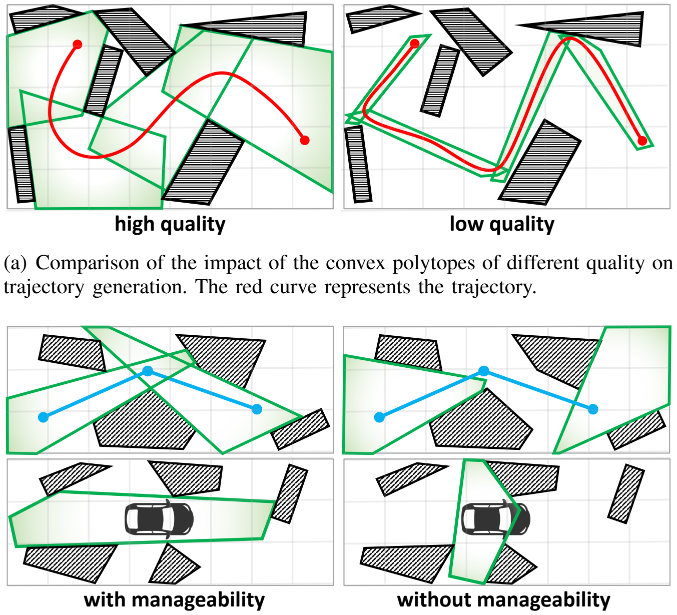
1) Restrictive Inflation (RsI): inflating the ellipsoid and using its contact planes tangent to obstacles to generate a set of halfspaces containing the seed, which separate a convex polytope from obstacles.
2) The Maximum Volume Inscribed Ellipsoid (MVIE) of the convex polytope is required to be calculated, which will be inflated in the next iteration.

MVIE serves as a lower bound of the volume of the convex polytope, which monotonically expands during the iterative update, leading to a growing obstacle-free region that ensures

† Equal contribution.

∗ Corresponding author.

All authors are with the State Key Laboratory of Industrial Control Technology, Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China and Huzhou Institute, Zhejiang University, Huzhou 313000, China. {qhwangaa, wangzhepei, fgaoaa}@zju.edu.cn



(a) Comparison of the impact of the convex polytopes of different quality on trajectory generation. The red curve represents the trajectory.

(b) Comparison of polytopes with or without manageability. **Top**: When generating polytopes based on the blue lines to construct safe corridor , the absence of manageability may lead to discontinuity in the generated corridor. **Bottom**: The absence of manageability during whole-body planning may result in the failure to generate convex polytopes that contain the robot.

Fig. 1. Illustration of quality and manageability. The gray polytopes represent obstacles, and the green polytopes represent the generated free polytopes.

the high quality of FIRI. The idea of monotonic updating the lower bound is inspired by Deits' fundamental work IRIS [1], which, however, lacks manageability. Additionally, its considerable computational overhead in complex environments [1], [9] limits its real-time applications. In contrast, FIRI significantly improves upon IRIS in terms of manageability and computational efficiency. For manageability, FIRI uses RsI to generate halfspaces that compose the polytope to necessarily exclude obstacles while containing the seed. For efficiency, we design specialized methods for FIRI's two optimization modules based on their geometric properties, resulting in a remarkable computational efficiency improvement with a speedup of orders of magnitude compared to IRIS.

For RsI, we convert the halfspace computation into a minimum-norm problem which is a strictly convex quadratic programming (QP). Considering its low-dimensional yet multi-constrained nature, we generalize Seidel's method [13], which is originally designed for solving linear programming with linear complexity, to handle with this minimum-norm. Compared with several general-purpose solvers [14]–[16] for QP, this method achieves the capability of obtaining analytical solution within a significantly shorter time.

MVIE, known as one of the most challenging extremal ellipsoid problems [17], is highly demanded in a number of applications [18], [19]. Most existing methods adopt semidefinite programming (SDP) formulations [1], [20] with interior point method variants for solution [17], [21]–[23]. However, they struggle with computational efficiency when handling massive constraint due to large equation systems required in each iteration. To reduce computational overhead without sacrificing solution quality, we present an equivalent second-order conic programming (SOCP) formulation for MVIE, bringing about a noticeable boost in computational efficiency.

Moreover, especially in 2-D scenarios, we gain ultra computational efficiency by proposing an analytical method for MVIE. The method exhibits a time complexity linear in the number of the hyperplanes of the input convex polytope, which is given for the first time to the best of our knowledge. As the dual problem of MVIE, Minimum Volume Enclosing Ellipse (MVEE) has long been equipped with linear-time analytical algorithms [24] based on its LP-type problem structure [25]. However, the corresponding approach for MVIE remained absent for decades due to failing LP-type problem properties and lacking analytical solutions for basis computation [26]. We address these challenges through an improved randomized algorithm with problem reformulation and a bottom-up strategy inspired by GJK algorithm's distance subalgorithm [27]. Consequently, the specialized 2-D method achieves a substantial efficiency gain compared to other state-of-the-art methods [1], [23], surpassing them by several orders of magnitude.

Building upon the above methods for RsI and MVIE, FIRI generates high-*quality* convex polytopes with *manageability* while maintaining high *efficiency*. To evaluate the performance of FIRI in these three requirements, we conduct comprehensive comparisons with various polytope generation algorithms [1], [2], [9]. The results provide compelling evidence that FIRI outperforms other approaches across all three requirements. Moreover, we extensively perform real-world applications to showcase the applicability of FIRI, which involves a 2-D vehicle with non-holonomic constraint and a 3-D quadrotor, as well as point-mass and whole-body model.

In summary, the contributions are:

1) RsI is introduced to ensure the manageability of the generated polytope. For its few-variable but rich-constraint nature, a specialized and efficient solver is designed.
2) A novel SOCP formulation for MVIE is proposed, avoiding the positive definite constraints and improving the computational efficiency.
3) Especially for 2-D MVIE, a linear-time analytical algorithm is introduced for the first time, further enabling ultra-fast computational performance.
4) Building upon the above methods, a novel convex polytope generation algorithm FIRI is proposed. Extensive experiments verify its superior comprehensive performance in terms of quality, efficiency, and manageability.

II. Related Work

A. Generating Free Convex Polytope

Deits et al. [1] propose Iterative Region Inflation by Semidefinite programming (IRIS) with the objective of generating the largest possible free convex polytope. IRIS involves inflating an ellipsoid, using it as a seed, to obtain a convex polytope formed by the intersection of contact planes. Then in the next iteration, the MVIE of the obtained convex polytope is selected as the new seed for inflation. However, this method requires solving semidefinite programming problems in each iteration to obtain the MVIE, which significantly hampers the computational efficiency. Furthermore, the hyperplane computation of IRIS cannot directly accommodate additional seed containment constraint, lacking manageability. In contrast, our RsI for FIRI unifies the forms of obstacle exclusion and seed containment constraints, enabling simpler and more efficient solutions. Wu et al. [28] use an IRIS-like method to obtain convex polytope through ellipsoid-based hyperplanes. They propose an ADMM-based approach that iteratively refines polytopes and optimizes trajectories to achieve shorter and better trajectory. Based on IRIS, Dai et al. [29] propose C-IRIS, which focuses on mapping collision in the task space to the configuration space based on kinematics and then generates certified safe convex hulls in the configuration space. Similarly, Mark et al. [30] propose IRIS-NP, extending IRIS to configuration space via nonlinear programming. Subsequently, building upon IRIS-NP, Werner et al. [31] significantly accelerate polytope generation by sampling nearby configuration-space obstacles. The above IRIS derivatives primarily target multi-joint robotic arms in configuration space, differing from our focus. To address the efficiency and local manageability deficiencies of IRIS, Liu et al. [2] propose the Regional Inflation by Line Search algorithm (RILS) which takes line segment as the seed input. RILS first generates a maximal ellipsoid that contains the segment yet excludes all obstacles. Then RILS inflates the ellipsoid to form a convex polytope with contact planes from the obstacles, which fundamentally aligns with the inflation step in a single iteration of IRIS. RILS shows high computational speed. Yet using input line as major axis of the ellipsoid, it tends to produce conservative convex hull. Specifically for voxel maps, Gao et al. [8] develop Parallel Convex Cluster Inflation algorithm. Starting from an unoccupied seed voxel, it grows incrementally in layers along coordinate axis using visibility, maintaining voxel set convexity. Despite parallel computing acceleration, this algorithm reaches near real-time only at low map resolution.

On the other hand, Savin et al. [12] utilize the concept of space inversion. Using spherical polar mapping from the input seed point, they flip all the obstacle points outside. After computing convex hull of the inverted obstacle points, they transform it back to the original space for the final hull. The strong nonlinearity of spherical polar mapping limits this method, as volume loss in inverted space creates gaps between final polytope and obstacles, yielding conservative results. To address this limitation, Zhong et al. [9] adopt sphere flipping mapping. Due to the properties of the hidden point remove operator inherent [32] in this mapping, this approach obtains the visible star-convex region around the seed. They then partition this star-convex region into final convex polytope through a heuristic method. Yet this approach uses Quickhull [33], which degrades with points distributed near the sphere after mapping. In addition, the heuristic partitioning yields conservative result.

Conclusively, existing methods consistently suffer from at least one of the following issues: inefficiency, conservatism, and lack of manageability, thereby limiting their practicality.

B. Maximum Volume Inscribed Ellipsoid (MVIE)

MVIE is also known as inner Löwner-John ellipsoid [34]. Nesterov et al. [22] utilize an interior-point algorithm with a specialized rescaling method on each iteration to achieve a polynomial time solution of MVIE, surpassing ellipsoid algorithm [35]. Khachiyan et al. [36] transform the problem into a sequence of subproblems with only linear constraints, constructed by using the barrier method. This approach requires fewer computations compared to Nesterov's method [22]. Then Anstreicher et al. [37] make improvements to both methods [22], [36] and demonstrate that computing an approximate analytic center of the polytope beforehand can reduce the complexity effectively. Zhang et al. [21] also provide a modification of Khachiyan's approach [36] by replacing the inefficient primal barrier function method with a primal-dual interior-point method to solve the subproblems. Additionally, instead of dealing with a number of subproblems, they propose a novel primal-dual interior-point method free of matrix variables to solve MVIE directly. Through nonlinear transformations, they eliminate the positive-definite constraint on the coefficient matrix of the ellipsoid during the iteration process and provide a proof demonstrating that these nonlinear transformations preserve the uniqueness of the solution. Building upon similar idea of eliminating matrix variables, Nemirovskii [38] reformulates MVIE as a saddle-point problem using Lagrangian duality. They then use path-following method for finding approximate saddle points based on Nesterov's self-concordance theory [22]. However, these interior-point methods struggle with scenarios having constraints far exceeding space dimension. In these scenarios which are the focus of this paper, the large-scale system of linear equations that these methods need to solve at each iteration prevents them from computing MVIE within an acceptable time. To address inefficiency, Lin et al. [39] employ the fast proximal gradient method [40] to introduce a first-order optimization-based approach for MVIE. Although this approach significantly improves efficiency, it lacks exact solutions due to approximating non-differentiable indicator with an one-sided Huber function for positive-definite constraints.

MVIE is the most challenging problem among the extremal ellipsoid problems [17]. Other extremal ellipsoid problems, such as Minimum Volume Enclosing Ellipsoid (MVEE), can be transformed into MVIE with a linear reduction, which is irreversible. Beyond interior-point methods, analytical solutions of MVEE can be obtained using a linear-time randomized method in 2-D case [24]. However, for the more challenging MVIE, the corresponding linear complexity algorithm are absent for several decades. This paper will address this gap.

III. Fast Iterative Region Inflation

A. Problem Formulation

Consider a convex seed in an obstacle-rich n-dimensional environment, where $n \in \{2, 3\}$. The geometrical shape of the seed is given by the $\mathcal{V}$-representation [41] of a convex polytope, i.e., convex combination of a finite number of points

$$\mathcal{Q} = \operatorname{conv}\{v_1, \dots, v_s\}, \tag{1}$$

in which $v_1, \dots, v_s \in \mathbb{R}^n$ are allowed to be redundant, which means that they are not required to only contain the extreme points of $\mathcal{Q}$. The obstacle region $\mathcal{O}$ is, albeit nonconvex,

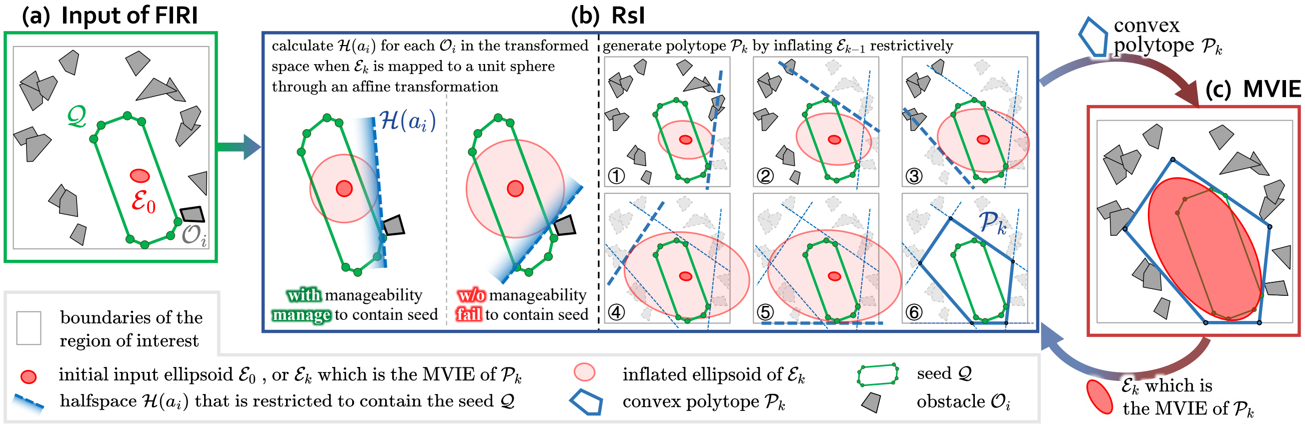

(a) Input of FIRI
$\mathcal{Q}$
$\mathcal{E}_0$
$\mathcal{O}_i$
(b) RsI
calculate $\mathcal{H}(a_i)$ for each $\mathcal{O}_i$ in the transformed space when $\mathcal{E}_k$ is mapped to a unit sphere through an affine transformation
$\mathcal{H}(a_i)$
with manageability
manage to contain seed
w/o manageability
fail to contain seed
generate polytope $\mathcal{P}_k$ by inflating $\mathcal{E}_{k-1}$ restrictively
①
②
③
④
⑤
⑥
$\mathcal{P}_k$
convex polytope $\mathcal{P}_k$
(c) MVIE
$\mathcal{E}_k$ which is the MVIE of $\mathcal{P}_k$
boundaries of the region of interest
initial input ellipsoid $\mathcal{E}_0$, or $\mathcal{E}_k$ which is the MVIE of $\mathcal{P}_k$
halfspace $\mathcal{H}(a_i)$ that is restricted to contain the seed $\mathcal{Q}$
inflated ellipsoid of $\mathcal{E}_k$
convex polytope $\mathcal{P}_k$
seed $\mathcal{Q}$
obstacle $\mathcal{O}_i$

Fig. 2. Overview of the computation process of FIRI, corresponding to the iterative modules RsI and MVIE calculation depicted in Algorithm. 1. The left diagram of (b) illustrates a specific example of halfspace computation w.r.t. $\mathcal{O}_i$ in the transformed space, comparing scenarios with and without manageability. ①-⑥ of (b) provide a visualization combined the process in Line 14-18. The increasing size of the inflated ellipsoid in (b) corresponds to the iterative search for the nearest halfspace in Line 14. The increasing number of halfspaces corresponds to Line 15, and the decreasing obstacles correspond to Line 16.

assumed to be the union of convex obstacles $\mathcal{O} = \cup_{i=1}^{N} \mathcal{O}_i$, of which the i-th one is determined by s_i points

$$\mathcal{O}_i = \operatorname{conv}\{u_{i,1}, \ldots, u_{i,s_i}\}. \quad (2)$$

which additionally enables FIRI to directly process polytope-type obstacles, as opposed to point-only methods [2], [9] that require discretization to handle such obstacles. We require no collision between the seed $\mathcal{Q}$ and the obstacle region $\mathcal{O}$, thus implying $\mathcal{Q} \cap \mathcal{O} = \varnothing$.

Our problem is to compute an obstacle-free convex polytope $\mathcal{P}$ which is required to contain the seed $\mathcal{Q}$ while excluding all obstacles $\mathcal{O}$. Besides, $\mathcal{P}$ should have the largest possible volume within a prescribed region of interest. For convenience, we define that the boundaries of the prescribed region are considered as obstacles and are encoded into $\mathcal{O}$, which makes the volume of obstacle-free space surrounding $\mathcal{Q}$ bounded. Concluding above requirements yields the optimization

$$\max_{\mathcal{P}} \ \operatorname{vol}(\mathcal{P}), \quad \text{s.t. } \mathcal{Q} \subseteq \mathcal{P}, \ \mathcal{O} \cap \operatorname{int}(\mathcal{P}) = \varnothing, \quad (3)$$

where $\operatorname{vol}(\mathcal{P})$ denotes the volume of the convex polytope $\mathcal{P}$ and $\operatorname{int}(\cdot)$ denotes the interior of a set. Note that the solution set of (3) will never be empty since the seed $\mathcal{Q}$ itself is already a feasible solution.

B. Algorithm Overview

Even if we simplify the optimization (3) by disregarding the constraint of including the seed, specifically $\mathcal{Q} \subseteq \mathcal{P}$, it will be transformed into a challenging problem, requiring $O(N^7)$ [42] in 2-D and NP-hard [43] in 3-D. Not to mention that bringing in this constraint further complicates the optimization (3). For efficiency-quality balance, we only seek a high-quality feasible solution to (3) efficiently instead of the global optimal solution. Additionally, since the computation of the objective function $\operatorname{vol}(\mathcal{P})$ of (3) is at least as hard as an NP-complete problem [44], we optimize a reasonable lower bound of $\operatorname{vol}(\mathcal{P})$, which is straightforward to evaluate, to maximize the original objective function. As adopted in [1], we also choose the volume of the MVIE of $\mathcal{P}$ as this lower bound.

To efficiently maximize the lower bound of the volume of the convex polytope $\mathcal{P}$ while satisfying the constraints of the original problem (3), we propose a novel algorithm called **F**ast **I**terative **R**egion **I**nflation (FIRI) as shown in Algorithm 1. Any ellipsoid that is strictly contained in $\mathcal{Q}$ can be used to initialize the algorithm. FIRI iteratively executes two modules: RsI and MVIE within its outer loop (Line 4-21). In the k-th iteration, the former module takes an ellipsoid as input and expands the ellipsoid to obtain a new convex polytope $\mathcal{P}_k$, and the latter module takes the convex polytope $\mathcal{P}_k$ as input and computes its MVIE $\mathcal{E}_k$. As illustrated in Fig. 2, the output of these two modules serves as the input for each other. Then we provide the details of these two modules and the convergence and manageability of FIRI.

In this paper, we define an ellipsoid $\mathcal{E}$ by

$$\mathcal{E} = \left\{ p \;\middle|\; p = A_{\mathcal{E}} D_{\mathcal{E}} x + b_{\mathcal{E}}, \ x \in \mathbb{R}^n, \ \|x\| = 1 \right\}, \quad (4)$$

where $A_{\mathcal{E}} \in \mathbb{R}^{n \times n}$ is orthonormal, $D_{\mathcal{E}} \in \mathbb{R}^{n \times n}$ is diagonal and positive-definite, and $b_{\mathcal{E}} \in \mathbb{R}^n$. The diagonal elements of $D_{\mathcal{E}}$ correspond to the lengths of the semi-axis of $\mathcal{E}$.

1) RsI: For each obstacle $\mathcal{O}_i$, we maximally inflate the input ellipsoid $\mathcal{E}$ under the constraint that there exists a halfspace which does not contain the obstacle yet includes both the inflated ellipsoid and seed $\mathcal{Q}$. Then we use these halfspaces to forms a new polytope as the output of RsI.

Specifically, $\mathcal{Q}$ and $\mathcal{O}_i$ are transformed (Line 8 and 10) by the inverse affine map determined by the ellipsoid $\mathcal{E}$ generated in the previous iteration. Since $\mathcal{Q}$ and $\mathcal{O}_i$ are both in $\mathcal{V}$-representation, their images after transformation are still the convex combinations of the images of their vertices, i.e.,

$$\bar{\mathcal{Q}} = \operatorname{conv}\{D_{\mathcal{E}}^{-1} A_{\mathcal{E}}^{\mathrm{T}} (v_j - b_{\mathcal{E}}), \ 1 \leq j \leq s\}, \quad (5)$$

$$\bar{\mathcal{O}}_i = \operatorname{conv}\{D_{\mathcal{E}}^{-1} A_{\mathcal{E}}^{\mathrm{T}} (u_{i,j} - b_{\mathcal{E}}), \ 1 \leq j \leq s_i\}. \quad (6)$$

It is important that in the transformed space the ellipsoid $\mathcal{E}$ is transformed to a unit ball $\mathcal{B}$ as

$$\mathcal{B} = \{x \in \mathbb{R}^n \mid \|x\| = 1\}, \quad (7)$$

whose collision check is far cheaper to handle than ellipsoids.

Then we calculate a restrictive halfspace for each transformed obstacle $\bar{\mathcal{O}}_i$ in the first inner loop (Line 10-11). Compared to the methods that lack the manageability to include the seed [1], [2], the proposed RsI requires that the

```
Algorithm 1: FIRI
  Notion:  number of obstacles $N$,
           parameters of ellipsoid $\mathcal{E}$: $A_{\mathcal{E}}, D_{\mathcal{E}}, b_{\mathcal{E}}$,
           halpspace $\mathcal{H}(a)$ defined in (9)
  Input:   seed $\mathcal{Q}$, obstacles $\mathcal{O}$, initial ellipsoid $\mathcal{E}_0$,
           threshold $\rho$
  Output:  convex polytope $\mathcal{P}$
1  begin
2      $k \leftarrow 0$
3      repeat
4          $k \leftarrow k+1$
5          /* RsI starts */
6          $\mathcal{I} \leftarrow \{1, \dots, N\}$
7          $\mathcal{E} \leftarrow \mathcal{E}_{k-1},\ \bar{\mathcal{P}} \leftarrow \mathbb{R}^n$
8          $\bar{\mathcal{Q}} \leftarrow D_{\mathcal{E}}^{-1} A_{\mathcal{E}}^{\mathrm{T}} (\mathcal{Q} - b_{\mathcal{E}})$
9          foreach $i \in \mathcal{I}$ do
10             $\bar{\mathcal{O}}_i \leftarrow D_{\mathcal{E}}^{-1} A_{\mathcal{E}}^{\mathrm{T}} (\mathcal{O}_i - b_{\mathcal{E}})$
11             $a_i \leftarrow \arg\max_{a \in \mathbb{R}^n} a^{\mathrm{T}} a,$
                   s.t. $\bar{\mathcal{Q}} \subseteq \mathcal{H}(a),\ \bar{\mathcal{O}}_i \cap \mathrm{int}(\mathcal{H}(a)) = \varnothing$
12         end
13         repeat
14             $j \leftarrow \arg\min_{i \in \mathcal{I}} a_i^{\mathrm{T}} a_i$
15             $\bar{\mathcal{P}} \leftarrow \bar{\mathcal{P}} \cap \mathcal{H}(a_j)$
16             $\mathcal{I} \leftarrow \mathcal{I} \backslash \left\{ i \in \mathcal{I} \mid \bar{\mathcal{O}}_i \cap \mathrm{int}(\mathcal{H}(a_j)) = \varnothing \right\}$
17         until $\mathcal{I} = \varnothing$
18         $\mathcal{P}_k \leftarrow A_{\mathcal{E}} D_{\mathcal{E}} \bar{\mathcal{P}} + b_{\mathcal{E}}$
19         /* RsI ends */
20         /* MVIE starts */
21         $\mathcal{E}_k \leftarrow \mathcal{E}^*(\mathcal{P}_k)$
22         /* MVIE ends */
23     until $\mathrm{vol}(\mathcal{E}_k) \le (1+\rho)\,\mathrm{vol}(\mathcal{E}_{k-1})$
24     return $\mathcal{P}_k$
25 end
```

halfspace contains $\bar{\mathcal{Q}}$ and the corresponding inflated ball $\mathcal{B}_i$ yet excludes $\bar{\mathcal{O}}_i$. Additionally, it aims to maximize the size of $\mathcal{B}_i$ which is defined by

$$\mathcal{B}_i = \left\{ x \in \mathbb{R}^n \mid \|x\| = a_i^{\mathrm{T}} a_i \right\}, \tag{8}$$

where a_i is the only contact point of the halfspace boundary with the inflated ball $\mathcal{B}_i$. The halfspace $\mathcal{H}(a_i)$ is defined as

$$\mathcal{H}(a_i) = \left\{ x \in \mathbb{R}^n \;\middle|\; a_i^{\mathrm{T}} x \le a_i^{\mathrm{T}} a_i \right\}. \tag{9}$$

The calculation of the restrictive halfspace can be written as an optimization as shown in Line 11, for which we provide an efficient solver in Sec. IV. In the left diagram of Fig. 2(b), we provide a specific example of the restrictive halfspace computation in the transformed space, comparing scenarios with and without manageability. Notably, in the scenario without manageability (employed by IRIS), the exclusive focus on maximizing ellipsoid inflation results in the computed halfspace being unable to guarantee seed point containment.

In the second inner loop (Line 14-16), we generate a new convex polytope based on the obtained halfspaces. We iteratively finds the closest halfspace $\mathcal{H}(a_i)$, adds it into $\bar{\mathcal{P}}$, and then remove the halfspaces corresponding to the obstacles outside of $\mathcal{H}(a_i)$. This widely adopted greedy strategy [1], [2], [31] is effective in reducing the number of halfspaces that constitute the generated convex polytope, aiming to minimize them as much as possible, compared to the large number of obstacles. Until all halfspaces are processed, an obstacle-free polytope in $\mathcal{H}$-representation [41] can be formed in the transformed space. Then a new polytope $\mathcal{P}_k$ is obtained by recovering $\bar{\mathcal{P}}$ to the original space (Line 18). Thus, the output of FIRI is an obstacle-free convex polytope in $\mathcal{H}$-representation, i.e., intersection of m halfspaces

$$\mathcal{P} = \left\{ x \in \mathbb{R}^n \;\middle|\; A_{\mathcal{P}} x \le b_{\mathcal{P}} \right\}, \tag{10}$$

where $A_{\mathcal{P}} \in \mathbb{R}^{m \times n}$ and $b_{\mathcal{P}} \in \mathbb{R}^m$. As shown in ①-⑥ of Fig. 2(b), we present a combined representation of the two processes mentioned above (Line 14-18).

2) MVIE: As long as a closed $\mathcal{P}$ has nonempty interior, its unique MVIE [45] $\mathcal{E}^*(\mathcal{P})$ can be determined by solving

$$\max_{A_{\mathcal{E}}, D_{\mathcal{E}}, b_{\mathcal{E}}} \ \mathrm{vol}(\mathcal{E}), \ \text{s.t. } \mathcal{E} \subseteq \mathcal{P}, \tag{11}$$

which is employed in Line 21. Notably, MVIE takes exclusively one convex polytope as input, independent of any obstacles used to generate the input polytope. To solve MVIE, we propose efficient methods in Sec. V and Sec. VI so that the computational overhead of MVIE will no longer be a stumbling block that prevents this monotonically inflating MVIE approach from being applied to real-time scenarios [1].

3) Manageability and Convergence of FIRI: During the iterative computation, FIRI always ensures the feasibility of the solution, which means the generated convex polytope $\mathcal{P}_k$ maintains satisfacting the constraints of the original problem as k increases. Moreover, it maintains the monotonicity of the volume of MVIE $\mathcal{E}_k$, which is the lower bound of the volume of the convex polytope $\mathcal{P}_k$. Then by analyzing the feasibility and monotonicity of the output of FIRI, we explain the manageability and convergence they bring about.

For the feasibility, since each halfspace computed in RsI (Line 11) satisfies the original constraints, the new convex polytope $\mathcal{P}_k$ formed by the intersection of these halfspaces is guaranteed to be feasible :

$$\mathcal{Q} \subseteq \mathcal{P}_k, \ \ \mathcal{O} \cap \mathrm{int}(\mathcal{P}_k) = \varnothing \tag{12}$$

which gives the manageability of FIRI. Additionally, as defined in Sec III-A, the prescribed region of interest is bounded, whose boundaries are encoded into $\mathcal{O}$, thus the feasibility also indicates that RsI always generates a closed polytope $\mathcal{P}_k$.

For the monotonicity, in the k-th iteration, we inflate the unit ball $\mathcal{B}$ for each transformed obstacle $\bar{\mathcal{O}}_i$ in RsI, which always ensures the contact point a_i satisfies $\|a_i\| \ge 1$. Thus, in the transformed space, $\bar{\mathcal{P}}$ acquired from the second inner loop (Line 14-16) always holds $\mathcal{B} \subseteq \bar{\mathcal{P}}$, which means that in the original space (Line 18) we have $\mathcal{E}_{k-1} \subseteq \mathcal{P}_k$. As $\mathcal{E}_k$ is the largest ellipsoid contained by $\mathcal{P}_k$, we have

$$\mathrm{vol}(\mathcal{E}_{k-1}) \le \mathrm{vol}(\mathcal{E}_k). \tag{13}$$

which gives the monotonicity.

Then a sequence $\{\mathcal{P}_1, \mathcal{E}_1, \dots, \mathcal{P}_k, \mathcal{E}_k, \dots\}$ is generated by repeating the iteration. Since $\mathrm{vol}(\mathcal{E}_k)$ is non-decreasing and bounded, the ellipsoid volume will converge to a finite value.

Consequently, FIRI terminates when this lower bound for $\mathrm{vol}(\mathcal{P}_k)$ cannot be sufficiently improved (Line 23).

In conclusion, RsI brings manageability to FIRI, while monotonic iterative updates optimize the polytope volume's lower bound for a high-quality output. Notably, the efficiency of FIRI relies strongly on the performance of solving two optimizations in Line 11 and 21. In subsequent sections, we propose efficient and reliable subalgorithms exploiting the geometric structure of these optimizations, greatly benefiting the computational efficiency of FIRI.

IV. Solving Restrictive Halfspace Computation in RsI via SDMN

A. Reformulation of Restrictive Halfspace Computation

In this section, we focus on designing efficient method for the computation of halfspaces in RsI, defined in Line 11 of Algorithm 1. Combining the definition of the halfspace $\mathcal{H}(a_i)$ in (9), we formulate the halfspace calculation into

$$\max_{a_i \in \mathbb{R}^n} \ a_i^{\mathrm{T}} a_i, \tag{14a}$$

$$\text{s.t.} \ \ v^{\mathrm{T}} a_i \le a_i^{\mathrm{T}} a_i, \ \forall v \in \bar{\mathcal{Q}}, \tag{14b}$$

$$u^{\mathrm{T}} a_i \ge a_i^{\mathrm{T}} a_i, \ \forall u \in \bar{\mathcal{O}}_i. \tag{14c}$$

Algorithm 1 keeps $a_i^{\mathrm{T}} a_i > 0$, which ensures that the origin always lies within the interior of the halfspace $\mathcal{H}(a_i)$. Thus, although such a maximization of the inflated ball $\mathcal{B}_i$ is nonconvex, we can obtain its equivalent minimum-norm formulation through its polar duality. Specifically, we reformulate the problem with a new variable b by substituting $a_i = b/(b^{\mathrm{T}} b)$ and obtain an equivalent L_2-norm minimization

$$\min_{b \in \mathbb{R}^n} \ b^{\mathrm{T}} b, \tag{15a}$$

$$\text{s.t.} \ \ v^{\mathrm{T}} b \le 1, \ \forall v \in \bar{\mathcal{Q}}, \tag{15b}$$

$$u^{\mathrm{T}} b \ge 1, \ \forall u \in \bar{\mathcal{O}}_i, \tag{15c}$$

which has a low-dimension but multi-constraint nature. In reformulation (15), both types of constraints share a unified inequality form. This formulation handles cases with and without the seed containment constraint, allowing subsequent solver SDMN adapt readily. In contrast, prior work such as IRIS [1] struggles to add seed containment while maintaining the original problem structure, making it challenging to achieve manageability via straightforward constraint addition.

B. Solution to Small-Dimensional Minimum-Norm with Massive Constraints

For efficiently solving the new formulation (15) of restrictive halfspace computation, we propose an analytical method for such **S**mall **D**imensional **M**inimum-**N**orm, called SDMN. This method generalizes Seidel's randomized algorithm [13] from Linear Programming (LP) to minimum-norm, and enjoys complexity linear in the constraint number.

Without loss of generality, we consider the following general small-dimensional minimum-norm problem,

$$\min_{y \in \mathbb{R}^n} y^{\mathrm{T}} y, \ \text{s.t.} \ Ey \le f, \tag{16}$$

where $E \in \mathbb{R}^{d \times n}$, $f \in \mathbb{R}^d$, d denotes the number of constraints, which is much larger than n. Hereafter, we present a randomized algorithm with linear complexity for the small-dimensional minimum-norm (16) in Algorithm 2.

Algorithm 2: SDMN

```
   Notion: constraints already checked $\mathcal{I}$,
           input and output of the recursive call ${\mathcal{H}_E}'$, $y'$
   Input:  set of halfspace constraints $\mathcal{H}_E$
   Output: $y$
 1 begin
 2     $y \leftarrow \mathbf{0}$
 3     if $\dim(\mathcal{H}_E) == 1$ then
 4         $y \leftarrow$ OneDimMinNorm($\mathcal{H}_E$)
 5         return $y$
 6     end
 7     $\mathcal{I} \leftarrow \{\}$
 8     foreach $h \in \mathcal{H}_E$ in a random order do
 9         if $y \notin h$ then
10             /* Detail in Sec. IV-B2 */
11             $\{M, v, {\mathcal{H}_E}'\} \leftarrow$ HouseholderProj($\mathcal{I}, h$)
12             $y' \leftarrow$ SDMN(${\mathcal{H}_E}'$)
13             $y \leftarrow My' + v$
14         end
15         $\mathcal{I} \leftarrow \mathcal{I} \cup \{h\}$
16     end
17     return $y$
18 end
```

As shown in Algorithm 2, this is a recursive algorithm. In the following, we first provide an outline of the algorithm, then we give a detailed description of how to construct the recursive problem, and finally present a complexity analysis.

1) Algorithm Outline: We denote $\mathcal{H}_E$ as the set of the hyperplanes corresponding to the constraints of the L_2-norm minimization (16), and denote $\mathcal{I}$ as the set of the constraints that have already been checked (Line 15). Algorithm 2 starts with $y = \mathbf{0}$ which is the solution of an unconstrained L_2-norm minimization (Line 2), and then gradually checks the constraint $h \in \mathcal{H}_E$ in a random order (Line 8). We check whether the solution under the constraints $\mathcal{I}$ violates the new constraint h. If it is not violated, the next constraint will be checked. If it is violated, h must be active. That is, the solution must be on the boundary of h, thus the minimization (16) under the constraints $\mathcal{I} \cup \{h\}$ can be written as

$$\min_{y \in \mathbb{R}^n} \ y^{\mathrm{T}} y, \tag{17a}$$

$$\text{s.t.} \ \ E_{\mathcal{I}} y \le f_{\mathcal{I}}, \tag{17b}$$

$$E_h y = f_h, \tag{17c}$$

where $E_{\mathcal{I}}$ and $f_{\mathcal{I}}$ represent the coefficients corresponding to the hyperplanes in $\mathcal{I}$, E_h and f_h represent the coefficients of h. Using the geometric structure of the problem (17), we transform it into a subproblem of $(n-1)$ dimensional L_2-norm minimization with the same form as (16) in Line 11, which is described in detail in Sec. IV-B2. This transformation allows for a recursive call of Algorithm 2 (Line 12). Additionally, when $n = 1$, the problem (16) is equivalent to a problem of computing the smallest absolute value in a interval (Line 4),

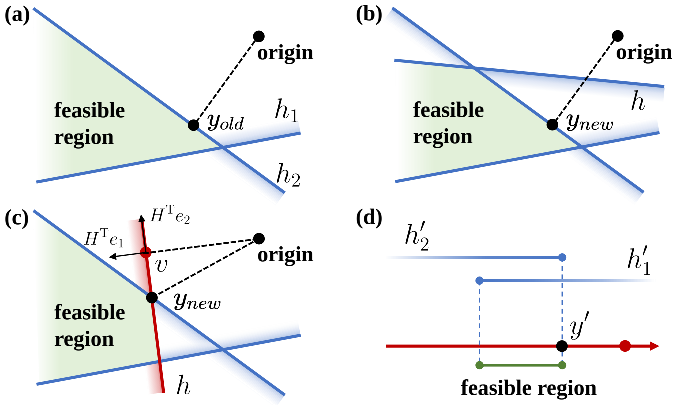


Fig. 3. Illustration of a specific instance of Algorithm 2 in 2-D for Sec. IV-B1 and Sec. IV-B2. **For Sec. IV-B1**: **(a)**: Both the inequality constraints h_1 and h_2 have been checked, which means $\mathcal{I} = \{h_1, h_2\}$. y_{old} is the solution of the 2-D L_2-norm minimization under $\mathcal{I}$. **(a)⇒(b)**: When y_{old} does not violate the newly added constraint h, $y_{new} = y_{old}$. **(a)⇒(c)**: When y_{old} violates the new constraint h, we need to find a new solution y_{new} on the constraint plane corresponding to h, implying equation constraint (17). **For Sec. IV-B2**: The vector v is normal to the constraint plane, and $H^{\mathrm{T}}e_1$, $H^{\mathrm{T}}e_2$ are a set of orthogonal basis of the 2-D space, where $H^{\mathrm{T}}e_2 \perp v$. **(d)**: We establish a new coordinate for the 1-D subspace on the constraint plane with v as the origin and $H^{\mathrm{T}}e_2$ as the orthogonal basis, transform the checked constraints h_1, h_2 to this coordinate as h_1', h_2'. Finally, we transforms (c) into a 1-D L_2-norm minimization with only inequality constraints (23).

whose solution can be calculated trivially. We assume that the subproblem can be successfully solved, and thus the new solution under the constraints $\mathcal{I} \cup \{h\}$ can be calculated (Line 13). Subsequently, the next constraint can be checked until all constraints in the set $\mathcal{H}_E$ are examined, from which we obtain the result of Algorithm 2. To provide an intuitive perception, we present an example of the violation check in Fig. 3, where we use y_{old} and y_{new} to distinguish between solutions under $\mathcal{I}$ and $\mathcal{I} \cup \{h\}$.

2) Recursive Problem Construction: An essential process of Algorithm 2 is the recursive problem construction (Line 11-12), which transforms (17) with an equality constraint into a $(n-1)$ dimensional subproblem with the same form as the original problem (16). This process's efficiency and stability greatly impact Algorithm 2. Since L_2-norm is invariant under the orthogonal transformation, we establish a new $(n-1)$ dimensional Cartesian coordinate system on the constraint plane (17c) to implement the construction of the reduced dimensional recursive problem. We provide a specific example of this process in Fig. 3(c) and 3(d). As illustrated in Fig. 3(c), we take the point of L_2-norm minimization on the constraint plane as the origin of the coordinate system,

$$v = \frac{f_h E_h^{\mathrm{T}}}{E_h E_h^{\mathrm{T}}}. \tag{18}$$

It is obvious that any point y on the constraint plane satisfies

$$y^{\mathrm{T}}y = (y-v)^{\mathrm{T}}(y-v) + v^{\mathrm{T}}v, \tag{19}$$

which means that, with the constraint (17c), the L_2-norm minimization of y is equivalent to the L_2-norm minimization of $(y-v)$ which can be transformed into an $(n-1)$ dimensional vector in the newly established coordinate system.

We construct the new coordinate system by finding an orthogonal matrix $H \in \mathbb{R}^{n\times n}$ that satisfies the requirement

$$Hv \propto e_j, \tag{20}$$

where $e_j \in \mathbb{R}^n$ is an unit vector whose j-th element is 1. We provide the details of the choice of j and the computation of H at the end of Sec. IV-B2. The requirement (20) indicates that the set of all column vectors of orthogonal H^{T} except the j-th one form an orthonormal basis for the orthogonal complement of v, as shown in Fig. 3(c). Since v is normal to the constraint plane, we adopt this orthonormal basis to define the new coordinate system of the $(n-1)$ dimensional subspace on the constraint plane, and introduce the corresponding coordinates $y' \in \mathbb{R}^{n-1}$. We denote $M \in \mathbb{R}^{n\times(n-1)}$ as the matrix obtained by removing the j-th column of H^{T}. Then y in the origin coordinate system corresponding to y' can be computed as

$$y = My' + v. \tag{21}$$

Based on the decomposition (19) and (21), minimizing the L_2-norm of y on the constraint plane is equivalent to minimizing

$$y^{\mathrm{T}}y - v^{\mathrm{T}}v = {y'}^{\mathrm{T}}M^{\mathrm{T}}My' = {y'}^{\mathrm{T}}y'. \tag{22}$$

Eventually, for (17) with an equality constraint, we construct its corresponding $(n-1)$ dimensional L_2-norm minimization with only inequality constraints on the constraint plane as

$$\min_{y' \in \mathbb{R}^{n-1}} \; {y'}^{\mathrm{T}}y', \tag{23a}$$

$$\text{s.t. } E_{\mathcal{I}}My' \leq f_{\mathcal{I}} - E_{\mathcal{I}}v. \tag{23b}$$

As Fig. 3(d) shows, the linear inequality constraints on y' can be obtained by substituting (21) into the inequality constraint (17b) of the original problem, and the set of their corresponding hyperplanes is denoted as $\mathcal{H}_E{}'$ (Line 11-12). Obviously, (23) has the same structure as (16) and has a lower dimension, thus it can be solved by recursively calling Algorithm 2 (Line 12), which gives rise to the formation of a recursive algorithmic structure.

Here we provide the details of j and H in (20). We obtain the orthogonal matrix H by Householder reflection [46]. v is a normal vector of the constraint plane proportional to the geometric scale of the problem, which has intuitive numerical stability. Thus, we use the normal vector v to compute the Householder reflection. First, we set j as the index of the element of v with the largest absolute value as

$$j = \mathrm{argmax}_{k\in\{1,\ldots,n\}} \|v^{\mathrm{T}}e_k\|. \tag{24}$$

Then the reflection vector u that transforms v to be parallel to e_j can be calculated by

$$u = v + \mathrm{sgn}(v_j)\|v\|e_j, \tag{25}$$

whose corresponding Householder matrix can be calculated by

$$H = \mathbf{I}_n - \frac{2uu^{\mathrm{T}}}{u^{\mathrm{T}}u}, \tag{26}$$

which is an orthogonal matrix and satisfies the aforementioned requirement (20). The use of the obtuse reflection vector u corresponding to the opposite direction of the largest absolute value element of v prevents the Householder transformation from being ill-conditioned due to a small reflection angle. This procedure is implicitly equivalent to a single-step operation of the Householder QR factorization, which has higher numerical stability than the Gram-Schmidt orthogonalization which may fall victim to the catastrophic cancelation problem [47].

3) Complexity Analysis: To conclude this section, we give the complexity analysis of the randomized Algorithm 2. Each iteration may trigger recursive calls to $(n-1)$ dimensional subproblems. For small dimension n with large constraint scale d, new constraints rarely become active, making recursive calls tiny. The expected complexity is $O(n!d)$ [13], so that in the common dimensions $n \in \{2, 3\}$, the expected complexity increases only linearly with the constraint scale[1]. In addition, random constraint permutation preprocessing ensures performance independent of input order, nearly always achieving expected linear complexity. With the help of this randomized algorithm, the complexity of handling all obstacles in the RsI of Algorithm 1 grows linearly only with the total number of vertices of the obstacles and the seed.

V. Solving SOCP-Reformulation of MVIE via Affine Scaling Algorithm

In this section, we propose an efficient algorithm for MVIE with a low dimension but massive constraints, required in FIRI (Line 21 of Algorithm 1). By carefully handling the orthogonal constraints on the elliptical matrix and the objective function involving the matrix determinant, we reformulate MVIE into a Second-Order Conic Programming (SOCP) form. Using Affine Scaling [48], we solve this SOCP efficiently for MVIE.

A. SOCP-Reformulation of MVIE

To begin with, we concretize the definition of the original abstract MVIE problem (11). According to the coefficients of the ellipsoid $\mathcal{E}$ defined in (4), the diagonal elements of $D_\mathcal{E}$ are the lengths of the semi-axis of $\mathcal{E}$. Thus the objective $\mathrm{vol}(\mathcal{E})$ of the problem (11) is proportional to the determinant $\det(D_\mathcal{E})$ [49]. Moreover, the semi-infinite constraint $\mathcal{E} \subseteq \mathcal{P}$ is equivalent to $[[A_\mathcal{P} A_\mathcal{E} D_\mathcal{E}]^2 \mathbf{1}]^{\frac{1}{2}} \le b_\mathcal{P} - A_\mathcal{P} b_\mathcal{E}$ [20] which is a second-order cone (SOC) constraint, where $A_\mathcal{P}$, $b_\mathcal{P}$ are the coefficients of the halfspace constituting the convex polytope $\mathcal{P}$ defined in (10), and $[\cdot]$ implies entry-wise operations. Then the problem (11) can be written as

$$\max_{A_\mathcal{E}, D_\mathcal{E}, b_\mathcal{E}} \ \det(D_\mathcal{E}), \tag{27a}$$

$$\text{s.t.} \ \ [[A_\mathcal{P} A_\mathcal{E} D_\mathcal{E}]^2 \mathbf{1}]^{\frac{1}{2}} \le b_\mathcal{P} - A_\mathcal{P} b_\mathcal{E}, \tag{27b}$$

$$D_\mathcal{E} = \mathrm{diag}(d_\mathcal{E}), \ d_\mathcal{E} \in \mathbb{R}^n_{\ge 0}, \tag{27c}$$

$$A_\mathcal{E}^\mathrm{T} A_\mathcal{E} = I_n, \tag{27d}$$

where $\mathrm{diag}(\cdot)$ indicates either constructing a diagonal matrix or taking all diagonal entries of a square matrix. However, this program still imposes orthonormality constraints on $A_\mathcal{E}$. It is worth mentioning that if we use $A_\mathcal{E} D_\mathcal{E}$ as the decision variable, (27) becomes an SDP with a challenging orthogonal constraint, which is often used to solve MVIE [1], [20], [23].

Then we eliminate the orthogonal constraints (27d). Noting that $A_\mathcal{E} D_\mathcal{E}^2 A_\mathcal{E}^\mathrm{T}$ is always positive definite for the optimal solution of the non-degenerate problem (27). Thus the Cholesky factorization $A_\mathcal{E} D_\mathcal{E}^2 A_\mathcal{E}^\mathrm{T} = L_\mathcal{E} L_\mathcal{E}^\mathrm{T}$ is unique [50] for this solution, where $L_\mathcal{E}$ is a lower triangular matrix with positive diagonal entries. If we treat $L_\mathcal{E}$ as decision variables, since $A_\mathcal{E}$ is orthonormal, the objective (27a) and constraints (27b) can be written as

$$\det(D_\mathcal{E}) = \sqrt{\det(A_\mathcal{E} D_\mathcal{E}^2 A_\mathcal{E}^\mathrm{T})} = \det(L_\mathcal{E}), \tag{28}$$

$$[[A_\mathcal{P} A_\mathcal{E} D_\mathcal{E}]^2 \mathbf{1}]^{\frac{1}{2}} = [[A_\mathcal{P} L_\mathcal{E}]^2 \mathbf{1}]^{\frac{1}{2}}. \tag{29}$$

Consequently, the orthonormality constraint (27d) on $A_\mathcal{E}$ is avoided, and (27) is equivalent to

$$\max_{L_\mathcal{E}, b_\mathcal{E}} \ \det(L_\mathcal{E}), \tag{30a}$$

$$\text{s.t.} \ \ [[A_\mathcal{P} L_\mathcal{E}]^2 \mathbf{1}]^{\frac{1}{2}} \le b_\mathcal{P} - A_\mathcal{P} b_\mathcal{E}, \tag{30b}$$

$$L_\mathcal{E} \text{ is lower triangular.} \tag{30c}$$

Now we simplify the objective function $\det(L_\mathcal{E})$ which is the product of the diagonal entries. Denoting the hypograph of geometric mean by

$$\mathcal{K}_{1/n} = \left\{ (x, t) \in \mathbb{R}^n_{\ge 0} \times \mathbb{R} \ \middle| \ (x_1 \cdots x_n)^{\frac{1}{n}} \ge t \right\}, \tag{31}$$

then maximizing the ellpsoid $\mathcal{E}$ is equivalent to maximizing a new variable t with the constraint $(\mathrm{diag}(L_\mathcal{E}), t) \in \mathcal{K}_{1/n}$. As for the constraints (27b), we denote the SOC as $\mathcal{K}_n$ and describe the Cartesian product of m SOC as $\mathcal{K}_n^m$,

$$\mathcal{K}_n = \left\{ (t, x) \in \mathbb{R} \times \mathbb{R}^{1 \times n-1} \ \middle| \ t \ge \|x\| \right\}, \tag{32}$$

$$\mathcal{K}_n^m = \mathcal{K}_n \times \cdots \times \mathcal{K}_n \subseteq \mathbb{R}^{m \times n}. \tag{33}$$

Then the optimization (30) is formulated as

$$\max_{t, L_\mathcal{E}, b_\mathcal{E}} \ t, \tag{34a}$$

$$\text{s.t.} \ \ (\mathrm{diag}(L_\mathcal{E}), t) \in \mathcal{K}_{1/n}, \tag{34b}$$

$$(b_\mathcal{P} - A_\mathcal{P} b_\mathcal{E}, A_\mathcal{P} L_\mathcal{E}) \in \mathcal{K}_{n+1}^m, \tag{34c}$$

$$L_\mathcal{E} \text{ is lower triangular,} \tag{34d}$$

where m denotes the number of halfspaces composing $\mathcal{P}$ defined in (10). Additionally, $\mathcal{K}_{1/n}$ can also be represented by SOC using additional $O(n)$ variables and cones of $\mathcal{K}_3$ [20]. Eventually, we transform (34) into an optimization with only SOC form constraints and a simple linear objective function.

B. Affine Scaling for Solving SOCP with Massive Constraints

As we reformulate MVIE into a pure SOCP from (34), for brevity, we denote it as

$$\min_{x} \ c_\mathcal{K}^\mathrm{T} x, \tag{35a}$$

$$\text{s.t.} \ \ (c_i^\mathrm{T} x + d_i, \ x^\mathrm{T} A_i) \in \mathcal{K}_{n_i}, \ 1 \le i \le \bar{m}, \tag{35b}$$

where $x \in \mathbb{R}^{\bar{n}}$, $c_\mathcal{K} \in \mathbb{R}^{\bar{n}}$, $c_i \in \mathbb{R}^{\bar{n}}$, $d_i \in \mathbb{R}$ and A_i are all constant. The new decision variable x consists of all lower triangular elements of $L_\mathcal{E}$, $b_\mathcal{E}$, t of (34), and the elements added to deal with constraint (34b) in form of hypograph of geometric mean. We have $\bar{n} = O(n^2)$ because an n

[1] For specific derivation regarding the complexity, readers can refer to Theorem 1 of Seidel's work [13].

dimensional ellipsoid already has $n(n+3)/2$ variables. For convenience, we set t as the $\bar{n}$-th element of x, then we have

$$c_{\mathcal{K}} = (0, 0, ..., 0, -1)^{\mathrm{T}} \in \mathbb{R}^{\bar{n}}. \tag{36}$$

$\bar{m} = O(m+n)$ indicates the amount of the SOC constraints. For the constraints in (35b) corresponding to the constraints of (34b) which are represented by additional cones of $\mathcal{K}_3$, $A_i \in \mathbb{R}^{\bar{n}\times 2}$ and $n_i = 3$. For the constraints in (35b) corresponding to the origin constraints of (34c), $A_i \in \mathbb{R}^{\bar{n}\times n}$ and $n_i = n+1$.

Then we aim to efficiently solve the SOCP (35). Although the dimension n of the original problem (11) is limited to 2 or 3, the large constraint number m results in a huge number of constraints $\bar{m}$ in the newly constructed SOCP. To tackle this multi-constraint SOCP, we generalize Affine Scaling (AS) [48], an interior-point method originally used for LP, to SOCP. This algorithm has an closed-form update step of each iteration and exhibits superlinear convergence [51]. At each iteration, AS uses the logarithmic barrier function of the constraints to compute a strictly feasible region and calculates the optimal update step within this region. For (35), its logarithmic barrier function $\phi(x)$ is defined as

$$\phi(x) = -\sum_{i=1}^{\bar{m}} \log\left(f_i(x)\right), \tag{37}$$

$$f_i(x) = \left(c_i^{\mathrm{T}} x + d_i\right)^2 - x^{\mathrm{T}} A_i A_i^{\mathrm{T}} x. \tag{38}$$

Then the Hessian of $\phi(x)$ is given by

$$H_\phi(x) = \sum_{i=1}^{\bar{m}} \frac{1}{f_i(x)^2} \nabla f_i(x) \nabla f_i(x)^{\mathrm{T}} - \sum_{i=1}^{\bar{m}} \frac{1}{f_i(x)} \nabla^2 f_i(x), \tag{39}$$

where the gradient and Hessian of $f_i(x)$ are

$$\nabla f_i(x) = 2\left(c_i^{\mathrm{T}} x + d_i\right) c_i - 2 A_i A_i^{\mathrm{T}} x, \tag{40}$$

$$\nabla^2 f_i(x) = 2\left(c_i c_i^{\mathrm{T}} - A_i A_i^{\mathrm{T}}\right). \tag{41}$$

Now we obtain the strictly feasible region of AS at $(k+1)$-th iteration based on the feasible k-th solution x_k as

$$\left\{x \in \mathbb{R}^n \mid (x - x_k)^{\mathrm{T}} H_\phi(x_k)(y - x_k) \le 1\right\}, \tag{42}$$

which is an ellipsoidal region as $H_\phi(x_k)$ is positive definite. Then the update step of (35) can be given by

$$x_{k+1} = x_k - \tau \frac{H_\phi^{-1} c_{\mathcal{K}}}{\sqrt{c_{\mathcal{K}}^{\mathrm{T}} H_\phi^{-1} c_{\mathcal{K}}}}, \tag{43}$$

where $\tau \in (0, 1]$ is the step size.

VI. Solving 2-D MVIE Analytically via A Linear-time Complexity Algorithm

Inspired by the linear-time algorithm of the dual problem Minimum Volume Enclosing Ellipsoid (MVEE) [24], in this section, we focus on the construction of a linear-time algorithm for 2-D MVIE, leveraging its special LP-type problem [25] structure. We begin by providing the necessary background knowledge related to LP-type problems and analyzing the limitations of existing general solutions for applying to MVIE in Sec. VI-A. Then we address these limitations and propose our algorithm with linear time complexity in Sec. VI-B. Finally we provide the analytic solution of the subproblems required in the proposed algorithm in Sec. VI-C.

A. Background of LP-type Problem

Let us consider an abstract minimization [25] specified by pairs $(\mathcal{H}, w)$, where $\mathcal{H}$ is a finite set of constraints and $w : 2^{\mathcal{H}} \to \mathcal{W}$ is a value function that maps subsets of $\mathcal{H}$ to values in a ordered set $(\mathcal{W}, <)$, which has a unique minimum value $-\infty$. For the sake of simplicity in subsequent descriptions, we define finite sets $\mathcal{G}$, $\mathcal{F}$ and a constraint h, which satisfy $\mathcal{G} \subseteq \mathcal{F} \subseteq \mathcal{H}$, $h \in \mathcal{H}$. The problem with $(\mathcal{H}, w)$ can be considered as an LP-type problem as long as the following two properties are satisfied [25]: for any $\mathcal{G}, \mathcal{F}$ and h we have

- *Monotonicity*: $w(\mathcal{G}) \le w(\mathcal{F})$,
- *Locality*: with $-\infty < w(\mathcal{G}) = w(\mathcal{F})$, if $w(\mathcal{F}) < w(\mathcal{F} \cup \{h\})$, then $w(\mathcal{G}) < w(\mathcal{G} \cup \{h\})$.

Three important definitions of LP-type problem are given:

- $w(\mathcal{H})$ is called the *value* of $\mathcal{H}$,
- constraint h is *violated* by $\mathcal{H}$, if $w(\mathcal{H}) < w(\mathcal{H} \cup \{h\})$,
- the *basis* of $\mathcal{H}$ is the minimal subset of $\mathcal{H}$ with the same value of $\mathcal{H}$.

Then two primitive operations is defined: given a basis $\mathcal{B}$,

- *Violation test*: determine whether $w(\mathcal{B}) < w(\mathcal{B} \cup \{h\})$, for a constraint $h \notin \mathcal{B}$,
- *Basis computation*: compute the basis of $\mathcal{B} \cup \{h\}$ when h is violated by the basis $\mathcal{B}$.

To give readers an intuitive understanding, we provide a specific example: for the dual problem MVEE [24], $\mathcal{H}$ is the input points set and $w(\mathcal{H})$ is the area of the minimum ellipse that can contain all the input points.

The goal of LP-type problem is to compute the basis of the input set $\mathcal{H}$ and its corresponding value $w(\mathcal{H})$. A generalized algorithm for the LP-type problem is given by Matoušek et al. [26]. The algorithm operates by randomly selecting a constraint h from the input set $\mathcal{H}$ and a known basis $\mathcal{B} \subseteq \mathcal{H}$ and performing a violation test. If a violation is detected, a basis of $\mathcal{B} \cup \{h\}$ is computed, then the algorithm is recursively called with the new basis and the set of checked constraints. This algorithm provides an efficient solution to LP-type problems with finite primitive operations, whose expected number is linear in the input set number $|\mathcal{H}|$, thanks to its randomized recursive structure. We refer the reader interested in this complexity conclusion to Matoušek's work [26].

However, for 2-D MVIE, the aforementioned generalized algorithm described above is not available for two reasons: **i)** In MVIE, the subset of input constraints may not form a closed polygon, which leads to an undefined ellipse. Using the volume of the ellipse, similar to its dual problem MVEE, to define the value function $w(\cdot)$ is not feasible. **ii)** There is currently no known method for performing analytical basis computation directly for any subsets in MVIE.

B. Randomized Maximal Inscribed Ellipse Algorithm

In this subsection, we address the aforementioned challenges, and then we present a linear-time complexity algorithm by improving the generalized algorithm [26] for 2-D MVIE.

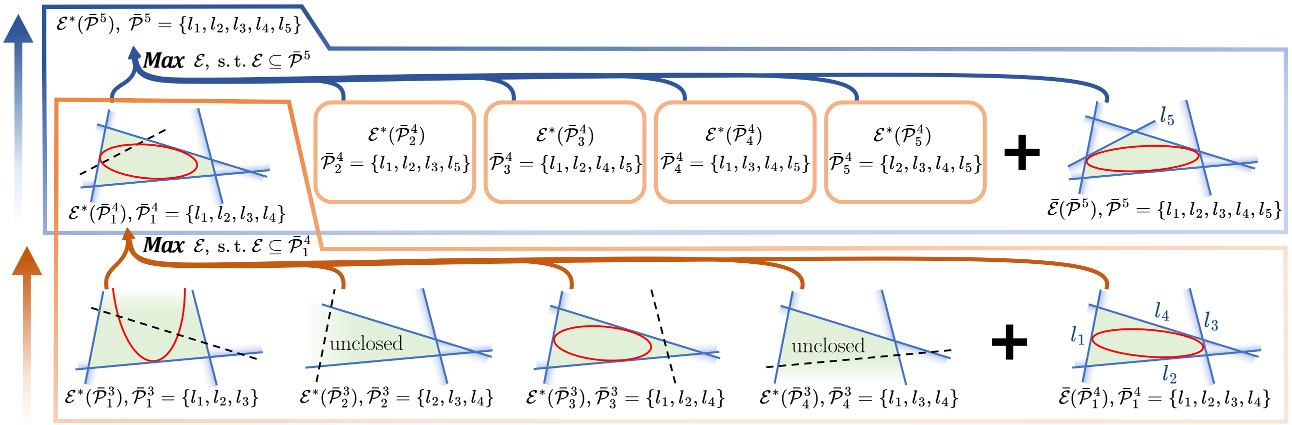

$\mathcal{E}^*(\bar{\mathcal{P}}^5)$, $\bar{\mathcal{P}}^5 = \{l_1, l_2, l_3, l_4, l_5\}$
Max $\mathcal{E}$, s. t. $\mathcal{E} \subseteq \bar{\mathcal{P}}^5$
$\mathcal{E}^*(\bar{\mathcal{P}}_1^4), \bar{\mathcal{P}}_1^4 = \{l_1, l_2, l_3, l_4\}$
$\mathcal{E}^*(\bar{\mathcal{P}}_2^4)$ $\bar{\mathcal{P}}_2^4 = \{l_1, l_2, l_3, l_5\}$
$\mathcal{E}^*(\bar{\mathcal{P}}_3^4)$ $\bar{\mathcal{P}}_3^4 = \{l_1, l_2, l_4, l_5\}$
$\mathcal{E}^*(\bar{\mathcal{P}}_4^4)$ $\bar{\mathcal{P}}_4^4 = \{l_1, l_3, l_4, l_5\}$
$\mathcal{E}^*(\bar{\mathcal{P}}_5^4)$ $\bar{\mathcal{P}}_5^4 = \{l_2, l_3, l_4, l_5\}$
+
l_5
$\bar{\mathcal{E}}(\bar{\mathcal{P}}^5), \bar{\mathcal{P}}^5 = \{l_1, l_2, l_3, l_4, l_5\}$
Max $\mathcal{E}$, s. t. $\mathcal{E} \subseteq \bar{\mathcal{P}}_1^4$
unclosed
unclosed
+
l_4
l_3
l_1
l_2
$\mathcal{E}^*(\bar{\mathcal{P}}_1^3), \bar{\mathcal{P}}_1^3 = \{l_1, l_2, l_3\}$
$\mathcal{E}^*(\bar{\mathcal{P}}_2^3), \bar{\mathcal{P}}_2^3 = \{l_2, l_3, l_4\}$
$\mathcal{E}^*(\bar{\mathcal{P}}_3^3), \bar{\mathcal{P}}_3^3 = \{l_1, l_2, l_4\}$
$\mathcal{E}^*(\bar{\mathcal{P}}_4^3), \bar{\mathcal{P}}_4^3 = \{l_1, l_3, l_4\}$
$\bar{\mathcal{E}}(\bar{\mathcal{P}}_1^4), \bar{\mathcal{P}}_1^4 = \{l_1, l_2, l_3, l_4\}$

Fig. 4. Illustration of a specific instance of the bottom-up strategy. The large orange box specifies the process described by (49) for calculating the MVIE of $\bar{\mathcal{P}}_1^4$ based on the MVIE of its subsets $\bar{\mathcal{P}}_j^3, 1 \leq j \leq 4$ and the MENN of itself. The red ellipses indicate MVIE $\mathcal{E}^*$ or MENN $\bar{\mathcal{E}}$, depending on the equation beneath each subset. Using the subset in the lower left corner as an example, the solid blue lines indicate elements in the subset $\bar{\mathcal{P}}_1^3$, the black dashed lines indicate elements in $\bar{\mathcal{P}}_1^4$ that are not included in the subset $\bar{\mathcal{P}}_1^3$. Similarly, the blue box illustrates the process of calculating the MVIE of $\bar{\mathcal{P}}^5$.

1) Value Function: We design an additional evaluation of the proximity to closure for unclosed case and use the MVIE volume for closed case to construct a new value function that satisfies the properties of the LP-type problem. We define the value domain $\mathcal{W}$ in lexicographical ordering set $(\mathcal{W}, <)$. We denote $<$ as the lexicographical ordering on $\mathbb{R}^2$, that is, for $\forall x = (x_1, x_2)^{\mathrm{T}}, y = (y_1, y_2)^{\mathrm{T}} \in \mathbb{R}^2$, $x < y$, if $x_1 < y_1$, or $x_1 = y_1$ and $x_2 < y_2$. This ordering is extended to $\mathbb{R}^2 \cup \{-\infty\}$ by the definition that $-\infty$ is the unique minimum value. Considering the frequent failure to form closed polygon for the subsets of constraints, we define the value function as

$$w(\mathcal{G}) = \begin{cases} -\infty & \textit{if } \mathcal{G} = \varnothing \\ \bar{w}_u(\mathcal{G}) & \textit{if } \mathcal{G} \textit{ is unclosed} \\ \bar{w}_c(\mathcal{G}) & \textit{if } \mathcal{G} \textit{ is closed} \end{cases}, \tag{44}$$

where $\bar{w}_u(\mathcal{G})$ represents a measure of the proximity to closure when $\mathcal{G}$ is not closed. To define $\bar{w}_u(\mathcal{G})$, we first define a new set $\mathcal{G}_n$ that does not contain duplicate elements, and all elements in the new set $\mathcal{G}_n$ come from the unit normal vectors corresponding to the halfspace constrain elements in the set $\mathcal{G}$. Then we define $\bar{w}_a(\mathcal{G})$ in lexicographical ordering by

$$\bar{w}_u(\mathcal{G}) = (-a_p(\mathcal{G}_n), \ \min\{|\mathcal{G}_n|, 3\}), \tag{45}$$

where $a_p(\mathcal{G}_n)$ denotes the angle of polar cone of $\mathcal{G}_n$. Such a particular definition is intended to deal with the corner cases that arise due to the parallel elements in $\mathcal{G}$.

As for $\bar{w}_c(\mathcal{G})$ when $\mathcal{G}$ is closed, we define it by

$$\bar{w}_c(\mathcal{G}) = (\mathrm{area}(\mathcal{E}^*(\mathcal{G}))^{-1}, \ \xi), \tag{46}$$

whose first element represents the inverse of the MVIE's area of $\mathcal{G}$, and the second element ξ serves to extend this function to $\mathbb{R}^2$. The first element of $\bar{w}_c(\mathcal{G})$ is always greater than 0, and the first element of $\bar{w}_u(\mathcal{G})$ is always less than or equal to 0. Thus no matter what value ξ takes, we have $\bar{w}_c > \bar{w}_u$. Now based on this value function that satisfies monotonicity and locality, whose proof is detailed in Appendix A and B, MVIE can be treated as an LP-type problem.

2) Basis Computation: To perform analytical basis computation for MVIE, we decompose MVIE into several subproblems that can be solved analytically, then MVIE of $\mathcal{P}$ defined in (11) is equivalent to

$$\mathcal{E}^*(\mathcal{P}) = \max_{\bar{\mathcal{P}}^N \subseteq \mathcal{P}} \mathrm{area}\left(\bar{\mathcal{E}}(\bar{\mathcal{P}}^N)\right), \ \text{s.t. } \bar{\mathcal{E}} \subseteq \mathcal{P}, \tag{47}$$

where $\mathcal{P}$ is the input polygon while simultaneously denoting the finite set of the halfspaces whose intersection constitutes the input polygon in this section. Its subset containing N elements is denoted as $\bar{\mathcal{P}}^N$. To prevent potential misunderstanding, it should be emphasized that the subset $\bar{\mathcal{P}}^N$ is derived from $\mathcal{P}$, the sole input of MVIE, and is independent of the obstacles used to generate $\mathcal{P}$ in RsI. Additionally, $\bar{\mathcal{E}}(\bar{\mathcal{P}}^N)$ is the subproblem defined by

$$\bar{\mathcal{E}}(\bar{\mathcal{P}}^N) = \max_{\mathcal{E}} \ \mathrm{area}(\mathcal{E}), \tag{48a}$$

$$\text{s.t.} \ \ \mathcal{E} \subseteq \bar{\mathcal{P}}^N, \tag{48b}$$

$$\mathcal{E} \text{ is tangent to } h, \ \forall \ h \in \bar{\mathcal{P}}^N. \tag{48c}$$

Since at least 3 halfspaces are required to form a closed space in 2-D and an ellipse has only 5 degrees of freedom, we require $3 \leq N \leq 5$. When $\bar{\mathcal{P}}^N$ is closed and has no redundant element, (48) has an analytical solution, and its result $\bar{\mathcal{E}}$ is the **M**aximal **E**llipse tangent to ***N*** edges of the ***N***-gon formed by $\bar{\mathcal{P}}^N$, denoted as **MENN**, detailed in Sec. VI-C. Notably, $\mathcal{E}^*$ and $\bar{\mathcal{E}}$ are used to denote MVIE and MENN, respectively. For the two cases where subproblems (48) are infeasible, we handle them with special treatment: **i)** If $\bar{\mathcal{P}}^N$ is unclosed, we define its $\bar{\mathcal{E}}$ as ∞. **ii)** If there are redundant elements in a closed $\bar{\mathcal{P}}^N$, that is, these elements are not any side of the formed closed polygon. At this point, it is impossible for these elements to satisfy (48c), then we also define its $\bar{\mathcal{E}}$ as ∞. Based on the subproblem (48), MVIE (47) can be solved through enumerating subproblems (48) which has analytical solution.

Inspired by the distance algorithm of GJK [27], we organize the enumeration in an orderly way by using a bottom-up strategy, about which we illustrate a detailed example in Fig. 4. In Fig. 4, we use subscript to distinguish between different subsets of a set, for instance, $\bar{\mathcal{P}}_1^4$ denotes one of the subsets of

Algorithm 3: MaxEllipse

```
   Notion: combinatorial dimension $\delta = 5$
   Input: $\mathcal{H}$: input set, $\mathcal{X}$: a subset of the basis of $\mathcal{H}$
   Output: $\mathcal{B}$: basis of $\mathcal{H}$, $v_{\mathcal{B}}$: value of $\mathcal{B}$
1  begin
2      $v_{\mathcal{X}} \leftarrow w(\mathcal{X})$   /* defined in (44) */
3      if $|\mathcal{X}| \geq \delta$ then
4          return $\mathcal{X}, v_{\mathcal{X}}$
5      end
6      $\mathcal{S} \leftarrow \{\}$
7      $\mathcal{B} \leftarrow \mathcal{X}$
8      $v_{\mathcal{B}} \leftarrow v_{\mathcal{X}}$
9      foreach $h \in \mathcal{H}$ in a random order do
10         if ViolateTest$(h, v_{\mathcal{B}}, \mathcal{X})$ then
11             $\mathcal{B}, v_{\mathcal{B}} \leftarrow$ MaxEllipse$(\mathcal{S}, \mathcal{X} \cup \{h\})$
12         end
13         $\mathcal{S} \leftarrow \mathcal{S} \cup \{h\}$
14     end
15     return $\mathcal{B}, v_{\mathcal{B}}$
16 end
```

$\bar{\mathcal{P}}^5$. Specifically regarding the bottom-up strategy, for a subset $\bar{\mathcal{P}}_1^4$ containing 4 elements, we compute its MVIE by

$$\begin{aligned} \mathcal{E}^*(\bar{\mathcal{P}}_1^4) = \max_{\mathcal{E}} \ & \text{area}(\mathcal{E}), && \text{(49a)} \\ \text{s.t.} \ & \mathcal{E} \subseteq \bar{\mathcal{P}}_1^4, && \text{(49b)} \\ & \mathcal{E} \in \{\bar{\mathcal{E}}(\bar{\mathcal{P}}_1^4)\} \cup \{\mathcal{E}^*(\bar{\mathcal{P}}_j^3)\}, 1 \leq j \leq 4, && \text{(49c)} \end{aligned}$$

where $\bar{\mathcal{P}}_j^3$ denotes a subset, which contains 3 elements, of $\bar{\mathcal{P}}_1^4$. Notably, the MVIE of a closed $\bar{\mathcal{P}}_j^3$ is exactly its MENN [52]. This equation (49) corresponds to the process in the large orange box in Fig. 4: among the MVIE for each subset $\bar{\mathcal{P}}_j^3$, and the MENN of $\bar{\mathcal{P}}_1^4$, we select the largest one that satisfies the constraint (49b) as the MVIE of $\bar{\mathcal{P}}_1^4$. Based on the result we can build upwards MVIE of $\mathcal{P}^5$ containing 5 elements in a similar way, as illustrated in the blue box of Fig. 4. It should be emphasized that for the actual input $\mathcal{P}$, which typically consists of more than 5 halfspaces, its MVIE is also computed through a similar bottom-up enumeration: calculating the MVIE of all its subsets $\bar{\mathcal{P}}^5$ and selecting the largest ellipse contained within $\mathcal{P}$. To summarize, by connecting the above processes through the bottom-up strategy, now we can compute the MVIE of the input $\mathcal{P}$ based entirely on its subsets' MENN ($3 \leq N \leq 5$) which can be solved analytically in an efficient enumeration order.

3) Algorithm Overview: Building upon the aforementioned efforts, we improve the general framework [26] to propose a randomized algorithm for solving 2-D MVIE as shown in Algorithm 3. For the initial call to Algorithm 3, we set $\mathcal{H} \leftarrow \mathcal{P}$ and $\mathcal{X} \leftarrow \{\}$. The recursive framework (Line 11) of the algorithm, combined with the previously mentioned bottom-up strategy, results in an effect: when a recursive call occurs, it indicates that all subsets of the input $\mathcal{X}$ have already been checked, and for any of its subsets $\bar{\mathcal{X}}$, there is always an element $h \in \mathcal{X}$ violated by $\bar{\mathcal{X}}$. Therefore, if the input $\mathcal{X}$ is closed, we can directly solve the MENN of the polygon formed by $\mathcal{X}$ for the value $w(\mathcal{X})$ in Line 2. And if $\mathcal{X}$ is unclosed, it is easy to use (45) to compute its value. Recalling the complexity conclusion we mentioned in Sec. VI-A, now we can obtain the result of Algorithm 3 by finite number of violation tests (Line 10), MENN (for closed case) and (45) (for unclosed case), whose expected number is linear in the input set number.

In addition, the maximum cardinality of any basis is denoted as combinatorial dimension δ, and based on the value function (44) we have $\delta = 5$. This implies that in Line 2, we will only encounter MENN calculations for triangles, quadrilaterals, and pentagons, which also consistent with our decomposition of MVIE in (48). The combinatorial dimension δ in Line 3 ensures that the maximum number of recursive iterations in the algorithm remains a constant.

The analytical solution method (detailed in Sec. VI-C) may lead to incorrect result when the input is redundant. Thus, as mentioned in Sec. VI-B2, by treating redundant cases separately and assigning them a solution of ∞, we prevent them from influencing the overall algorithm. In Algorithm 3, we add this feature in ViolateTest (Line 10): in addition to the evaluation involved h and $v_{\mathcal{B}}$ based on (44), in the case where h is violated by $\mathcal{B}$, if $\mathcal{X} \cup \{h\}$ forms a convex N-gon but $N < |\mathcal{X} \cup \{h\}|$ (redundant case), then false is returned. Since we perform checks based on the bottom-up strategy, the non-redundant closed subsets (if they exist) of the redundant closed set $\mathcal{X} \cup \{h\}$ will definitely be checked in the algorithm. Thus there is no need to check with the redundant closed set.

Now we construct the linear complexity algorithm for 2-D MVIE. Another factor that affects the practical efficiency of the algorithm is the solution of MENN (48), for which we present efficient analytical solutions in Sec. VI-C.

C. Maximal Ellpise tangent to N edges of the N-gon

As demanded in Algorithm 3, in this section we present the analytical computation of MENN, which is the maximal inscribed ellipse that is tangent to all edges of arbitrary convex N-gon, $N \in \{3, 4, 5\}$. For arbitrary non-degenerate triangle [52], convex quadrilateral [53] or convex pentagon [45], there exists a unique such ellipse. Since the inputs from Algorithm 3 are non-degenerate and convex, in the following we default to the existence and uniqueness of the MENN.

Referring to the definition in (4), in the 2-D case we define the point $p \in \mathbb{R}^2$ on the boundary of the ellipse $\mathcal{E}$ to satisfy

$$(p - b_{\mathcal{E}})^{\mathrm{T}} (A_{\mathcal{E}} D_{\mathcal{E}}^2 A_{\mathcal{E}}^{\mathrm{T}})^{-1} (p - b_{\mathcal{E}}) = 1, \tag{50}$$

We denote the *homogeneous coordinate* [54] of point p as $\hat{p} = \left(p^{\mathrm{T}}, \ 1\right)^{\mathrm{T}}$. Since $A_{\mathcal{E}} D_{\mathcal{E}}^2 A_{\mathcal{E}}^{\mathrm{T}}$ is positive definite symmetric, (50) is a second-degree polynomial equation in two elements of p. We transform (50) into a quadratic form, then the ellipse $\mathcal{E}$ can be described in homogeneous coordinate as

$$\mathbb{P} = \left\{\hat{p} \in \mathbb{R}^3 \;\middle|\; \hat{p}^{\mathrm{T}} M_P \hat{p} = 0\right\}, \tag{51}$$

where the coefficient matrix $M_P \in \mathbb{R}^{3\times 3}$ is symmetric.

For the MENN problem in this section, it is intractable to solve the problem by (51) which demand the points of tangency. Thus we adopte to leverage the information of the tangents directly, based on the polarity of points and lines with respect to the ellipse $\mathcal{E}$ [55]. First, we require an algebraic characterization of a line. Specifically, in the projective plane, given a point with a homogeneous coordinate $\hat{p}$, the line passing through the point $\hat{p}$ can be denote in the form of *line corrdinate* [54] as $l \in \mathbb{R}^3$ that satisfies

$$\hat{p}^{\mathrm{T}} l = 0, \tag{52}$$

based on which, the calculation of the line coordinate can be performed by the Grassmanian expansion [56]. According to the polarity of ellipse [55], when $\hat{p}$ is on the ellipse $\mathcal{E}$ (51), the line coordinate l of the line tangent to $\mathcal{E}$ at $\hat{p}$ is given by

$$l = M_P \hat{p}. \tag{53}$$

Since the ellipse is not degenerate, M_P is invertible. Based on (51, 53), similarly to the representation by a set of the points in (51), the ellipse $\mathcal{E}$ now can be described by the set of all its tangents in line coordinate as

$$\mathbb{L} = \left\{ l \in \mathbb{R}^3 \mid l^{\mathrm{T}} M_L l = 0 \right\}, \tag{54}$$

$$M_L \propto {M_P}^{-1}. \tag{55}$$

Now we can calculate the MENN by utilizing the information of the tangents directly. Specifically, we first compute the line coordinates of edges of the polygon for solving M_L, then obtain M_P, and eventually get the ellipse in the desired form of (4). This process involves using (52), (54), (55), (50) and (51) sequentially. In the following, we present the implementation details for different N.

1) MENN of a Convex Pentagon: For (54), the symmetric M_L has 6 variables. Bringing the line coordinates of the five edges of the convex pentagon into (54) respectively, we can obtain a six-element homogeneous system of linear equations consisting of five equations. Since there exists and only exists unique ellipse tangent to all edges of the convex pentagon [45], the homogeneous system always has a non-trivial solution. Then based on the solution, we can obtain the MENN progressively through the process aforementioned.

2) MENN of a Convex Quadrilateral: In contrast to convex pentagon, a convex quadrilateral only provides four tangents. Thus there are a unique one-parameter set of inscribed ellipses that are tangent to all edges of the quadrilateral, whose parameter can be taken to be a prescribed point contact on any single edge of the quadrilateral [45] as shown in Fig. 5. Additionally, only one of them has the largest area [53], which is the MENN of the quadrilateral.

For simplicity of calculation, as shown in Fig. 5, we translate the quadrilateral so that one of its vertices coincides with the origin and one of the edges connected to that vertex coincides with the x-axis, which will not change the shape of the quadrilateral. Then we denote the line coordinate of the coinciding edge as l_λ. Inspired by the work of Hayes [57], we introduce new constraint by using the point of tangency on l_λ, and denote the point as $\hat{p}_\lambda$. Then we have

$$\hat{p}_\lambda = (\lambda, 0, 1)^{\mathrm{T}}, \quad l_\lambda = (0, 1, 0)^{\mathrm{T}}, \tag{56}$$

where λ is a variable. Combining (53) and (55), their polarity relationship can be written as

$$M_L l_\lambda \propto \hat{p}_\lambda, \tag{57}$$

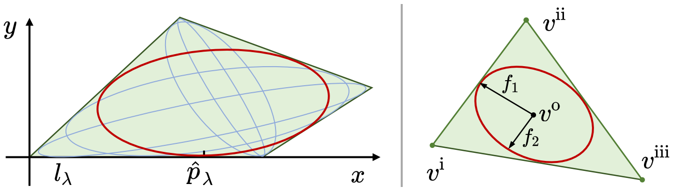


Fig. 5. **Left**: Illustration of the calculation of the MENN of a convex quadrilateral. The blue and red ellipses represent the ellipses obtained by sampling different points $\hat{p}_\lambda$ on the coinciding edge l_λ. **Right**: Illustration of the calculation of the MENN of a triangle.

which is an independent new constraint on M_L. The subsequent operations are similar to Sec. VI-C1, with the difference that eventually we calculate the area [57] of $\mathcal{E}$ as a function $A_{\mathcal{E}}(\lambda)$ in terms of λ. The optimal λ corresponding to the MENN of the quadrilateral can be computed by calculating the zeros of the first order derivative of $A_{\mathcal{E}}(\lambda)$. This calculation only involves solving a quadratic equation w.r.t. λ, which can be solved quickly and analytically. Due to space limitations, we do not delve into further details of the calculation.

3) MENN of a Triangle: The Steiner inellipse tangent to the edges of the triangle at their midpoints, is the MENN [52]. As shown in Fig. 5, we denote the 2-D Cartesian coordinates of the vertices of the triangle as $v^* = (v_0^*, v_1^*)^{\mathrm{T}}, * = \{\mathrm{i}, \mathrm{ii}, \mathrm{iii}\}$. The center v^o and two conjugate diameters $f_1, f_2 \in \mathbb{R}^2$ of the Steiner inellipse can be written as

$$v^o = \frac{1}{3}(v^{\mathrm{i}} + v^{\mathrm{ii}} + v^{\mathrm{iii}}), \tag{58}$$

$$f_1 = \frac{1}{2}(v^o - v^{\mathrm{iii}}), \; f_2 = \frac{1}{2\sqrt{3}}(v^{\mathrm{i}} - v^{\mathrm{ii}}). \tag{59}$$

Then the parameters required in (4) can be calculated by

$$b_{\mathcal{E}} = v^o, \; A_{\mathcal{E}} D_{\mathcal{E}}^2 A_{\mathcal{E}}^{\mathrm{T}} = \begin{pmatrix} f_1 & f_2 \end{pmatrix} \begin{pmatrix} f_1 & f_2 \end{pmatrix}^{\mathrm{T}}. \tag{60}$$

VII. Evaluation and Benchmark

To comprehensively demonstrate the outstanding advantages of FIRI, in terms of efficiency, quality, and manageability, we design extensive benchmarks comparing FIRI against several state-of-the-art convex polytope generation algorithms. Additionally, the efficiency of FIRI lies in the methods that we develop specifically for the two optimizations (Line 11 and 21 of Algorithm 1). To demonstrate their effectiveness, we conduct comparative experiments for these methods which are proposed for solving minimum-norm with small dimension but massive constraints and for solving MVIE, respectively. All benchmarks run on Intel Core i7-12700KF CPU, Linux, using C++14 without hardware acceleration.

A. Comparison of Generating Free Convex Polytope

Based on the description and analysis of several state-of-the-art algorithms for generating free convex polytopes in Sec. II, we benchmark the proposed FIRI against IRIS [1], Galaxy [9] and RILS [2]. Since Gao's method [8] relies on modeling the environment as a grid map and can only operate in near real-time with low map resolution, it is not compared here. Additionally, Galaxy can be regarded as an enhancement of Savin's method [12], both of which are based on space inversion, thus we directly adopt Galaxy for benchmark.

TABLE I
Comparison of Adaptability of Different Methods for Generating Free Convex Polytope

Adaptability Method	Seed Type			Obstacles Type	
	Point	Line	Polytope	Point	Polytope
FIRI	✓	✓	✓	✓	✓
IRIS [1]	✓	✗	✗	✓	✓
Galaxy [9]	✓	✗	✗	✓	✗
RILS [2]	✓	✓	✗	✓	✗

TABLE II
Input Obstacles Number of Different Scenarios

Scenario		Input Obstacle Number			
		avg	std	min	max
2-D	Sparse	246.7	111.0	44	525
	Medium	1157.6	277.8	553	1683
	Dense	3007.5	515.2	1443	4048
3-D	Sparse	453.6	119.6	303	670
	Medium	2677.8	613.6	1524	3929
	Dense	12659.0	764.9	11032	14536

For IRIS and our proposed method FIRI, we set the same parameter $\rho = 0.02$, that is, the stopping condition of them is that the volume of the MVIE of the convex hull obtained in this iteration grows less than 2% from the last iteration. For Galaxy and RILS, we used the default parameters.

To begin with, we compare the capability of these methods to adapt to different types of seed and obstacle inputs. As evidenced by the results in Tab. I, FIRI demonstrates the highest level of adaptability to various inputs. In Tab. II, the term of adaptability of polytope-type obstacles denotes the ability to handle polytope obstacles directly without discretization. As polytope representation is a widely adopted approach for environmental modeling [58], [59], the capability to process polytope obstacles directly is worthwhile. Subsequently, based on the reported adaptability, we conduct extensive benchmarks to validate the superior performance of FIRI.

1) Manageability: As analyzed in Sec. I, manageability is crucial in many applications. For instance, there are situations where we require the convex hull to contain the line segment of the path generated by the frontend [10], or during whole body planning, we demand the convex hull to encompass the robot [11]. Thus we compare the manageability of various methods by using point, line, and convex polytope as the seed. Although the type of obstacles has no effect on manageability, for fairness, the obstacles input are characterized by points.

We conduct benchmark in a complex environment of $50 \times 50\ m$ size (with a height of $10m$ for the 3-D case), where we generate random obstacles by using Perlin noise [60]. For each test, we randomly generate a collision-free seed in the environment as input. The boundary of each convex hull generation algorithm is constrained to be a square (cube for 3-D) with side length $6\ m$ centered on the seed's center and parallel to the coordinate axis. And the obstacle input is the points within the boundary of the square in the map. When the seed input is a line, we set its length to $1.5\ m$. When the seed input is a convex polytope, we set it as a $1.5 \times 0.75 \times 0.25\ m$ rectangular for the 3-D case, or a $1.5 \times 0.75\ m$ rectangle for the 2-D case. We generate the experimental environments with different obstacle densities, whose numbers of input obstacles are shown in Tab. II. For each density, we create different 10 environments. Furthermore, for each environment, we conduct 500 random trials for each type of seed input.

To be fair, we perform several adjustments in different seed cases based on Tab. I. When the seed is a line, for Galaxy and IRIS, which can only use one point as the seed input, we use both the endpoints and midpoints of the line as seed inputs for them to compute 3 convex polytopes. We then select the convex hull with the highest degree of line containment as the result. When the seed is a convex polytope, FIRI can directly take the polytope as input. IRIS and Galaxy take each vertex and the center point of the seed as inputs to compute multiple corresponding convex hulls. For RILS, which is adapted to use a line as input, we compute multiple convex hulls using the seed's diagonals as inputs. For each of the above three methods, we choose the convex hull that maximizes the inclusion of the seed as the respective result.

We calculate the success rate in generating a convex polytope that fully contains the seed, as shown in Tab. III. In addition, for a more intuitive presentation, we present an example of the results of each method in an application of whole-body planning that has a need for manageability. As shown in Fig. 12(b), we use a rectangular robot as the seed, and the details of the application will be described in Sec. VIII-A2. Due to greedy approach of IRIS in seeking the largest possible volume of the convex hull, it performs poorly

TABLE III
Success Rate of Different Methods for Generating Convex Polytopes Containing Seed across Different Scenarios and Seed Types

Scenario		Success Rate [%]											
		Point Seed				Line Seed				Polytope Seed			
		FIRI	IRIS [1]	Galaxy [9]	RILS [2]	**FIRI**	IRIS [1]	Galaxy [9]	RILS [2]	**FIRI**	IRIS [1]	Galaxy [9]	RILS [2]
2-D	Sparse	**100**	98.4	100	100	**100**	74.7	81.6	100	**100**	88.9	95.8	97.0
	Medium	**100**	97.0	100	100	**100**	53.9	64.2	100	**100**	79.7	91.1	95.3
	Dense	**100**	96.6	100	100	**100**	48.5	39.1	100	**100**	69.7	73.9	90.6
3-D	Sparse	**100**	99.1	100	100	**100**	96.5	79.1	100	**100**	96.2	85.1	98.0
	Medium	**100**	97.8	100	100	**100**	78.2	61.3	100	**100**	70.6	78.0	88.3
	Dense	**100**	96.2	100	100	**100**	59.6	47.0	100	**100**	37.3	45.1	70.6

TABLE IV
Comparison of Computation Time of Different Methods for Generating Free Convex Polytope

Seed Type	Scenario	Method	Computation Time $[ms]$											
			Sparse				Medium				Dense			
			avg	std	min	max	avg	std	min	max	avg	std	min	max
Point	2-D	**FIRI**	**0.038**	**0.013**	**0.013**	**0.069**	**0.120**	**0.035**	**0.032**	**0.189**	**0.273**	**0.068**	**0.127**	**0.451**
		IRIS [1]	34.444	2.978	24.997	40.125	37.730	2.982	28.001	46.620	39.671	5.111	20.848	48.155
		Galaxy [9]	0.069	0.013	0.042	0.099	0.123	0.023	0.077	0.168	0.233	0.038	0.143	0.327
		RILS [2]	0.011	0.004	0.005	0.021	0.037	0.010	0.016	0.061	0.082	**0.017**	0.027	0.122
		FIRI(SI)	**0.008**	**0.003**	**0.002**	**0.017**	**0.032**	**0.010**	**0.014**	**0.052**	**0.066**	0.021	**0.016**	**0.103**
	3-D	**FIRI**	**0.143**	**0.035**	**0.069**	**0.237**	**0.660**	**0.224**	**0.259**	**1.259**	**2.116**	**0.560**	**1.109**	**3.535**
		IRIS [1]	34.638	7.083	16.966	76.988	55.724	15.327	11.173	103.176	87.897	20.823	14.826	167.441
		Galaxy [9]	0.144	0.054	0.054	0.328	1.337	0.467	0.445	2.476	5.916	1.079	4.321	9.185
		RILS [2]	0.044	0.017	0.020	0.113	0.346	0.165	0.120	0.658	1.649	0.431	0.920	2.817
		FIRI(SI)	**0.020**	**0.007**	**0.008**	**0.041**	**0.149**	**0.058**	**0.046**	**0.341**	**0.544**	**0.173**	**0.310**	**1.049**
Line	2-D	**FIRI**	0.038	0.014	0.010	0.068	0.120	0.039	0.047	0.204	0.263	0.086	0.040	0.440
		RILS [2]	0.014	0.004	0.007	0.028	0.041	0.012	0.016	0.073	0.085	0.028	**0.030**	0.142
		FIRI(SI)	**0.008**	**0.003**	**0.002**	**0.018**	**0.029**	**0.010**	**0.011**	**0.050**	**0.079**	**0.017**	0.045	**0.115**
	3-D	**FIRI**	0.163	0.037	0.090	0.239	0.678	0.208	0.272	1.110	2.977	0.930	1.709	4.651
		RILS [2]	0.061	0.020	0.027	0.114	0.417	0.134	0.148	0.721	2.069	0.532	1.099	3.214
		FIRI(SI)	**0.025**	**0.007**	**0.012**	**0.039**	**0.148**	**0.049**	**0.053**	**0.249**	**0.696**	**0.216**	**0.349**	**1.084**

in terms of manageability, to the extent that it cannot guarantee to contain the seed even when the seed is just a single point. While RILS can ensure to contain the line seed, it is not sufficient to guarantee that the generated convex hull contains the seed when the seed is represented as a convex hull. As for Galaxy, its heuristic cut method does not ensure that the seed is included either, when the seed is a line or a convex polytope. In contrast, although FIRI employs a similar approach to IRIS and RILS in computing halfspaces through ellipsoid inflation for polytope generation, the integration of our proposed RsI provides a distinct advantage: each halfspace computed by FIRI is guaranteed to contain the seed. This fundamental property ensures that the resulting polytope invariably encompasses the seed, thus maintaining manageability.

2) Efficiency and Quality: We conduct experiments to compare efficiency and quality in the aforementioned random environments. using the same settings of obstacles and boundary as Sec. VII-A1. We record the computation time for each algorithm with point seed. If IRIS generates a convex polytope during the iteration process that does not include the seed, we force IRIS to terminate prematurely and return the polytope from the previous iteration that includes the seed as the result. Since both RILS and FIRI guarantee manageability for line seeds, we document their respective time overhead when the seed input is a line as well. In addition, since the non-iterative method RILS essentially represents a single iteration of IRIS without MVIE computation, we also record the outcomes of a single iteration of FIRI, denoted as **FIRI(SI)**.

To provide an intuitive comparison of the size of convex polytopes generated by each algorithm, we take the proposed FIRI, which aims to maximize the convex hull volume, as the baseline. Specifically, we report the ratio of the volume of the convex hull obtained by each algorithm to the volume of the one obtained by FIRI, as demonstrated in Fig. 6. The efficiency results are presented in Tab. IV, where, for clarity, we distinguish between iterative and non-iterative methods using dashed lines in the case of point seed. In addition, since the number of input obstacles is the primary factor influencing the computational efficiency, we showcase the quantity of input obstacles in different density environments in Tab. II to provide readers with an insight into the computational efficiency of each algorithm. As the results illustrated in Fig. 6, both IRIS and FIRI iteratively compute larger convex polytopes by continuously inflating the MVIE, resulting in similar sizes. However, in IRIS, the SDP-based MVIE solving method consumes significant computation time [1], whereas FIRI achieves significant efficiency improvements in MVIE calculations as shown in Sec. VII-C, leading to remarkably higher efficiency compared to IRIS as shown in Tab. IV. Moreover, FIRI achieves a computational time that is within three times the time of non-iterative RILS without involving MVIE. The other three non-iterative methods yield smaller polytopes, but achieve better efficiency than IRIS in Tab. IV. Galaxy, however, is even less efficient than FIRI due to the fact that Quickhull [33] involved degrades in the scene to which Galaxy corresponds, which is particularly noticeable in the 3D case. As for FIRI(SI), similar to RILS, it directly updates

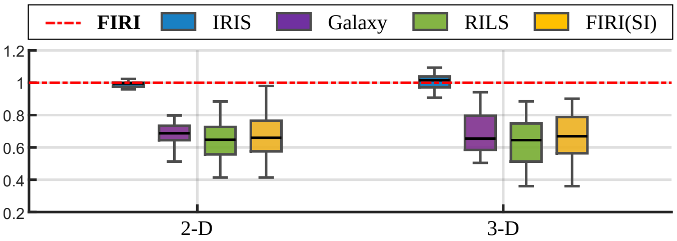


Fig. 6. Comparison of the sizes of the free convex polytopes generated by different methods. The dashed line with a value of 1 indicates the result obtained by FIRI, which we take as the baseline.

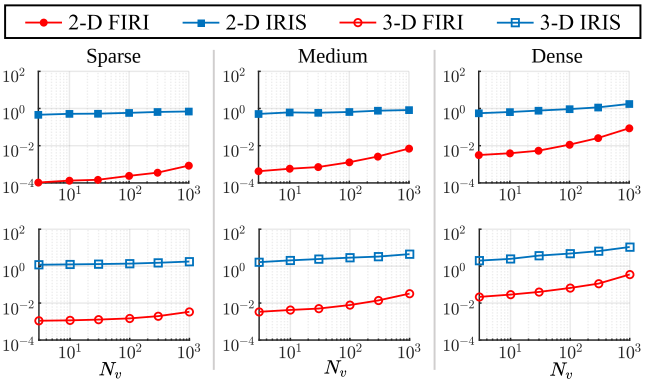
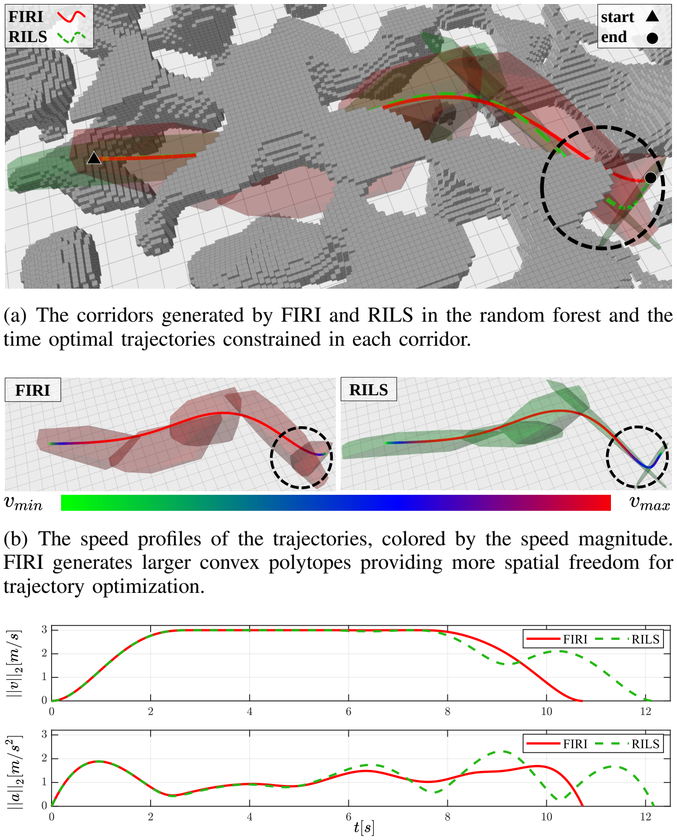


Fig. 7. The benchmark on computation time with the number of obstacle vertices for FIRI and IRIS in different environmental densities. The top represents the 2-D case and the bottom represents the 3-D case. FIRI has orders of magnitude advantage in computational efficiency over IRIS.

the convex hull and can generate polytope of comparable size to RILS, but with higher computational efficiency. Moreover, benefiting from the manageability brought about by the RsI, if the objective is to obtain a free convex polytope that includes the seed as fast as possible, without pursuing maximum volume, we believe that FIRI(SI) is a more suitable choice.

Apart from using point representation for obstacles as used above, employing convex polytopes is also a commonly used method for representing the environment. Although methods such as RILS can handle polytope obstacles through discretizing them into points, this approach may incur increased computational burden when discretizing large-scale simple convex polytope, and certain discretization methods (like gridding) may compromise environmental modeling accuracy. Hence, we consider the capability to directly process polytope obstacles to be valuable. Among above methods in Tab. I, only FIRI and IRIS support obstacle inputs in the form of convex hulls, thus we further compare these two algorithms. For each test we generate a certain number of randomly distributed obstacles in a closed space and select the center of the space as the seed input. This closed space serves as the boundary. We conduct benchmarks by setting the number N_v of the vertices of one obstacle to vary from 3 to 10^3. Compared to the case using point obstacles, the relative volumes of obstacles that are polytopes are larger, thus we vary the number of obstacles to vary from 10 to 10^3, which we denote as spare, medium and dense in the results shown in Fig. 7. For each the number of the vertices of each different number of obstacles, we perform 500 tests. The increasing number of obstacles, the number of halfspaces generated for convex hull construction tends to stabilize, thus the computational efficiency advantage provided by the porposed MVIE solution gradually diminishes. However, benefiting from the efficient SDMN we propose for halfspace computation, FIRI can still maintain a significant advantage over IRIS, even in scenarios with dense obstacles, with a reduction of over 95% in computational requirements.

3) Case Study of Quality: To demonstrate the significance of convex hull quality and its impact on trajectory planning, we conduct an experiment on trajectory planning based on different convex hull generation methods in the random environment as shown in Fig. 8(a). Using RRT* [61], we generate a collision-free path between start and end points. This path can be viewed as a set of connected line segments, based on which we generate convex hulls subsequently. As indicated in Tab. III, only FIRI and RILS exhibit manageability over line seed. Thus we build corridors for comparison based on these two methods, which correspond to iterative and non-iterative strategies, regarding the pursuit of maximum volume, respectively. The process of generating a safe flight corridor is as follows: We sequentially traverse each line segment on the path. If the line segment is already included in the previously generated polytopes, we move on to the next segment. Otherwise, we use this line as a seed to generate a new convex polytope. Due to manageability, the generated set of convex polytopes must have an intersection between two neighboring pairs, forming a corridor. Based on the generated corridor, we adopt GPOPS-II [62] to obtain the optimal trajectory constrained within the corridor. This collocation-based method transcribes the trajectory optimization into a constrained Nonlinear Programming using the Gauss pseudospectral method, which is then solved by the well-established NLP solver SNOPT [63]. In GPOPS-II, each trajectory phase is confined within one polytope, and we set the feasibility constraints for velocity and acceleration as $3m/s$ and $6m/s^2$, and time weight as 20. As depicted in Fig. 8(b), RILS generates narrow polytopes in the area marked by black dashed circles, resulting in limited maneuvering space for trajectory optimization. Consequently, the trajectory constrained in the corridor generated



(a) The corridors generated by FIRI and RILS in the random forest and the time optimal trajectories constrained in each corridor.

(b) The speed profiles of the trajectories, colored by the speed magnitude. FIRI generates larger convex polytopes providing more spatial freedom for trajectory optimization.

(c) The velocity and acceleration magnitude for the trajectories constrained in the corridors generated by FIRI and RILS respectively. Greater spatial freedom provided by FIRI leads to more aggressive trajectory performance.

Fig. 8. The comparison between the corridor established based on FIRI and RILS in a complex environment and the optimal trajectories constrained within the generated corridor.

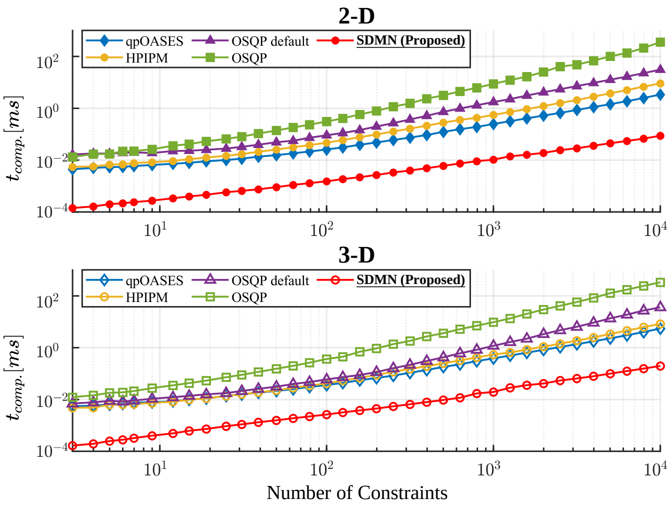


Fig. 9. Computation time $t_{comp.}$ of different methods for solving strictly convex small dimensional QP under different constraint numbers. The proposed SDMN outperforms other methods by orders of magnitudes.

TABLE V
COMPARISON OF PRECISIONS ψ_N BETWEEN DIFFERENT METHODS FOR SOLVING SMALL DIMENSIONAL MINIMUM-NORM

Scenario	Precisions ψ_N				
	qpOASES [14]	OSQP [15] default	OSQP [15]	HPIPM [16]	**SDMN (Proposed)**
2-D	1.28e-15	5.05e-2	6.61e-11	7.79e-06	**2.78e-17**
3-D	2.91e-15	2.43e-2	2.95e-11	4.41e-06	**3.55e-17**

by RILS exhibits a conservative behavior in the marked area . In contrast, FIRI, due to its pursuit of maximizing convex hulls, is capable of generating larger corridors, providing greater spatial flexibility in trajectory planning.

B. Comparison of Solving Strictly Convex Small Dimensional Minimum-Norm with Massive Constraints

For minimum-norm (16), we compare SDMN with several cutting-edge general-purpose QP solvers: the parametric active-set algorithm qpOASES [14], the operator-splitting based first-order approximation algorithm OSQP [15] and the interior-point method-based quadratic programming approximation algorithm HPIPM [16]. We use the open source code [2][3][4] of these solvers to obtain the solution of (16) respectively. For qpOASES and HPIPM, we use the default parameters. However, we find that the default parameters of OSQP are insufficient when the constraint size becomes large. Therefore, we adjust the relative convergence tolerance to 10^{-12} and the maximal number of iterations to 10^6, denoted as OSQP, while the implementation with the default parameters are denoted as OSQP default. As for the proposed SDMN, we do not require any additional parameter settings.

We compare the solving time of each method, as well as their average precision which is defined as

$$\psi_N = \mathcal{L}_\psi(Ey^* - f), \tag{61}$$

where y^* is the corresponding solution of each method, E and f are the parameters used to define the constraints of the minimum-norm (16), and $\mathcal{L}_\psi(\cdot)$ denotes a function that takes the absolute value of the largest element of the input vector. The precision ψ_N indicates the degree of proximity of the solution of each method to the most active constraints of the input minimum-norm. The performance is reported in Fig. 9 and Tab. V. The results demonstrate that SDMN exhibits remarkably high computational efficiency, surpassing other methods by orders of magnitude. Additionally, SDMN provides analytical solutions, achieving high accuracy without the need for additional parameters.

C. Comparison of Solving MVIE

For MVIE (11), we propose two algorithms, namely the SOCP-reformulation algorithm and the randomized algorithm specialized for 2-D case. The randomized algorithm yields an analytical solution, here we abbreviate it as **RAN**. To distinguish from above methods, we denote the proposed SOCP-reformulation algorithm which is solved by affine scaling method as **CAS**. We compare them with three methods: 1) The optimization method based on SDP formulation of MVIE in IRIS [1]. 2) The example [5] of the cutting-edge solver Mosek [23] to computes the Lowner-John inner ellipsoidal approximations of a polytope. 3) A strategy of solving the SOCP form (35) of the MVIE by Mosek. As the example in Mosek formulate MVIE into a mixed conic quadratic and semidefinite problem, we denote it as **Mosek SDP** and denote another strategy using Mosek as **Mosek SOCP**. We use the default parameters for IRIS and Mosek.

We compare the computation time and average precision of each method to calculate the maximum ellipsoid in closed convex polytopes consisting of different numbers of halfspaces. The precision is defined as

$$\psi_\mathcal{E} = \mathcal{L}_\psi\left([[A_\mathcal{P} A_\mathcal{E}{}^* D_\mathcal{E}{}^*]^2 \mathbf{1}]^{\frac{1}{2}} + A_\mathcal{P} b_\mathcal{E}{}^* - b_\mathcal{P}\right), \tag{62}$$

where $A_\mathcal{E}{}^*$, $D_\mathcal{E}{}^*$ and $b_\mathcal{E}{}^*$ are the coefficients of the maximum inscribed ellipsoid solved by each method, $A_\mathcal{P}$ and $b_\mathcal{P}$ are the parameters that define the halfspaces of the input polytope $\mathcal{P}$ defined in (10), and $\mathcal{L}_\psi(\cdot)$ is the function defined in (61). This precision $\psi_\mathcal{E}$ represents the proximity of the ellipsoids obtained by different methods to the most active halfspace constraint, which are constructed in the form of second-order cone constraint as (27b). The results are summarized in Fig. 10 and Tab. VI. By comparing the performance of Mosek SDP and Mosek SOCP, we can get the conclusion that transforming the commonly used SDP formulation to the SOCP formulation, which is presented in Sec. V-A, leads to a significant improvement in computational efficiency without sacrificing precision. Additionally, in the low-dimension massive-constraint case faced in this paper, the use of the affine scaling method avoids the requirement of solving a large-scale system of linear equations at each iteration, compared to the primal-dual interior point method used in Mosek. Therefore

[2] https://github.com/coin-or/qpOASES
[3] https://github.com/osqp/osqp
[4] https://github.com/giaf/hpipm

[5] https://docs.mosek.com/latest/cxxfusion/examples-list.html#doc-example-file-lownerjohn-ellipsoid-cc

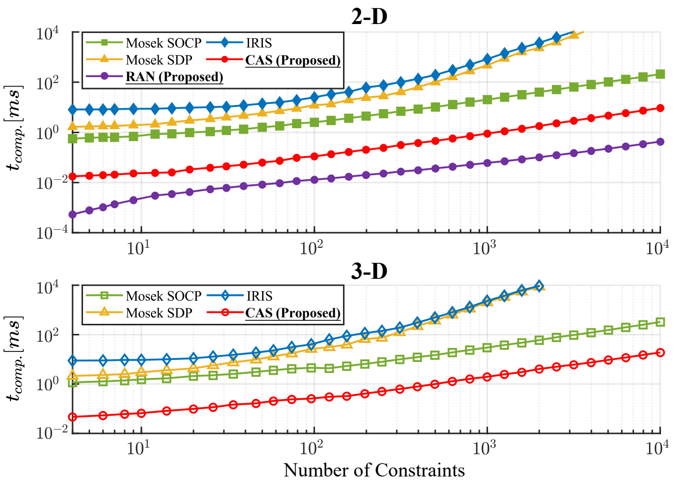


Fig. 10. Computation time $t_{comp.}$ of different methods for solving MVIE under different constraint numbers. The proposed CAS outperforms other methods by orders of magnitudes. Additionally, the proposed analytical method RAN further enhances efficiency for the 2D case.

TABLE VI
COMPARISON OF PRECISION $\psi_{\mathcal{E}}$ BETWEEN DIFFERENT METHODS FOR SOLVING MVIE

Scenario	Precisions $\psi_{\mathcal{E}}$				
	IRIS [1]	Mosek [23] SDP	Mosek [23] SOCP	**CAS (Proposed)**	**RAN (Proposed)**
2-D	8.56e-8	6.47e-9	4.87e-12	1.59e-8	**4.41e-16**
3-D	1.54e-8	1.56e-8	4.05e-12	2.04e-8	/

as the results demonstrate, CAS is capable of solving MVIE in less time while maintaining a comparable precision. Moreover, for the 2-D case, our proposed linear-time complexity algorithm RAN further enhances the computational efficiency by orders of magnitude and compute analytical results.

VIII. Real-world Application

To validate the performance of FIRI in practical applications, we designed two real-world applications: a global trajectory planning for a vehicle robot with nonholonomic constraints in 2-D environment and a local planning for a quadrotor in 3-D environment.

A. Dense Corridor for 2-D Whole-body Planning

The trajectory planning for large-volume vehicle in narrow environments requests the entire vehicle to be constrained within the safe region, which puts a demand on manageability.

We adopt a differential driven AgileX SCOUT MINI [6] as the physical platform. The platform is equipped with an Intel NUC [7] with an Intel Core i7-1165G7 as onboard processor and a LiDAR sensor [8] integrated with an IMU for perception. We use FAST-LIO2 [64] for both pre-mapping the environment and real-time localization during execution of the planned trajectory. That is, the obstacles are represented by points.

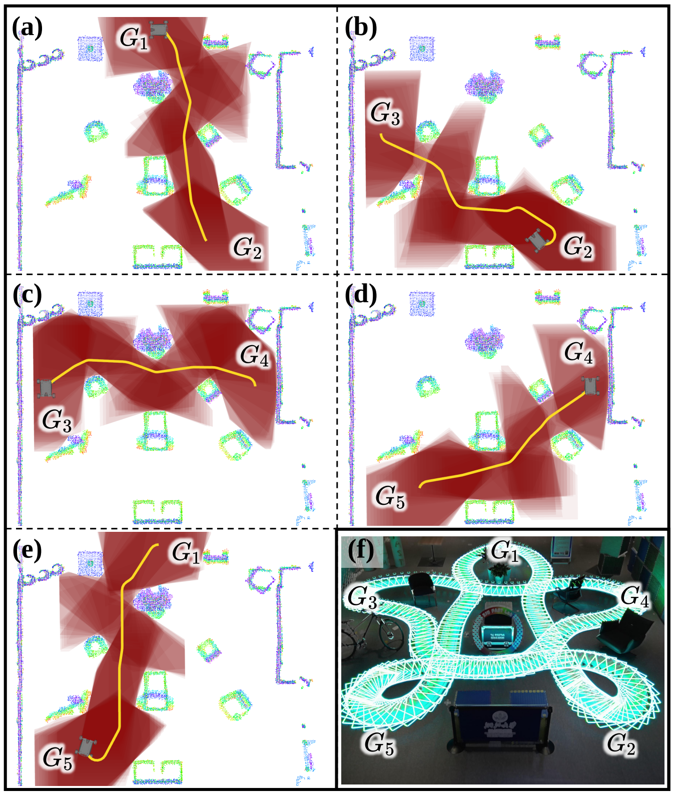


Fig. 11. **(a)-(e)**: Illustration of the corridors generated by FIRI during the traversal of the robot from G_1 through G_2-G_5 and finally back to G_1 in a cluttered environment. **(f)**: Snapshots of the robot's navigation result.

Building on our previous work [11], we use hybrid A* as the front-end to obtain an initial feasible trajectory, and then finely sample on the trajectory. The robot shape corresponding to each sampled state is used as a seed to generate a convex polytope by FIRI, forming a dense safety corridor, as shown in Fig. 12. In the back-end, we utilize polynomial representation for the trajectory and constrain that the vehicle shape corresponding to each sampled constraint state on the trajectory is fully contained within the corresponding convex polytope of the safety corridor. Considering localization errors and control errors, for safety purposes, we inflate the robot's shape during planning to generate a safer corridor and trajectory.

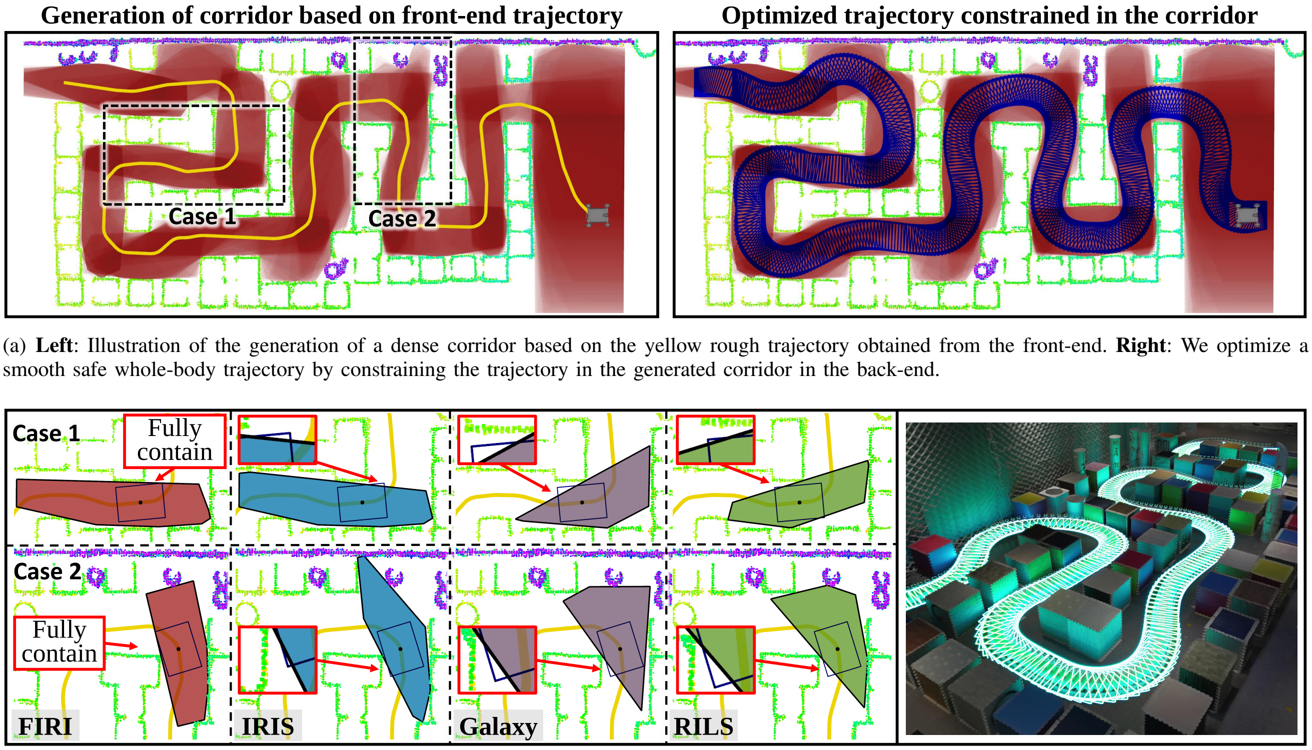
1) Cluttered Environment: The results are shown in Fig 11. We require the robot to navigate through a cluttered environment filled with irregular obstacles, starting from G_1 and sequentially reaching G_2-G_5, and finally returning to G_1. Since the focus of this paper is on the generation of free convex polytope, only the front-end trajectory that guides the generation of convex polytope and the generated convex polytopes are demonstrated. We refer the readers to the accompanying video for the detailed navigation process. In this experiment, as shown in Fig 11, we generate 5 dense corridors, with an average of 91.8 convex polytopes in each corridor. For each convex polytope, the average number of input obstacle points was 715.3, with an average processing time of 0.131 ms. The average time taken to generate a corridor is 12.02 ms.

2) Narrow Maze: We task the robot with traversing through a narrow maze. The result, as depicted in Fig. 12(a), demonstrates that FIRI, based on the yellow front-end trajectory, generates high-quality convex polytopes, providing sufficient

[6] https://global.agilex.ai/products/scout-mini

[7] https://www.intel.com/content/dam/support/us/en/documents/intel-nuc/NUC11PH_TechProdSpec.pdf

[8] https://ouster.com/products/hardware/os1-lidar-sensor


Generation of corridor based on front-end trajectory
Optimized trajectory constrained in the corridor
Case 1
Case 2
Case 1
Fully contain
Case 2
Fully contain
FIRI
IRIS
Galaxy
RILS

(a) **Left**: Illustration of the generation of a dense corridor based on the yellow rough trajectory obtained from the front-end. **Right**: We optimize a smooth safe whole-body trajectory by constraining the trajectory in the generated corridor in the back-end.

(b) **Left**: For the two specific cases in (a), we demonstrate the convex polytopes generated by the different methods. The results show that only FIRI is able to satisfy the whole-body planning requirement of generating polytopes that fully contain the robot. **Right**: Snapshots of the robot's navigation result.

Fig. 12. A real-world application of a differential driven robot tasked with traversing a maze, utilizing FIRI to abstract the feasible region.

spatial degrees of freedom for trajectory optimization in the back-end. Additionally, as shown in Fig. 12(b), to intuitively demonstrate the manageability of our experiment, we select two particularly narrow cases within the maze environment, illustrating the results of the convex polytopes generated by various methods [1], [2], [9] compared in Sec. VII-A. It can be observed that only FIRI is capable of fully enclosing the robot. A more detailed demonstration of the generation from each seed and its corresponding convex polytope along the entire trajectory is presented in the accompanying video. In this experiment, we generate a dense corridor consisting of 550 convex polytopes. For each convex polytope, the average number of input obstacle points is 1307.1, with an average processing time of $0.249\ ms$. The total time taken for generating the entire corridor is $136.74\ ms$.

B. Sparse Corridor for 3-D Local Replanning

Our platform is a customized quadrotor equipped with an NVIDIA Orin NX[9] as the onboard processor and a Livox MID-360[10] LiDAR for perception. For the quadrotor, due to its small size, we model it as a point mass, similar to many other works [2], [8]. We employ FAST-LIO2 [64] for online state estimation and utilize occupancy grid map to filter sensor noise, which is easy to inflate for safety margins. For trajectory planning, the path generation and corridor generation are the same as Sec. VII-A3. Then building upon our previous work [3], we utilize piecewise polynomials to represent the trajectory. Each piece of the trajectory is constrained within the corresponding polytope to ensure safety.

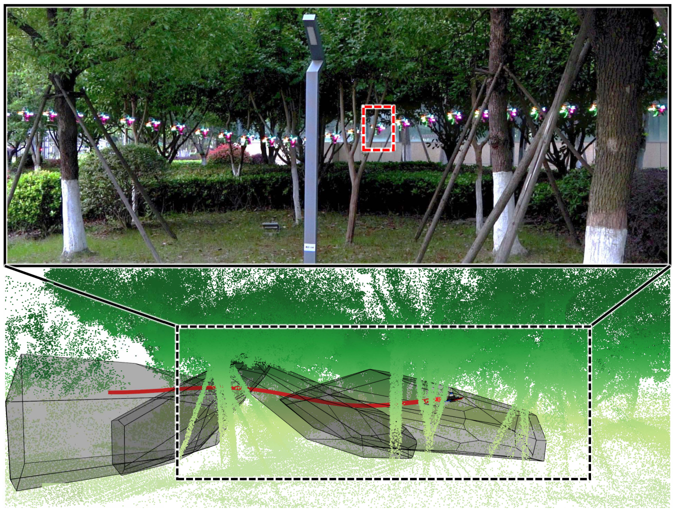


Fig. 13. Illustration of the application of FIRI in a local trajectory planning framework for a quadrotor to navigate through a cluttered forest. **Top**: A snapshots of the quadrotor traveling through the forest. **Bottom**: The visualization of quadrotor replanning for the moments corresponding to the red boxes. The black transparent polytopes are generated by FIRI.

As shown in Fig. 13, the experiment is conducted in a dense forest, and the quadrotor is required to navigate through a series of waypoints. Since the environment is unknown, we make the quadrotor perform high-frequency replan (20Hz) according to the real-time perception. The maximum velocity of the UAV during flight is up to 4.5 m/s. Each replan involves 3-7 convex polytopes, the average generation time for

[9] https://www.nvidia.com/en-us/autonomous-machines/

[10] https://www.livoxtech.com/mid-360

each polytope is 2.76 ms, and the average number of input obstacle points for FIRI is 8219.6. We refer readers to the accompanying video for more information.

IX. Discussion and Conclusion

Our algorithm FIRI aims to generate the largest possible convex polytopes. However, when it comes to practical applications such as trajectory planning, we cannot guarantee that the inflation direction of polytope is always favorable for trajectory optimization. For example, when the robot has a high velocity, if the inflation direction is perpendicular to the velocity direction, it may lead to generated polytope that are not conducive to trajectory optimization. Our future research direction will focus on generating convex polytopes that are application-friendly. Additionally, the number of faces of the generated polytope significantly affects how fast the motion planning problem can be solved. As discussed in Sec. III.B1, we currently employ a widely used greedy method to reduce this number, and in the future, we will explicitly incorporate the face number as a further criterion of the region quality, while balancing it with the volume of the polytope.

In conclusion, we propose a novel obstacle-free convex polytope generation algorithm called FIRI, which achieves high quality, efficiency, and manageability simultaneously. To achieve efficiency, we design targeted methods for the two optimization problems involved in FIRI. Specifically, for 2-D MVIE, we develop a linear-time complexity method, which is proposed for the first time. We perform extensive benchmarks against several convex polytope generation algorithms to confirm the superior performance of FIRI. The comparisons with general-purpose solvers demonstrate the speedup over orders of magnitude of our targeted methods. Two typical applications showcase the practicality of FIRI.

CGAL [65] lacks a dedicated solver for 2-D MVIE. In the future, we will continue our research on 2-D MVIE and work towards implementing a version of rational predicates.

X. Acknowledgment

The linear-time complexity algorithm for 2-D MVIE is initially designed by Zhepei Wang and finalized by Qianhao Wang. The authors would like to thank Yuan Zhou and Mengze Tian for their help in the hardware of the 2-D application.

Appendix

We provide a proof of monotonicity and locality of the proposed value function (44) for 2-D MVIE in Sec. VI-B1.

A. Proof of Monotonicity

Proof. $\mathcal{G}$ is unclosed: For the polar cone of a set, when we add elements to the set, the angle of the cone either remains the same or decreases. Therefore, the first part of the function $\bar{w}_u(\mathcal{G})$ is monotonically increasing. Additionally, the second part of the function is clearly monotonically increasing. Since we combine these two parts using the lexicographical ordering $<$, the function $\bar{w}_u(\mathcal{G})$ satisfy monotonicity.

$\mathcal{G}$ is closed: Since the second element of the function $\bar{w}_c(\mathcal{G})$ is constant, we only need to prove the monotonicity of the first part. We denot a constraint $h \notin \mathcal{G}$. The MVIE of $\mathcal{G} \cup \{h\}$ certainly satisfies the original constraints of $\mathcal{G}$, thus its area must be less than or equal to the area of the MVIE of $\mathcal{G}$.

As stated in Sec. VI-B1, the first element of $\bar{w}_c$ is always greater than 0, and the first element of $\bar{w}_u$ is always less than or equal to 0, thus we have $\bar{w}_c > \bar{w}_u$. □

B. Proof of Locality

Proof. For finite sets $\mathcal{G}$ and $\mathcal{F}$ and a constraint h such that $\mathcal{G} \subseteq \mathcal{F}$, $h \notin \mathcal{F}$. When $-\infty < w(\mathcal{G}) = w(\mathcal{F})$, these two sets must either both be closed or both be unclosed.

$\mathcal{G}$ is unclosed: Based on the above condition, the polar cones of the new sets formed by the unit normal vectors of the respective elements of $\mathcal{G}$ and $\mathcal{F}$ are the same. Thus the new element h makes any change in this cone or even its corresponding normal vector falling in this cone leads to a closed polytope, which is the same for both $\mathcal{G}$ and $\mathcal{F}$.

$\mathcal{G}$ is closed: Due to the uniqueness of MVIE, if $\mathcal{G} \subseteq \mathcal{F}$ and $w(\mathcal{G}) = w(\mathcal{F})$, then their MVIE corresponds to the same ellipse. Thus if h is violated by $\mathcal{F}$, it will be violated by $\mathcal{G}$. □

References

[1] R. Deits and R. Tedrake, “Computing large convex regions of obstacle-free space through semidefinite programming,” in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 109–124.

[2] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, “Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments,” *IEEE Robotics and Automation Letters*, pp. 1688–1695, 2017.

[3] Z. Wang, X. Zhou, C. Xu, and F. Gao, “Geometrically constrained trajectory optimization for multicopters,” *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 2022.

[4] J. Guo, Z. Xun, S. Geng, Y. Lin, C. Xu, and F. Gao, “Dynamic free-space roadmap for safe quadrotor motion planning,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 523–10 528.

[5] P. Werner, A. Amice, T. Marcucci, D. Rus, and R. Tedrake, “Approximating robot configuration spaces with few convex sets using clique covers of visibility graphs,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 10 359–10 365.

[6] R. Deits and R. Tedrake, “Efficient mixed-integer planning for UAVs in cluttered environments,” in *IEEE International Conference on Robotics and Automation*, Seattle, USA, 2015, pp. 42–49.

[7] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, “Motion planning around obstacles with convex optimization,” *Science robotics*, vol. 8, no. 84, p. eadf7843, 2023.

[8] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, “Teach-Repeat-Replan: A complete and robust system for aggressive flight in complex environments,” *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1526–1545, 2020.

[9] X. Zhong, Y. Wu, D. Wang, Q. Wang, C. Xu, and F. Gao, “Generating large convex polytopes directly on point clouds,” *arXiv preprint arXiv:2010.08744*, 2020.

[10] J. Ji, N. Pan, C. Xu, and F. Gao, “Elastic tracker: A spatio-temporal trajectory planner for flexible aerial tracking,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 47–53.

[11] Z. Han, Y. Wu, T. Li, L. Zhang, L. Pei, L. Xu, C. Li, C. Ma, C. Xu, S. Shen *et al.*, “An efficient spatial-temporal trajectory planner for autonomous vehicles in unstructured environments,” *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[12] S. Savin, “An algorithm for generating convex obstacle-free regions based on stereographic projection,” in *2017 International Siberian Conference on Control and Communications (SIBCON)*. IEEE, 2017, pp. 1–6.

[13] R. Seidel, “Small-dimensional linear programming and convex hulls made easy,” *Discrete & Computational Geometry*, vol. 6, no. 3, pp. 423–434, 1991.

[14] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, “qpoases: A parametric active-set algorithm for quadratic programming,” *Mathematical Programming Computation*, vol. 6, pp. 327–363, 2014.

[15] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, “Osqp: An operator splitting solver for quadratic programs,” *Mathematical Programming Computation*, pp. 1–36, 2020.

[16] G. Frison and M. Diehl, “Hpipm: a high-performance quadratic programming framework for model predictive control,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[17] L. G. Khachiyan and M. J. Todd, “On the complexity of approximating the maximal inscribed ellipsoid for a polytope,” *Mathematical Programming*, vol. 61, no. 1, pp. 137–159, 1993.

[18] L. Lovász, *An algorithmic theory of numbers, graphs and convexity*. SIAM, 1986.

[19] H. W. Lenstra Jr, “Integer programming with a fixed number of variables,” *Mathematics of operations research*, vol. 8, no. 4, pp. 538–548, 1983.

[20] A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. SIAM, 2001.

[21] Y. Zhang and L. Gao, “On numerical solution of the maximum volume ellipsoid problem,” *SIAM Journal on Optimization*, vol. 14, no. 1, pp. 53–76, 2003.

[22] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, 1994.

[23] MOSEK Aps, “MOSEK Optimizer API for C,” 2020. [Online]. Available: https://www.mosek.com

[24] E. Welzl, “Smallest enclosing disks (balls and ellipsoids),” in *New Results and New Trends in Computer Science: Graz, Austria, June 20–21, 1991 Proceedings*. Springer, 2005, pp. 359–370.

[25] M. Sharir and E. Welzl, “A combinatorial bound for linear programming and related problems,” in *STACS 92: 9th Annual Symposium on Theoretical Aspects of Computer Science Cachan, France, February 13–15, 1992 Proceedings 9*. Springer, 1992, pp. 567–579.

[26] J. Matoušek, M. Sharir, and E. Welzl, “A subexponential bound for linear programming,” in *Proceedings of the eighth annual symposium on Computational geometry*, 1992, pp. 1–8.

[27] M. Montanari, N. Petrinic, and E. Barbieri, “Improving the gjk algorithm for faster and more reliable distance queries between convex objects,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 3, pp. 1–17, 2017.

[28] Y. Wu, I. Spasojevic, P. Chaudhari, and V. Kumar, “Optimal convex cover as collision-free space approximation for trajectory generation,” *arXiv preprint arXiv:2406.09631*, 2024.

[29] H. Dai, A. Amice, P. Werner, A. Zhang, and R. Tedrake, “Certified polyhedral decompositions of collision-free configuration space,” *The International Journal of Robotics Research*, p. 02783649231201437, 2023.

[30] M. Petersen and R. Tedrake, “Growing convex collision-free regions in configuration space using nonlinear programming,” *arXiv preprint arXiv:2303.14737*, 2023.

[31] P. Werner, T. Cohn, R. H. Jiang, T. Seyde, M. Simchowitz, R. Tedrake, and D. Rus, “Faster algorithms for growing collision-free convex polytopes in robot configuration space,” *arXiv preprint arXiv:2410.12649*, 2024.

[32] S. Katz, A. Tal, and R. Basri, “Direct visibility of point sets,” in *ACM SIGGRAPH 2007 papers*, 2007, pp. 24–es.

[33] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, “The quickhull algorithm for convex hulls,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996.

[34] F. John, “Extremum problems with inequalities as subsidiary conditions,” *Traces and emergence of nonlinear programming*, pp. 197–215, 2014.

[35] S. P. Tarasov, “The method of inscribed ellipsoids,” in *Soviet Mathematics-Doklady*, vol. 37, no. 1, 1988, pp. 226–230.

[36] L. G. Khachiyan and M. J. Todd, “On the complexity of approximating the maximal inscribed ellipsoid for a polytope,” Cornell University Operations Research and Industrial Engineering, Tech. Rep., 1990.

[37] K. M. Anstreicher, “Improved complexity for maximum volume inscribed ellipsoids,” *SIAM Journal on Optimization*, vol. 13, no. 2, pp. 309–320, 2002.

[38] A. Nemirovski, “On self-concordant convex–concave functions,” *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 303–384, 1999.

[39] C.-H. Lin, R. Wu, W.-K. Ma, C.-Y. Chi, and Y. Wang, “Maximum volume inscribed ellipsoid: A new simplex-structured matrix factorization framework via facet enumeration and convex optimization,” *SIAM Journal on Imaging Sciences*, vol. 11, no. 2, pp. 1651–1679, 2018.

[40] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[41] C. D. Toth, J. O’Rourke, and J. E. Goodman, *Handbook of Discrete and Computational Geometry*. CRC Press, 2017.

[42] J.-S. Chang and C.-K. Yap, “A polynomial solution for the potato-peeling problem,” *Discrete & Computational Geometry*, vol. 1, no. 2, pp. 155–182, 1986.

[43] P. Song, B. Deng, Z. Wang, Z. Dong, W. Li, C.-W. Fu, and L. Liu, “Cofifab: coarse-to-fine fabrication of large 3d objects,” *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–11, 2016.

[44] M. E. Dyer and A. M. Frieze, “On the complexity of computing the volume of a polyhedron,” *SIAM Journal on Computing*, vol. 17, no. 5, pp. 967–974, 1988.

[45] M. Agarwal, J. Clifford, and M. Lachance, “Duality and inscribed ellipses,” *Computational Methods and Function Theory*, vol. 15, pp. 635–644, 2015.

[46] A. S. Householder, “Unitary triangularization of a nonsymmetric matrix,” *Journal of the ACM (JACM)*, vol. 5, no. 4, pp. 339–342, 1958.

[47] G. H. Golub and F. V. Loan, *Matrix Computations*. The Johns Hopkins University Press, 2013.

[48] J. Lagarias and R. Vanderbei, “Ii dikin’s convergence result for the affine scaling algorithm,” *Contemp. Math*, vol. 114, p. 109, 1990.

[49] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[50] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 2012.

[51] T. Tsuchiya and R. D. C. Monteiro, “Superlinear convergence of the affine scaling algorithm,” *Mathematical Programming*, vol. 75, no. 1, pp. 77–110, 1996.

[52] D. Minda and S. Phelps, “Triangles, ellipses, and cubic polynomials,” *The American Mathematical Monthly*, vol. 115, no. 8, pp. 679–689, 2008.

[53] A. Horwitz, “Ellipses of maximal area and of minimal eccentricity inscribed in a convex quadrilateral,” *Australian Journal of Mathematical Analysis and Applications*, vol. 2, no. 1, p. 12, 2005.

[54] A. C. Jones, *An introduction to algebraical geometry*. Clarendon Press, 1912.

[55] J. Richter-Gebert and J. Richter-Gebert, “Conics and their duals,” *Perspectives on projective geometry: A guided tour through real and complex geometry*, pp. 145–166, 2011.

[56] F. Klein, C. A. T. Noble, and E. R. T. Hedrick, *Elementary Mathematics from an Advanced Standpoint-Geometry: Transl. from the Third German Ed. by ER Hedrick and CA Noble*. Dover, 1939.

[57] M. J. D. Hayes, Z. A. Copeland, P. J. Zsombor-Murray, and A. Gfrerrer, “Largest area ellipse inscribing an arbitrary convex quadrangle,” in *Advances in Mechanism and Machine Science: Proceedings of the 15th IFToMM World Congress on Mechanism and Machine Science 15*. Springer, 2019, pp. 239–248.

[58] J. Lin, C. Yuan, Y. Cai, H. Li, Y. Ren, Y. Zou, X. Hong, and F. Zhang, “Immesh: An immediate lidar localization and meshing framework,” *IEEE Transactions on Robotics*, 2023.

[59] Q. Wang, Z. Wang, L. Pei, C. Xu, and F. Gao, “A linear and exact algorithm for whole-body collision evaluation via scale optimization,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3621–3627.

[60] J. C. Hart, “Perlin noise pixel shaders,” in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, 2001, pp. 87–94.

[61] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[62] M. A. Patterson and A. V. Rao, “Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 1, pp. 1–37, 2014.

[63] P. E. Gill, W. Murray, and M. A. Saunders, “Snopt: An sqp algorithm for large-scale constrained optimization,” *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[64] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, “Fast-lio2: Fast direct lidar-inertial odometry,” *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[65] A. Fabri and S. Pion, “Cgal: The computational geometry algorithms library,” in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2009, pp. 538–539.