A Novel Approach for Ellipsoidal Outer-Approximation of the Intersection Region of Ellipses in the Plane

Noname manuscript No.
(will be inserted by the editor)

A Novel Approach for Ellipsoidal Outer-Approximation of the Intersection Region of Ellipses in the Plane

Siamak Yousefi · Xiao-Wen Chang · Henk Wymeersch · Benoit Champagne · Godfried Toussaint

Received: date / Accepted: date

Abstract In this paper, a novel technique for tight outer-approximation of the intersection region of a finite number of ellipses in 2-dimensional (2D) space is proposed. First, the vertices of a tight polygon that contains the convex intersection of the ellipses are found in an efficient manner. To do so, the intersection points of the ellipses that fall on the boundary of the intersection region are determined, and a set of points is generated on the elliptic arcs connecting every two neighbouring intersection points. By finding the tangent lines to the ellipses at the extended set of points, a set of half-planes is obtained, whose intersection forms a polygon. To find the polygon more efficiently, the points are given an order and the intersection of the half-planes corresponding to every two neighbouring points is calculated. If the

Funding for this work was provided in parts by research grants from the Natural Sciences and Engineering Research Council of Canada

S. Yousefi
Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada
E-mail: siamak.yousefi@mail.mcgill.ca

X.-W. Chang
School of Computer Science, McGill University, Montreal, QC, Canada
E-mail: chang@cs.mcgill.ca

H. Wymeersch
Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden
E-mail: henkw@chalmers.se

B. Champagne
Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada
E-mail: benoit.champagne@mcgill.ca

G. Toussaint
Computer Science Program, New York University Abu Dhabi, United Arab Emirates
E-mail: gt42@nyu.edu

polygon is convex and bounded, these calculated points together with the initially obtained intersection points will form its vertices. If the polygon is non-convex or unbounded, we can detect this situation and then generate additional discrete points only on the elliptical arc segment causing the issue, and restart the algorithm to obtain a bounded and convex polygon. Finally, the smallest area ellipse that contains the vertices of the polygon is obtained by solving a convex optimization problem. Through numerical experiments, it is illustrated that the proposed technique returns a tighter outer-approximation of the intersection of multiple ellipses, compared to conventional techniques, with only slightly higher computational cost.

Keywords Computational geometry · convex optimization · ellipsoidal outer approximation · intersection of ellipses · intersection of half-planes · minimum volume enclosing ellipsoid

1 Introduction

In many areas of science and engineering, such as computational geometry, image processing, control systems, parameter estimation, and wireless communications, a complex convex set needs to be represented by a simpler geometric shape containing it, e.g., see in [1–9] and the references therein. Among possible different shapes, ellipsoids are often considered since they can be easily described in terms of vectors and matrices. Ellipsoids often provide tight outer-approximations of the underlying convex sets, and are invariant under affine transformations [8]. Therefore, ellipsoidal calculus has gained significant attention in many different fields due to its importance and usefulness. For instance, in control theory, ellipsoidal bounds are used to describe the uncertainty of sets associated with state space models [10]. In sensor network localization, ellipsoids are employed to provide constraints on the unknown positions of sensors [11], [12].

According to [9, p.44], verifying that an ellipsoid covers the intersection of a given number of ellipsoids is NP-complete, so the problem cannot be expressed as a linear matrix inequality (LMI). As a result, finding the minimum volume ellipsoid which contains the intersection of multiple ellipsoids may not be recast as a convex optimization problem. To the authors' knowledge, there exists no practically efficient computer code to find the optimal solution to this problem, and all the proposed techniques tested offer sub-optimal solutions. However, it is possible that the framework of linear-programming type problems might be suitable to solve this problem optimally in theory [13]. Earlier work on this topic focused on the special case of two ellipsoids due to its simplicity. For instance in [14], one of the ellipsoids is approximated by a half-space, and then the problem of finding the tightest ellipsoid containing the intersection of a half-space and an ellipsoid can be solved optimally. It is mentioned in [15] that the optimal ellipsoid can be expressed as a linear convex combination of the two ellipsoids. The optimization problem for searching within the convex combinations of two ellipsoids is investigated in [10] and

further analysis and theoretical results are derived in [16]. For a larger number of ellipsoids, standard convex optimization techniques can be used in order to efficiently find an outer-approximation, as summarized in [9]. For instance, one well known technique is to first obtain the largest ellipsoid enclosed in the intersection of the given ellipsoids, which can be done optimally by solving a convex optimization problem [17]. Then, as shown by John and Lowner (see [18] and the references therein), upon scaling the calculated ellipsoid by the dimension of the space, the resulting ellipsoid is guaranteed to contain the intersection region. These techniques are approximate and suboptimal, although bounding ellipses can be obtained in polynomial time using these convex optimization techniques. We will show this fact through simulations.

It is clear that the problem is quite challenging for arbitrary dimensions, but for 2-dimensional (2-D) space, low-cost geometrical techniques can be developed. Recently, in the context of sensor network localization under non-line-of-sight (NLOS) propagation, the authors in [19] proposed a 2-stage method for tight outer-approximation of the intersection of multiple ellipses in 2-D space, and then plugged-in their technique to the distributed bounding algorithm developed in [11]. In this method, first a polygon is obtained by generating discrete points on the boundary of each ellipse, rejecting those that fall outside the feasible region, and then intersecting the half planes tangent to the ellipses at those remaining points. Subsequently, the tightest ellipse that contains the vertices of the resulting polygon is obtained by solving a convex optimization problem [20]. This method can generally outperform existing approaches in the literature (see [9]) in 2-D, if enough discrete points are generated on each ellipse; however, it is always possible that the polygon becomes unbounded or non-convex. Although this problem can be avoided by generating a large number of discrete points on each ellipse, this will render the method inefficient when the number of ellipses is large.

In this work, we extend the work of [19] for outer-approximation of the intersection of several ellipses in 2-D space in the following ways, such that a tight bounded and convex polygon can be formed, and the drawbacks mentioned above can be avoided:

- For every elliptic arc forming the boundary of the intersection of ellipses, we find the two end points, which are the intersection points of two or more ellipses, and generate a desired number of points on that arc. Compared to [19], in this way, we avoid generating unnecessary discrete points, and thus improve the efficiency.
- The tangent lines corresponding to every two neighbouring points intersect, leading to a new set of points. These points along with the intersection points of the ellipses obtained earlier are used as the vertices of the possibly bounded and convex polygon.
- The boundedness and convexity of the polygon are verified and if it is unbounded or non-convex, the number of discrete points on the elliptic arc which causes the problem is increased. Therefore, the new method will

eventually find a bounded convex polygon containing the intersection of ellipses.

Finally, the tightest ellipse containing the vertices of the generated polygon is obtained by solving a convex optimization problem. Through numerical evaluations we observe that the proposed method performs better than other techniques in the literature, such as the ones summarized in [9], albeit with slightly higher computational cost. Furthermore, the proposed method can yield a tighter polygon than the method considered in [19] with similar computational cost, and can avoid a non-convex or unbounded polygon.

The organization of the paper is as follows: In Section 2, background and definitions are given, the problem is stated, and a brief summary of existing techniques is provided. The proposed method is described in more detail in Section 3. The performance analysis of each method in different scenarios is evaluated numerically in Section 4. Finally, Section 5 concludes the paper.

2 Background and Problem Statement

2.1 Notation

Small and capital bold letters represent vectors and matrices, respectively. The vector 2-norm operation is denoted by $\|\cdot\|$, and the matrix transpose and inverse operations are denoted by $(\cdot)^T$ and $(\cdot)^{-1}$, respectively. The determinant of a matrix $\boldsymbol{A}$ is denoted by $\mathsf{det}(\boldsymbol{A})$. The symbol $\boldsymbol{I}$ denotes an identity matrix of appropriate dimension. The notation $\boldsymbol{A} \succ 0$ ($\boldsymbol{A} \prec 0$) means that $\boldsymbol{A}$ is a positive definite (negative definite) matrix and $\boldsymbol{A} \succeq 0$ ($\boldsymbol{A} \preceq 0$) means that the matrix is symmetric positive semi-definite (negative semi-definite) [21]. By $\boldsymbol{x} \in \mathbb{R}^M$ and $\boldsymbol{X} \in \mathbb{R}^{M \times N}$ we mean that the vector $\boldsymbol{x}$ and matrix $\boldsymbol{X}$ are of size M and $M \times N$, respectively.

2.2 Various Forms of Representing an Ellipsoid

An ellipsoid in $\mathbb{R}^\nu$ can be defined by different but equivalent forms:

- Image of the unit ball: An ellipsoid can be obtained by mapping a unit ball as
$$\xi = \Big\{ \boldsymbol{x} \in \mathbb{R}^\nu : \boldsymbol{x} = \boldsymbol{P}\boldsymbol{y} + \boldsymbol{x}_c, \|\boldsymbol{y}\| \leq 1, \boldsymbol{y} \in \mathbb{R}^\nu \Big\} \tag{1}$$
where without loss of generality we can assume $\boldsymbol{P} \in \mathbb{R}^{\nu \times \nu}$ is symmetric positive definite, see, e.g., [22]. The volume of the ellipsoid is $V_\nu \mathsf{det}(\boldsymbol{P})$, where V_ν is the volume of the unit ball in $\mathbb{R}^\nu$.
- Quadratic form I:
$$\xi = \Big\{ \boldsymbol{x} \in \mathbb{R}^\nu : \ \|\boldsymbol{B}\boldsymbol{x} + \boldsymbol{d}\| \leq 1 \Big\} \tag{2}$$

where $\boldsymbol{B} \in \mathbb{R}^{\nu \times \nu}$ and $\boldsymbol{d} \in \mathbb{R}^{\nu}$. The two sets in (1) and (2) are identical when $\boldsymbol{B} = \boldsymbol{P}^{-1}$ and $\boldsymbol{d} = -\boldsymbol{P}^{-1}\boldsymbol{x}_c$. Herein we also assume that $\boldsymbol{B}$ is symmetric positive definite. The volume of the ellipsoid is then $V_\nu \mathsf{det}(\boldsymbol{B}^{-1})$.

– Quadratic form II:

$$\xi = \left\{ \boldsymbol{x} \in \mathbb{R}^{\nu} : \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} + 2\boldsymbol{x}^T \boldsymbol{b} + c \leq 0 \right\} \tag{3}$$

where $\boldsymbol{A} \in \mathbb{R}^{\nu \times \nu}$ is symmetric positive definite, $\boldsymbol{b} \in \mathbb{R}^{\nu}$, and $c \in \mathbb{R}$. The sets in (2) and (3) are identical when $\boldsymbol{A} = \boldsymbol{B}^T \boldsymbol{B}$, $\boldsymbol{b} = \boldsymbol{B}^T \boldsymbol{d}$, $c = \boldsymbol{d}^T \boldsymbol{d} - 1$ (so $c = \boldsymbol{b}^T \boldsymbol{A}^{-1} \boldsymbol{b} - 1$). When $\boldsymbol{A}$, $\boldsymbol{b}$ and c satisfy $c = \boldsymbol{b}^T \boldsymbol{A}^{-1} \boldsymbol{b} - 1$, the volume of the ellipsoid is equal to $V_\nu \mathsf{det}(\boldsymbol{A}^{-1/2})$.

In this paper, by referring to an ellipsoid we mean the closed convex body in ν-dimensions rather than just its boundary. In 2-D, i.e, when $\nu = 2$, the ellipsoid is referred to as an ellipse. In this work, we will be using the above different forms of description for the same ellipse.

2.3 Problem Statement and Existing Techniques

2.3.1 Problem Statement

We denote the intersection region of the ellipses $\xi_i = \{\boldsymbol{x} \in \mathbb{R}^2 : \|\boldsymbol{B}_i \boldsymbol{x} + \boldsymbol{d}_i\| \leq 1\}$, for $i = \{1, \ldots, M\}$, by

$$\mathcal{E} = \bigcap_{i=1}^{M} \xi_i, \tag{4}$$

where without loss of generality we assume these ellipses are distinct. Throughout this work, we refer to $\mathcal{E}$ as the *feasible region*, and assume that it is a non-empty region. The problem is to find the smallest area ellipse $\xi_0 = \{\boldsymbol{x} \in \mathbb{R}^2 : \|\boldsymbol{B}_0 \boldsymbol{x} + \boldsymbol{d}_0\| \leq 1\}$ such that it contains the feasible region, i.e.,

$$\xi_0 \supseteq \mathcal{E}. \tag{5}$$

Given ξ_0, verifying that (5) holds is NP-complete, and thus finding the smallest ellipse ξ_0 such that (5) holds is not tractable [9, p.44]. However, there are some sub-optimal solutions to find less tight ellipsoids in arbitrary dimensions, which will be described below.

2.3.2 Popular Existing Techniques

Different techniques have been proposed for finding a sub-optimal solution to the aforementioned problem. Below, we describe two of the most popular techniques, each of which formulates the problem as a standard convex optimization one, which is solvable in polynomial time.

i) *Approximation Using the Sufficient Condition*: It is shown in [9, p.44] that by using the so called $\mathcal{S}$-procedure, the sufficient condition for (5) to hold can be expressed as a linear matrix inequality (LMI):

$$\begin{bmatrix} \boldsymbol{A}_0 & \boldsymbol{b}_0 & 0 \\ \boldsymbol{b}_0^T & -1 & \boldsymbol{b}_0^T \\ 0 & \boldsymbol{b}_0 & -\boldsymbol{A}_0 \end{bmatrix} - \sum_{i=1}^{M} \tau_i \begin{bmatrix} \boldsymbol{A}_i & \boldsymbol{b}_i & 0 \\ \boldsymbol{b}_i^T & c_i & 0 \\ 0 & 0 & 0 \end{bmatrix} \preceq 0, \tag{6}$$

where τ_i for $i = \{1, \ldots, M\}$ are positive unknowns to be estimated, and $\boldsymbol{A}_i$, $\boldsymbol{b}_i$, and c_i are related to $\boldsymbol{B}_i$ and $\boldsymbol{d}_i$ based on the definitions given earlier in Section 2.2. Note that $\boldsymbol{A}_0$, $\boldsymbol{b}_0$, and c_0 are normalized such that $c_0 = \boldsymbol{b}_0^T \boldsymbol{A}_0^{-1} \boldsymbol{b}_0 - 1$. This LMI is not a necessary condition for (5) to hold, and thus with (6) we cannot characterize all the ellipses that cover the intersection of multiple ellipses. However, among all the ellipses ξ_0 with variables $\boldsymbol{A}_0, \boldsymbol{b}_0$ satisfying (6), one can find the best outer approximation of the intersection region of $\xi_1, \ldots, \xi_M$ by solving the following semi-definite programming (SDP) problem:

$$\begin{aligned} &\min_{\boldsymbol{A}_0, \boldsymbol{b}_0, \tau_1, \ldots, \tau_M} \quad \log\det \boldsymbol{A}_0^{-1} \\ &\text{subject to} \quad \boldsymbol{A}_0 \succ 0,\ \tau_1, \ldots, \tau_M \geq 0,\ (6) \end{aligned} \tag{7}$$

ii) *Ellipse Obtained by Expansion of the Largest Inscribed Ellipse*:

Another approach [17, p.414] is to first find the maximum area ellipse $\xi_{max} = \{\boldsymbol{x} : \boldsymbol{x} = \boldsymbol{P}_0 \boldsymbol{y} + \boldsymbol{x}_{c_0}, \|\boldsymbol{y}\| \leq 1\}$ inscribed by the intersection of several ellipses by solving

$$\begin{aligned} &\max_{\boldsymbol{P}_0, \boldsymbol{x}_{c_0}, \tau_1, \ldots, \tau_M} \quad \log\det \boldsymbol{P}_0 \\ &\text{subject to} \quad \begin{bmatrix} -\tau_i - c_i + \boldsymbol{b}_i^T \boldsymbol{A}_i^{-1} \boldsymbol{b}_i & \boldsymbol{0} & (\boldsymbol{x}_{c_0} + \boldsymbol{A}_i^{-1} \boldsymbol{b}_i)^T \\ \boldsymbol{0} & \tau_i \boldsymbol{I}_\nu & \boldsymbol{P}_0 \\ \boldsymbol{x}_{c_0} + \boldsymbol{A}_i^{-1} \boldsymbol{b}_i & \boldsymbol{P}_0 & \boldsymbol{A}_i^{-1} \end{bmatrix} \succeq 0, \\ &\qquad \boldsymbol{P}_0 \succ 0,\ \tau_i \geq 0, \qquad\qquad i = 1, \ldots, M \end{aligned} \tag{8}$$

which is a convex SDP optimization problem whose solution can be obtained efficiently. Then as shown by John and Lowner, e.g., see [18] and the references therein, by scaling this ellipse by a factor of $\nu = 2$ (since we are currently considering a 2-D space), an ellipse covering the intersection of multiple ellipses can be obtained, see [17, p.414].

Either of the above methods can be applied to find the ellipsoidal outer approximation of the intersection of ellipsoids with arbitrary dimensions. However, due to the approximations made, the obtained ellipses might not always be tight (which will be shown through simulations), limiting their applicability. In the following section, we develop a geometrical bounding method in 2-D space, which through numerical evaluation, is shown to offer a tighter outer approximation of the intersection of multiple ellipses.

3 Proposed Bounding Ellipse Method

Our solution to the problem consist of two stages: (i) finding a tight polygon, which contains $\mathcal{E}$, and (ii) finding the tightest ellipse, which contains the vertices of the obtained polygon. The second stage involves a well-known optimization problem, which can be solved by iterative optimization techniques, as done in [6, 20, 23, 24]. It can also be formulated as a standard convex optimization problem and solved by standard optimization packages on a computer [17, p. 411]. The main aim of this paper is to find a tight polygon, with a small number of vertices, to cover $\mathcal{E}$ such that the smallest area ellipse which contains the vertices of the polygon is a tighter outer-approximation of $\mathcal{E}$ compared to the ellipses obtained by other approximation techniques mentioned in Section 2.3.2. In the following, we describe the two main stages of our proposed technique in more detail.

3.1 Finding a Tight Polygon Containing the Intersection of Ellipses

We divide the task of finding a polygon containing $\mathcal{E}$ into several steps as follows:

Step 1 (Finding the intersection points of ellipses on the boundary of $\mathcal{E}$): First, the intersection points of the boundaries of every pair of ellipses are found and the ones not lying in $\mathcal{E}$ are rejected. Finding the intersection points of two ellipses can be done by computing the roots of a polynomial of degree 4, as discussed in [25]. The possible number of intersection points can vary from 0 to 4, and in the special case that the two ellipses are disks, this number could be 0, 1, or 2. If the number of intersection points is 0 or 1, then either the two ellipses have no intersection region, or one of them is contained in the other one. Since we assume that $\mathcal{E}$ is a non-empty region, one of the ellipses has to contain the other. In this case we should remove the larger ellipse in the process of finding a tight polygon. To this end, we generate one point on the boundary of each ellipse randomly, and if one of these points does not satisfy the defining inequality of the other ellipse, then the former ellipse is the larger one. In the worst case there are 4 intersection points for every pair of ellipses and since there are $M(M-1)/2$ different pairs of ellipses, there will be at most $2M(M-1)$ intersection points to be verified. Since checking if an intersection point satisfies the inequalities of the remaining $M-2$ ellipses takes $\mathcal{O}(M)$ operations, $\mathcal{O}(M^3)$ operations are sufficient to find the intersection points on the boundary of $\mathcal{E}$. The $\mathcal{O}(M^3)$ algorithm implemented herein is straightforward and can easily be implemented. It is also very efficient when the value of M is small. Theoretically faster algorithms exist that run closer to $\mathcal{O}(M^2)$ if the arrangement of the ellipses is computed [31]. For very large values of M such algorithms may turn out to be useful. However, they are also computationally involved, and it is not clear if they are useful in practice.

There is a possibility that some of the intersection points are non-distinct, e.g., more than two ellipses intersect at exactly the same point. Note that due

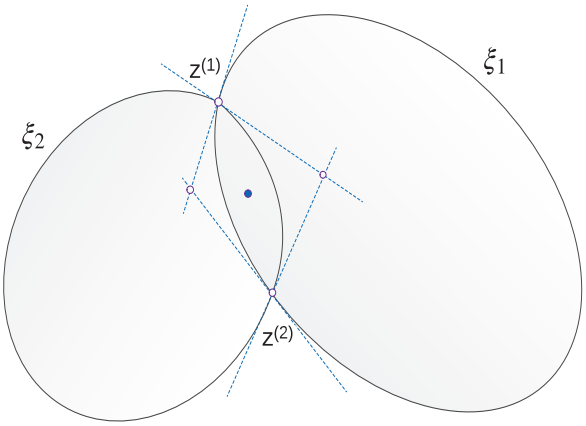


Fig. 1: The intersection of the half-planes tangent at the intersection points of the boundary of the ellipses forms a closed polygon.

to rounding errors, two nearby points might also be regarded as the same point. This does not result in error, however, the obtained polygon might be slightly less tight if one of these points is used in the algorithm. In these cases, we only use one of these points in our algorithm but keep the indices of the ellipses corresponding to these intersection points. For later use, the total number of intersection points remaining on the boundary of $\mathcal{E}$ is denoted by m_c.

Step 2 (Generating extra points on $\mathcal{E}$): After rejecting the intersection points not on the boundary of $\mathcal{E}$, we let $z^{(l)}$ for $l = 1, \ldots, m_c$ be the remaining intersection points. We find the mean of the these intersection points as

$$\boldsymbol{z}_{\text{mean}} = \frac{1}{m_c} \sum_{l \in m_c} \boldsymbol{z}^{(l)} \tag{9}$$

The vectors connecting $\boldsymbol{z}_{\text{mean}}$ to the intersection points are

$$\boldsymbol{v}^{(l)} = \boldsymbol{z}^{(l)} - \boldsymbol{z}_{\text{mean}}, \quad l = 1, \ldots, m_c \tag{10}$$

and the angles they make with the x-axis in the Cartesian coordinates are denoted by $\alpha^{(l)} \in [0, 2\pi)$. The vectors $\boldsymbol{v}^{(l)}$ are then sorted according to increasing angles. The intersection points may be used to generate a polygon covering $\mathcal{E}$ as shown in Fig. 1. However, as observed, the generated polygon may not be tight enough if only the intersection points are used. Furthermore, we might face degeneracy problems (to be illustrated with examples) where the intersection of half-planes forms an unbounded polygon which can not be used as a finite outer-approximation of the feasible region. To overcome these problems,

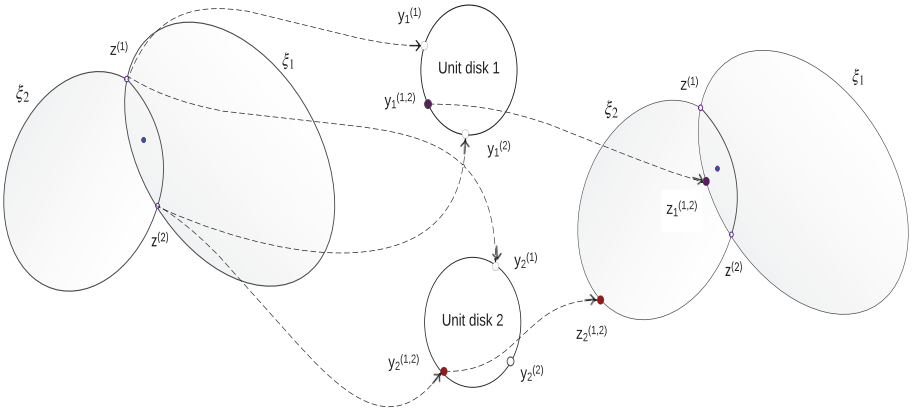
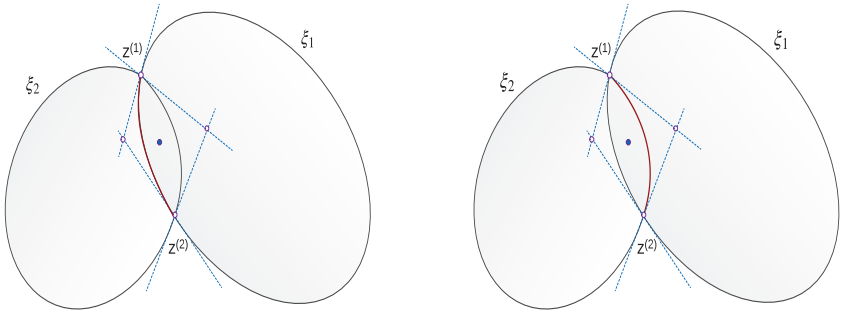


Fig. 2: Detecting the curve connecting $\boldsymbol{z}^{(1)}$ to $\boldsymbol{z}^{(2)}$ in a counter-clockwise manner.

we generate a number of additional points on the elliptic arc segments of $\mathcal{E}$ between two neighbouring intersection points.



(a) (b)

Fig. 3: The detected arc segments and the corresponding ellipses forming $\mathcal{E}$. (a) The segment (shown with red) is detected to be from ξ_1, connecting $\boldsymbol{z}^{(1)}$ and $\boldsymbol{z}^{(2)}$. (b) The segment (shown with red) is from ξ_2, connecting $\boldsymbol{z}^{(2)}$ and $\boldsymbol{z}^{(1)}$.

To do this, we need to know the ellipse corresponding to each elliptic arc segment forming the boundary of $\mathcal{E}$. Note however that as mentioned

earlier, an intersection point $\boldsymbol{z}^{(l)}$ could be related to more than two ellipses. Thus, in general, the two neighbouring points correspond to several different ellipses (minimum of two different ones), among which some are common. Obviously, the elliptic arc segment on the boundary of $\mathcal{E}$ that connects the two intersection points corresponds to an ellipse, the boundary of which passes through both neighbouring points. Thus the index of this ellipse is common to both intersection points.

Hence we remove the indices of the ellipses that are not common to both neighbouring intersection points, and assume that the remaining indices form a set, temporarily denoted as $\mathcal{P}$. With two or more indices left, there are multiple elliptic arc segments that connect the two intersection points, among which only one of them is part of the boundary of $\mathcal{E}$; thus there is an ambiguity in knowing the ellipse corresponding to that arc segment. To resolve this ambiguity and detect the correct indices, we randomly generate one point on the boundary of each ellipse, with index $k \in \mathcal{P}$, between the two intersection points (in a counter-clockwise manner) and see which point satisfies all the inequalities of the remaining ellipses, i.e., it falls inside or on the boundary of all the ellipses, whose indices are in $\mathcal{P}$. Specifically, to generate a point randomly on each ellipse, the two neighbouring intersection points, $\boldsymbol{z}^{(l_c)}$ and $\boldsymbol{z}^{(l_c+1)}$ where $l_c \in \{1, \ldots, m_c - 1\}$, are mapped onto the boundary of unit disks based on the equations of the corresponding ellipses, i.e., the inverse mapping defined in (1), as

$$\boldsymbol{y}_k^{(1)} = \boldsymbol{P}_k^{-1}(\boldsymbol{z}^{(l_c)} - \boldsymbol{x}_{c,k}) \tag{11}$$

$$\boldsymbol{y}_k^{(2)} = \boldsymbol{P}_k^{-1}(\boldsymbol{z}^{(l_c+1)} - \boldsymbol{x}_{c,k}) \tag{12}$$

Assume that the two generated points on the k-th disk are denoted by $\boldsymbol{y}_k^{(1)}$ and $\boldsymbol{y}_k^{(2)}$ and the angles corresponding to these points are denoted by $\theta_k^{(1)}$ and $\theta_k^{(2)}$, respectively. Now we generate one point on the arc corresponding to the k-th disk connecting $\boldsymbol{y}_k^{(1)}$ and $\boldsymbol{y}_k^{(2)}$ and then transform this point $\boldsymbol{y}_k^{(1,2)}$ back onto the k-th ellipse by means of:

$$\boldsymbol{z}_k^{(1,2)} = \boldsymbol{P}_k \boldsymbol{y}_k^{(1,2)} + \boldsymbol{x}_{c,k}, \tag{13}$$

Then by verifying if $\boldsymbol{z}_k^{(1,2)}$ falls inside or on the boundary of all the remaining ellipses with indices in $\mathcal{P}$, we can determine the ellipse which is forming the boundary of $\mathcal{E}$. The actions done so far in *Step 2* are summarized in Algorithm I, lines 14-23.

The process of detecting the ellipses which form $\mathcal{E}$ is shown with a simple example in Fig. 2. The two ellipses, labeled as ξ_1 and ξ_2, have two intersection points $\boldsymbol{z}^{(1)}$ and $\boldsymbol{z}^{(2)}$, which are first ordered based on the angles the corresponding vectors $\nu^{(l)}$ (10) make with the x-axis. We start from the point with the smallest angle, i.e., $\boldsymbol{z}^{(1)}$ and try to find the curve connecting it to the neighbouring point with the next smallest angle, which is $\boldsymbol{z}^{(2)}$, as shown in Fig. 2. The two points are mapped onto the boundary of unit disks based on

the equations of each ellipse, yielding, $\boldsymbol{y}_1^{(1)}$ and $\boldsymbol{y}_1^{(2)}$ for the disk corresponding to ξ_1 and $\boldsymbol{y}_2^{(1)}$ and $\boldsymbol{y}_2^{(2)}$ for the disk corresponding to ξ_2. Then a point is generated on the boundary of each disk on the arc connecting the two mapped points in a counterclockwise manner, yielding $\boldsymbol{y}_1^{(1,2)}$ and $\boldsymbol{y}_2^{(1,2)}$. The obtained points, $\boldsymbol{y}_1^{(1,2)}$ and $\boldsymbol{y}_2^{(1,2)}$ are then mapped back onto the ellipse corresponding to each disk, yielding $\boldsymbol{z}_1^{(1,2)}$ and $\boldsymbol{z}_2^{(1,2)}$, respectively. Since $\boldsymbol{z}_1^{(1,2)}$ on ξ_1 is the point which falls on $\mathcal{E}$, the corresponding elliptic arc segment, shown in red in Fig. 3-(a), is detected to be the one sought after, and ξ_1 is the corresponding ellipse. The same process is repeated for detecting the elliptic arc connecting $\boldsymbol{z}^{(2)}$ to $\boldsymbol{z}^{(1)}$ in a counter clockwise manner, which becomes the one corresponding to ξ_2, as shown in red in Fig. 3-(b).

Suppose that the j-th ellipse remains after removing all the indices of the irrelevant ellipses. Then the next task is to generate a number of points on the elliptic arc segment of the j-th ellipse, between the two neighbouring intersection points, e.g., $\boldsymbol{z}^{(l_c)}$ and $\boldsymbol{z}^{(l_c+1)}$, where $l_c \in \{1, \ldots, m_c - 1\}$. To do so, we first generate a certain number of points on the boundary of the unit disk obtained by inverse mapping of the ellipse ξ_j, between the points $\boldsymbol{y}_j^{(1)}$ and $\boldsymbol{y}_j^{(2)}$ in a counter-clockwise manner. The angles between a reference axis and the vectors connecting these points $\boldsymbol{y}_j^{(1)}$ and $\boldsymbol{y}_j^{(2)}$ to the origin, are computed and denoted as $\theta_j^{(1)}$ and $\theta_j^{(2)}$, respectively. Depending on the difference between $\theta_j^{(1)}$ and $\theta_j^{(2)}$ we can generate a number of points on the unit circle. For instance, if in total it is desired to generate m points on each circle, the arc length between every two points is $2\pi/m$. Thus we may want to generate the points $\boldsymbol{y}_{1,2}^{(l_j)}$ between $\boldsymbol{y}_j^{(1)}$ and $\boldsymbol{y}_j^{(2)}$ on the unit circle, for $l_j = 1, \ldots, \texttt{floor}[(\theta_j^{(1)} - \theta_j^{(2)})m/(2\pi)]$, where `floor[.]` returns the largest integer not greater than its argument. After generating the points $\boldsymbol{y}_{1,2}^{(l_j)}$ on the unit circle, they are transformed back onto the j-th ellipse by means of

$$\boldsymbol{z}_j^{(l_j)} = \boldsymbol{P}_j \boldsymbol{y}_{1,2}^{(l_j)} + \boldsymbol{x}_{c,j}, \quad l_j = 1, \ldots, \texttt{floor}[(\theta^{(1)} - \theta^{(2)})m/(2\pi)] \tag{14}$$

The half planes, tangent to the j-th ellipse at the corresponding generated points are computed as follows

$$(\boldsymbol{B}_j \boldsymbol{z}_j^{(l_j)} + \boldsymbol{d}_j)^T (\boldsymbol{B}_j \boldsymbol{x} + \boldsymbol{d}_j) \leq 1, \quad l_j = 1, \ldots, \texttt{floor}[(\theta^{(1)} - \theta^{(2)})m/(2\pi)] \tag{15}$$

and the half planes, tangent to the j-th ellipse at the two ends of the arc segment under consideration, i.e., $\boldsymbol{z}^{(l_c)}$ and $\boldsymbol{z}^{(l_c+1)}$ are calculated as

$$(\boldsymbol{B}_j \boldsymbol{z}^{(l)} + \boldsymbol{d}_j)^T (\boldsymbol{B}_j \boldsymbol{x} + \boldsymbol{d}_j) \leq 1, \quad l = l_c, l_c + 1 \tag{16}$$

This process of generating points using mapping on to a unit disk is summarized in Algorithm I, lines 24-28.

In this way, for each arc of the feasible region connecting the two intersection points, a number of discrete points, denoted by m_d, are generated and the half-planes, tangent to the ellipse corresponding to that segment (e.g., j-th ellipse) are computed. For the m_c intersection points also $2m_c$ half planes

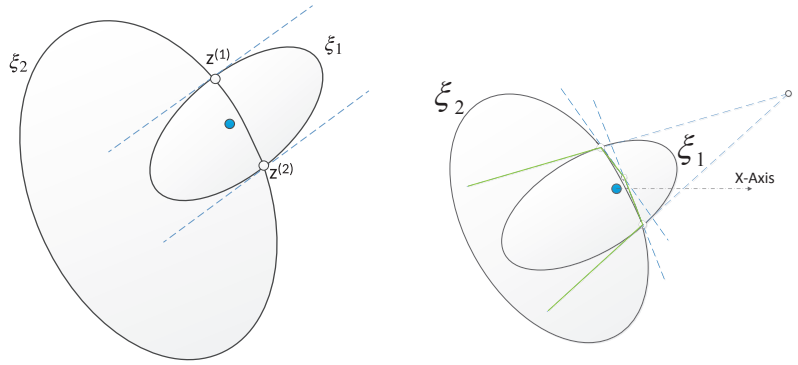
are obtained. Therefore, in total there will be $m_d + m_c$ discrete points with $m_d + 2m_c$ half planes corresponding to these points.

Step 3 (Intersecting the tangent lines to find the vertices of the polygon): The intersection of the obtained m_d+2m_c half planes usually forms a bounded and convex polygon. One way to compute the vertices of the polygon is to use the divide-and-conquer algorithm [26], or simplified versions thereof using the techniques proposed in [27,28]; in which the cost is $\mathcal{O}\Big((m_d+m_c)\log(m_d+m_c)\Big)$. However, we will find the polygon more efficiently for the 2-D case as explained below. Since $\mathcal{E}$ is convex and its boundary is piecewise smooth, it can be observed that each vertex of the bounding polygon is the intersection of two tangent lines corresponding to two neighbouring points. Thus to obtain these vertices, we solve the linear system of equations corresponding to the two tangent lines at two neighbouring points on each segment. Since we only need to solve $m_d + m_c$ systems of linear equations (the intersection of tangent lines at the intersection points of ellipses are already obtained), only $\mathcal{O}(m_d + m_c)$ flops are sufficient. This step is summarized in lines 24-29 of Algorithm I.

Step 4 (Detecting the degeneracy problem): Even after the above steps, there is a possibility of degeneracy problems, i.e., the polygon formed is not bounded or convex and does not cover $\mathcal{E}$. These two situations, illustrated in Fig. 4, are as follows:

- If the tangent lines corresponding to two neighbouring points do not intersect, i.e., the two lines are parallel, the polygon will be unbounded. Therefore, the number of points on the elliptic arc connecting two neighbouring points needs to be increased.
- If the intersection exists but does not satisfy the remaining of the m_d+2m_c affine inequalities, this point can not be a vertex of the desired polygon. Note that the desired polygon should be formed as a result of intersecting half-planes, and if the intersection region is unbounded, a bounded convex polygon which contains $\mathcal{E}$ can not be found by relying only on these half-planes. Consequently, the tangent lines corresponding to two neighbouring points cannot be the sides of the polygon, and thus there are no support lines to $\mathcal{E}$ at these points.

Detecting the first case is simple as the intersection of two parallel lines has no solution. To detect the second case, we find the intersection of the tangent lines corresponding to every two neighbouring points and verify if it satisfies all the affine inequalities corresponding to the half planes of the other discrete points. This can be done with $\mathcal{O}(m_d + m_c)$ flops as there are $m_d + 2m_c - 2$ remaining affine inequalities to be verified for every obtained point. Detecting the degeneracy can be done for each arc segment just after *Step 3* is implemented for that segment. If degeneracy occurs, then we only increase the number of points on the specific arc segment by generating $\texttt{floor}[(\theta^{(1)} - \theta^{(2)})(m + \Delta m)/(2\pi)]$ points on the unit circle and then mapping them onto the corresponding ellipse, where Δm is a relatively small integer. One strategy is to always double the value of m so it can guarantee to quickly find a number that can avoid degeneracy.



(a) (b)

Fig. 4: The degenerate cases where a closed polygon can not be obtained: (a) parallel tangent lines, (b) the intersection point does not satisfy the affine inequalities corresponding to the other half-planes.

At the end of these steps, the intersection of every pair of tangent lines corresponding to two neighbouring points, as well as the previously obtained intersection points of every pair of ellipses, represent the m_p vertices of the desired polygon.[1]

3.2 Finding the Tightest Ellipse Containing the Polygon

Assume that we have found a relatively tight polygon, represented by vertices $\tilde{\boldsymbol{z}}^{(l)}$ for $l = 1, \ldots, m_p$, which covers $\mathcal{E}$. Then one can find the smallest area ellipse (minimum spanning ellipse) which contains the vertices of this polygon (and hence contains $\mathcal{E}$) as done in [6, 20, 23, 24]. This problem can also be formulated as a standard convex optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{B}_0, \boldsymbol{d}_0} \quad \log\det \boldsymbol{B}_0^{-1} \\ &\text{subject to } \|\boldsymbol{B}_0 \tilde{\boldsymbol{z}}^{(l)} + \boldsymbol{d}_0\| \leq 1, \quad l = 1, \ldots, m_p \end{aligned} \tag{17}$$

where $\log\det \boldsymbol{B}_0^{-1}$ is proportional to the area of the ellipse [17]. Since the inequalities can also be written as an LMI, this optimization problem can be formulated as a standard SDP. If the feasible region is tightly outer-bounded by the polygon, then it is very likely that it is tightly outer-approximated by the bounding ellipse.

[1] Note that the generated intersection points on the boundary of $\mathcal{E}$, excluding the intersection points of ellipses, are not required to represent the polygon because they lie on its sides.

3.3 Algorithm Summary and Remarks

The proposed method is summarized in Algorithm 1. Sometimes, one is interested in obtaining a desired bounding ellipse with a certain level of tightness, and may not know in advance how to chose an appropriate m. To make the algorithm useful for such applications, after solving (17), we can increase the number of points by generating one point between every two points that we have already generated. Then we can compare the area of the updated bounding ellipse with the one corresponding to the previous value of m. If their difference is more than a predefined threshold then we continue this process. If for the initial m, a bounded polygon cannot be formed, we increase m until a bounded polygon is formed, and then proceed as described above. By iterating this process, we can determine if the area of the bounding ellipse obtained by the proposed technique has converged to some limit, and thus a possibly tight ellipse is obtained.

4 Numerical Results

In this section, we compare the performance of different algorithms in obtaining an ellipse to cover the intersection of several ellipses. To this end, we use Matlab 2010b on a 64-bit computer with Intel i7-2600 3.4GHz processor and 12GB of RAM. The metric utilized for evaluation is the area of the bounding ellipse a_E, which based on the different formats given in Section 2.2, is equal to $a_\nu \mathsf{det}(\boldsymbol{P})$, $a_\nu \mathsf{det}(\boldsymbol{B}^{-1})$, or $a_\nu \mathsf{det}(\boldsymbol{A}^{-1/2})$, where $a_\nu = \pi$ is the area of the unit disk in 2D. We randomly generate $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 20$ intersecting ellipses to see the performance of our technique in different situations. For comparison, we solve the optimization problem in (7) and denote this method as *S-procedure*. We also consider finding the ellipse by solving (8) and then expanding it by a factor of $\nu = 2$ to find an ellipse covering the intersection region. This method is denoted as *Expanded* in the rest of this paper. We also consider the technique proposed in [19] where a bounding polygon is obtained in a different way than the method proposed in this paper. However, the optimization problem in (17) is finally solved to find the smallest area ellipse covering the polygon. This method is denoted as *YWCC16* throughout this section. For solving the optimization problems we use Sedumi solver [29] and CVX optimization toolbox for Matlab [30]. We compare the performance of our proposed method, denoted as *Proposed*, with the aforementioned approaches.

The areas of the obtained bounding ellipses, as well as the computation times for each method are given below for different scenarios. The *Expanded* and *S-Procedure* methods do not depend on m (i.e., the number of discrete points generated on each ellipse in *YWCC16* and *Proposed*), thus their performances do not change. However, *YWCC16* and *Proposed* depend on m and in general the area of the obtained bounding ellipse decreases with m. We also compare the computation time of each method for the corresponding values

Algorithm 1 Ellipse Outer-approximation

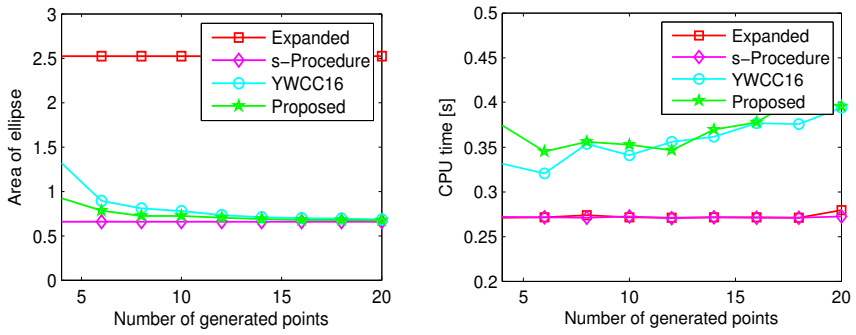
Set a predefined maximum iteration number k_{max}
for $k_{\text{Iter}} = 1$ to $k_{\max}$ **do**
  for every pair of ellipses **do**
    Find the intersection points of the ellipses.
    if the number of intersection points is 0 or 1 **then**
      Find the ellipse containing the other one and remove it from the set of intersecting ellipses.
    end if
    if the number of remaining ellipses is equal to 1 **then**
      Use that ellipse as the tightest outer approximation of the feasible region.
      End the algorithm.
    end if
  end for
  Reject the intersection points not lying on the boundary of $\mathcal{E}$ and get $\boldsymbol{z}^{(l)}$ for $l = 1, \ldots, m_c$.
  Find the mean of the intersection points and find the vector connecting the former to the latter.
  Find the angles between the x-axis and these vectors and sort them according to their angles.
  for every two neighbour points **do**
    Remove the indices of the ellipses not common between two neighbouring intersection points.
    if $|\mathcal{P}| \geq 2$ **then**
      For each ellipse with index $k \in \mathcal{P}$, map the points onto the corresponding unit circle.
      For $\boldsymbol{y}_k^{(1)}$ and $\boldsymbol{y}_k^{(2)}$, find the angles between the reference axis and the vectors connecting them to the centre of the unit circle, i.e., $\theta_k^{(1)}$ and $\theta_k^{(2)}$, respectively.
      For every pair of points, generate one point on the unit circle with angle between $\theta_k^{(1)}$ and $\theta_k^{(2)}$.
      Map every point back onto the corresponding ellipse and verify if it lies on the feasible region.
    end if
    Denote the remaining ellipse with index j.
    Map the two neighbouring points $\boldsymbol{z}^{(l_c)}$ and $\boldsymbol{z}^{(l_c+1)}$ onto a unit circle through (11) and (12).
    Find the angles between the Cartesian axis and the vector connecting $\boldsymbol{y}_j^{(1)}$ and $\boldsymbol{y}_j^{(2)}$ to the centre of the unit circle, i.e., $\theta_j^{(1)}$ and $\theta_j^{(2)}$, respectively.
    Generate $\texttt{floor}[(\theta_j^{(1)} - \theta_j^{(2)})m/(2\pi)]$ points on the curve between $\boldsymbol{y}_j^{(1)}$ and $\boldsymbol{y}_j^{(2)}$.
    Map the points back onto the ellipse and find the tangent lines to the curve at those points.
    Find the intersection of every two neighbouring points to obtain the vertices of the polygon $\tilde{z}^{(l)}$.
    Do the test in *Step 4* to check if a degenerate case has happened.
    if degeneracy occurs **then**
      $m \leftarrow m + \Delta m$ and go to line (27).
    end if
  end for
end for

of m and see which one generates a tighter ellipse with a lower computational cost. Since evaluating the computational cost in terms of number of operations is exhaustive and difficult, we use CPU time as a metric for comparison of computational cost.

Fig. 5-(a) illustrates the area of the obtained ellipse for the scenario of $M = 2$ ellipses. As observed, *S-Procedure* has the best performance among all, and the area obtained by *Proposed* converges to the one obtained by *S-Procedure* when m grows. *YWCC16* has a good performance as long as $m \geq 6$ because for small m there are not enough points to form a tight polygon. The *Expanded-inner* has the worst performance among all with area of 2.5 units. The computation times of the different methods are given in Fig. 5-(b). As observed, *YWCC16* and *Proposed* have slightly higher computational cost, however, increasing the number of initial discrete points m does not change the computation times of *YWCC16* and *Proposed* noticeably.

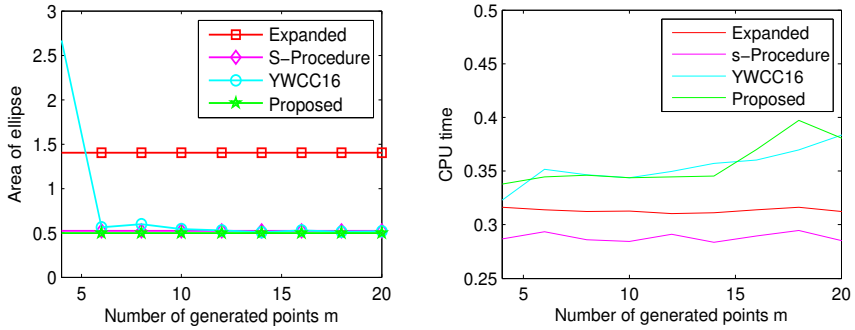
In Fig. 6-(a), we compare the area obtained by different methods for the scenario in which three ellipses intersect. As observed, *Proposed* and *S-Procedure* have the best performance, while *YWCC16* starts to converge by increasing m. *Expanded* has the worst performance again. The computation times of different techniques are also given in Fig. 6-(b). The CPU time for *Expanded* and *S-Procedure* are almost the same while *YWCC16* and *Proposed* have higher run-times, which grow gradually with increasing m.

In Fig. 7-(a), we compare the area obtained by different methods for the scenario in which five ellipses intersect. Once again, *Proposed* outperforms all the other techniques. *YWCC16* faces degeneracy problems with the choice of $m = 4$. However, for the other values of m it outperforms the *S-Procedure* and *Expanded*. The computation times of different techniques are also in the same range, as shown in Fig. 7-(b), while *Proposed* yields a slightly lower cost. Note that *Proposed* obtains a tighter result than *YWCC16*, with similar computational cost.



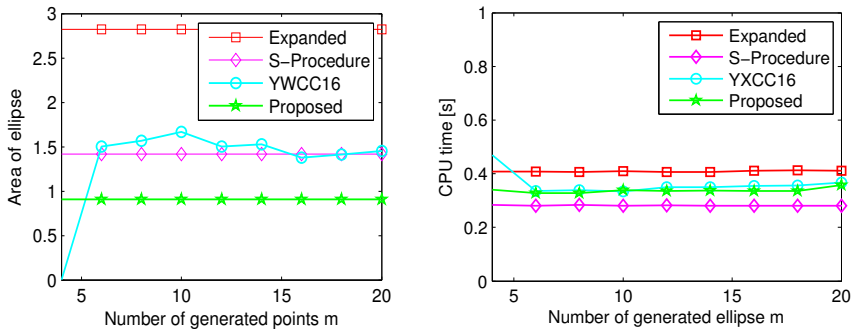
(a) (b)

Fig. 5: Numerical test for M=2: (a) Areas of bounding ellipses, (b) CPU times.



(a) (b)

Fig. 6: Numerical test for M=3: (a) Areas of bounding ellipses, (b) CPU times.

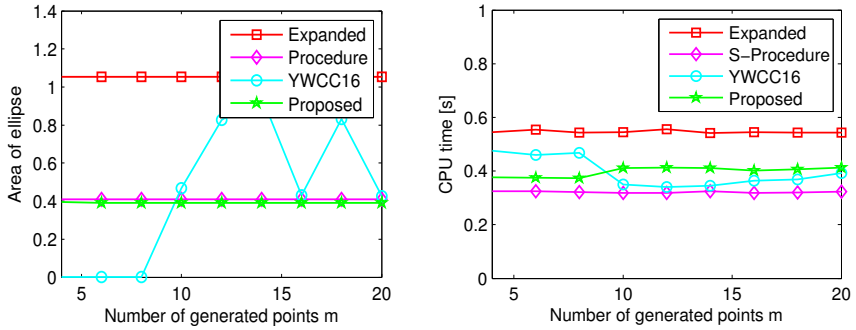


(a) (b)

Fig. 7: Numerical test for M=5: (a) Areas of bounding ellipses, (b) CPU times.

In Fig. 8-(a), the number of ellipses has increased to $M = 10$. Once again, *Proposed* outperforms all the other techniques and *S-Procedure* has very similar performance. *YWCC16* faces degeneracy problems with small m and the area is stated to be zero, however, for the large values of m, i.e., $m \geq 10$ it yields a bounding ellipse, but the tightness is only sometimes better than *Expanded*. The computation times of different techniques are also in the same range, as shown in Fig. 8-(b), while *S-Procedure* yields a slightly lower cost. The computational cost of *Proposed* is lower than *YWCC16* when similar bounding ellipses are obtained, i.e., for $m = 10$, $m = 16$, and $m = 20$.

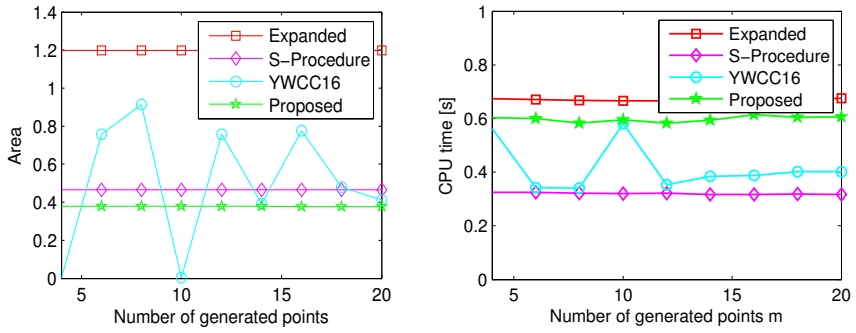
In Fig. 9-(a), the number of ellipses has increased to $M = 20$. Once again, *Proposed* outperforms all the other techniques and *S-Procedure* has very similar performance. *YWCC16* faces degeneracy problems with some values of m such as $m = 4$ and $m = 10$ and the area is equal to 0. However, for the other values of m it works, but is only sometimes better than *Expanded*. Only for $m > 18$ it seems to start converging to somewhere close to the area of the



(a) (b)

Fig. 8: Numerical test for M=10: (a) Areas of bounding ellipses, (b) CPU times.

ellipse obtained by *Proposed*. The computation times of different techniques are shown in Fig. 9-(b). The *S-Procedure* yields the lowest cost, while *Proposed* has slightly higher cost and *Expanded* has the worst computation time. The proposed method *Proposed* has a higher CPU time than *YWCC16*, however, the latter does not work properly for small values of m. Only for larger values of m it starts yielding similar bounding ellipses. When the number of ellipses grows, it takes $\mathcal{O}(M^3)$ for *Proposed* to find the intersection points of ellipses. Although this might appear to be a disadvantage of the proposed method, the application where the intersection of very large number of ellipses is required is very limited to the authors' knowledge.



(a) (b)

Fig. 9: Numerical test for M=20: (a) Areas of bounding ellipses, (b) CPU times.

Remarks:

- From Fig. 5-(a), Fig. 6-(a), Fig. 7-(a), Fig. 8-(a), and Fig. 9-(a), we observe that the area of the bounding ellipse obtained by *YWCC16* does not always decrease when m increases slightly. This is because some of the discrete points that are generated on ellipse ξ_i that lie on the intersection region, may not necessarily be on the intersection region when m is increased. This situation happens mostly when m is increased by a small amount say 2 or 3. However, *Proposed* does not face this issue because the number of discrete points on each arc segment of the feasible region remains either the same or increases when m is increased. This is a salient advantage of *Proposed* compared to *YWCC16*.
- From Fig. 5-(b), Fig. 6-(b), Fig. 7-(b), Fig. 8-(b), and Fig. 9-(b), we observe that the computation times of *YWCC16* and *Proposed* do not necessarily increase when m is increased slightly. The first reason is that sometimes when m is increased slightly, the number of remaining discrete points on the intersection region does not necessarily increase and hence m_p remains the same; thus the optimization problem in (17) does not change. Since the computation times of *YWCC16* and *Proposed* are also related to the number of constraints in (17) and this number might not change much by a slight increase in m, it is unlikely that the computation times increase noticeably. The second reason is that the CVX optimization packages does not necessarily take the same amount of time to solve the same optimization problem. This is because CVX uses an iterative interior point method that is initialized randomly and its convergence speed might be different every time it is utilized. The computation times of *S-Procedure* and *Expanded* methods also show that although for different m the same optimization problem is solved, the computation times are not exactly the same.

Although the computation costs of *Proposed* remained nearly constant for a small increase in m, we will show that when m is increased by a larger amount, the computational cost grows noticeably. The same behaviour is observed for *YWCC16*. In Fig. 10, the computation times of the different algorithms are plotted with respect to the logarithm of the number of discrete points, i.e., $\log_2(m)$. Since the other benchmark approaches do not depend on m, their computational costs are almost fixed. As observed, for small m, the computation times of *YWCC16* and *Proposed* are in the same range as those of *Expanded-inner* and *S-Procedure*. However, by increasing m further beyond 64, *YWCC16* and *YWCC16* will become more computationally demanding due to their dependence on the number of discrete points, both in the generation of the polygon, and in solving the optimization problem in (17). While this increase may seem to be a weak point of the proposed method, however, using large number of discrete points such as 1000 will be unnecessary for *Proposed*. This is because as the number of ellipses increases, the number of intersection points on $\mathcal{E}$ also increases and thus by only relying on these intersection points in order to find a polygon, a tight bounding ellipse can be obtained. Therefore, generating too many points on each arc segment be-

tween two neighbouring intersection points will not improve the tightness of the bounding ellipse noticeably.

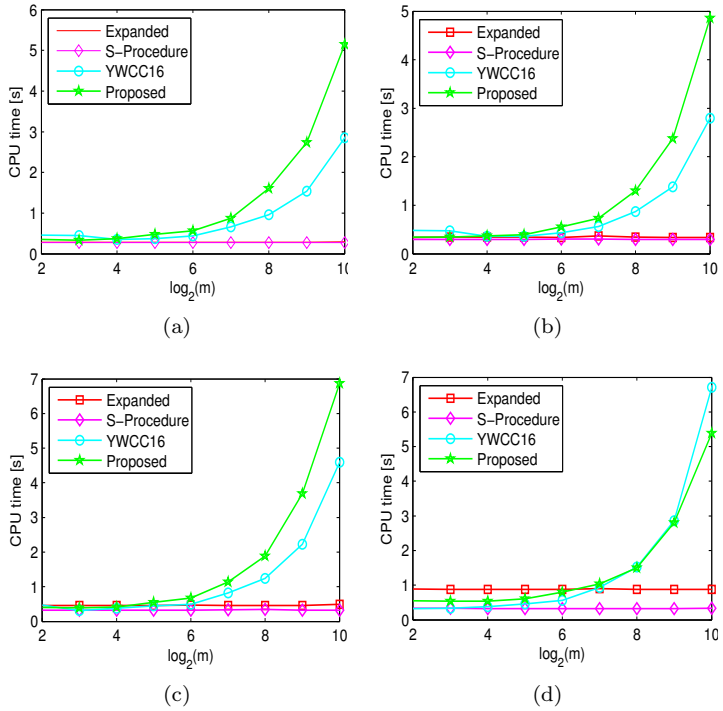


Fig. 10: Comparison of CPU time for different algorithms as a function of $\log_2(m)$ for the three scenarios with M ellipses: (a) $M = 3$, (b) $M = 5$, (c) $M = 10$, (d) $M = 20$.

5 Conclusion

In this paper, we developed and studied tight outer approximation of the intersection region of a finite number of ellipses in 2-D space. The main idea is to outer-approximate the feasible region by a tight polygon, and then find the smallest area ellipse containing the vertices of the polygon. To find the polygon, we proposed to first find a set of discrete points on the boundary of the intersection region, and by linearizing the curves at those points find the half planes which form the polygon. In order to generate the discrete points on the boundary of the intersection region we first determined the intersection points, and then generated a required number of points on each segment of the intersection region connecting the two neighbouring points. Through

numerical experiments, it was illustrated that the proposed method could offer a tighter outer-approximation of the intersection of ellipses compared to the conventional methods found in the literature with similar computational cost. Therefore, the proposed method, i.e., *Proposed*, offers the best trade-off between accuracy of the outer-approximation and computational cost, and hence will be preferred most of the time. In future work it may be worthwhile to program the techniques developed in [13] and [31], and apply them to the problem studied here to determine if they offer any practical advantages over the methods empirically investigated here.

References

1. H. Freeman and R. Shapira, "Determining the minimum-area encasing rectangle for an arbitrary closed curve," *Commun. ACM*, vol. 18, no. 7, pp. 409–413, Jul. 1975.
2. C.-T. Chang, B. Gorissen, and S. Melchior, "Fast oriented bounding box optimization on the rotation group so(3;r)," *ACM Trans. Graph.*, vol. 30, no. 5, pp. 122:1–122:16, Oct. 2011.
3. J. O'Rourke, "Finding minimal enclosing boxes," *International Journal of Computer and Information Sciences*, vol. 14, no. 3, pp. 183–199, 1985.
4. V. Klee and M. C. Laskowski, "Finding the smallest triangles containing a given convex polygon," *Journal of Algorithms*, vol. 6, no. 3, pp. 359 – 375, 1985.
5. M. Lahanas, T. Kemmerer, N. Milickovic, K. Karouzakis, D. Baltas, and N. Zamboglou, "Optimized bounding boxes for three-dimensional treatment planning in brachytherapy," *Medical Physics*, vol. 27, no. 10, pp. 2333–2342, 2000.
6. M. J. Post, "A minimum spanning ellipse algorithm," in *22nd Annual Symposium on Foundations of Computer Science*, Oct 1981, pp. 115–122.
7. G. Toussaint, "Applications of the rotating calipers to geometric problems in two and three dimensions," *International Journal of Digital Information and Wireless Communications (IJDIWC)*, vol. 4, no. 3, pp. 372–386, 2014.
8. A. Kurzhanski and I. Valyi, *Ellipsoidal Calculus for Estimation and Control*. Springer, 1994.
9. S. P. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*. SIAM, 1994, vol. 15.
10. D. Maksarov and J. Norton, "State bounding with ellipsoidal set description of the uncertainty," *International Journal of Control*, vol. 65, no. 5, pp. 847–866, 1996.
11. M. Gholami, H. Wymeersch, S. Gezici, and E. Strom, "Distributed bounding of feasible sets in cooperative wireless network positioning," *IEEE Commun. Letters*, vol. 17, no. 8, pp. 1596–1599, 2013.
12. J. Zhang and S. Sastary, "Distributed position etimation for sensor networks," in *Proc. 15th Triennial World Congress, Barcelona, Spain*, 2002.
13. J. Matousek, M. Sharir, and E. Welzl, "A subexponential bound for linear programming," *Algorithmica*, vol. 16, no. 4-5, pp. 498–516, 1996.
14. F. L. Chernousko, "Ellipsoidal bounds for sets of attainability and uncertainty in control problems," *Optimal Control Applications and Methods*, vol. 3, no. 2, pp. 187–202, 1982.
15. W. Kahan, "Circumscribing an ellipsoid about the intersection of two ellipsoids," *Can. Math. Bull*, vol. 11, no. 3, pp. 437–441, 1968.
16. L. Ros, A. Sabater, and F. Thomas, "An ellipsoidal calculus based on propagation and fusion," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 32, no. 4, pp. 430–442, 2002.
17. S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.
18. M. Henk, "Löwner–John ellipsoids," *Documenta Math.*, pp. 95–106, 2012.
19. S. Yousefi, H. Wymeersch, X. Chang, and B. Champagne, "Tight 2-dimensional outer-approximation of uncertainty regions in a wireless sensor network," *IEEE Communication Letters*, vol. 20, no. 3, pp. 570–573, 2016.

20. N. Z. Shor and O. Berezovski, "New algorithms for constructing optimal circumscribed and inscribed ellipsoids," *Optimization Methods and Software*, vol. 1, no. 4, pp. 283–299, 1992.
21. R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1990.
22. A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization*. Society for Industrial and Applied Mathematics, 2001.
23. B. W. Silverman and D. M. Titterington, "Minimum covering ellipses," *SIAM Journal on Scientific and Statistical Computing*, vol. 1, no. 4, pp. 401–409, 1980.
24. E. Welzl, "Smallest enclosing disks (balls and ellipsoids)," in *Results and New Trends in Computer Science*. Springer-Verlag, 1991, pp. 359–370.
25. D. Eberly, *Intersection of ellipses*. Geometric Tools, Jun. 2000, vol. 200. [Online]. Available: https://www.geometrictools.com/Documentation/IntersectionOfEllipses.pdf
26. F. Preparata and D. Muller, "Finding the intersection of n half-spaces in time o(n log n)," *Theoretical Computer Science*, vol. 8, no. 1, pp. 45 – 55, 1979.
27. G. Toussaint, "A simple linear algorithm for intersecting convex polygons," *The Visual Computer*, vol. 1, no. 2, pp. 118–123, 1985.
28. J. O'Rourke, C.-B. Chien, T. Olson, and D. Naddor, "A new linear algorithm for intersecting convex polygons," *Computer Graphics and Image Processing*, vol. 19, no. 4, pp. 384 – 391, 1982.
29. J. F. Strum, "Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones," 1998.
30. M. Grant, S. Boyd, and Y. Ye, "CVX: Matlab software for disciplined convex programming, version 2.1," Sep. 2014.
31. H. Edelsbrunner, L. Guibas, J. Pach, R. Pollack, R. Seidel, and M. Sharir, "Arrangements of curves in the planetopology, combinatorics, and algorithms," *Theoretical Computer Science*, vol. 92, no. 2, pp. 319–336, 1992.