Finding and Optimizing Certified, Collision-Free Regions in Configuration Space for Robot Manipulators

Alexandre Amice*[1], Hongkai Dai*[2], Peter Werner[1], Annan Zhang[1], and Russ Tedrake[1,2]

[1] MIT {amice, wernerpe, annanz, russt}@mit.edu
[2] Toyota Research Institute {hongkai.dai}@tri.global

Abstract. Configuration space (C-space) has played a central role in collision-free motion planning, particularly for robot manipulators. While it is possible to check for collisions at a point using standard algorithms, to date no practical method exists for computing collision-free C-space *regions* with rigorous certificates due to the complexities of mapping task-space obstacles through the kinematics. In this work, we present the first to our knowledge method for generating such regions and certificates through convex optimization. Our method, called C-IRIS (C-space Iterative Regional Inflation by Semidefinite programming), generates large, convex polytopes in a rational parametrization of the configuration space which are guaranteed to be collision-free. Such regions have been shown to be useful for both optimization-based and randomized motion planning. Our regions are generated by alternating between two convex optimization problems: (1) a simultaneous search for a maximal-volume ellipse inscribed in a given polytope and a certificate that the polytope is collision-free and (2) a maximal expansion of the polytope away from the ellipse which does not violate the certificate. The volume of the ellipse and size of the polytope are allowed to grow over several iterations while being collision-free by construction. Our method works in arbitrary dimensions, only makes assumptions about the convexity of the obstacles in the *task* space, and scales to realistic problems in manipulation. We demonstrate our algorithm's ability to fill a non-trivial amount of collision-free C-space in a 3-DOF example where the C-space can be visualized, as well as the scalability of our algorithm on a 7-DOF KUKA iiwa and a 12-DOF bimanual manipulator.

1 Introduction and Related Work

The notion of configuration space (C-space) has become a foundational idea in robot motion planning since its proposal in the seminal work [1]. In the presence

* equal contribution

This work was supported by Office of Naval Research, Award No. N00014-18-1-2210, National Science Foundation, Award No. EFMA-1830901., and Air Force Research Lab Award No. FA8750-19-2-1000

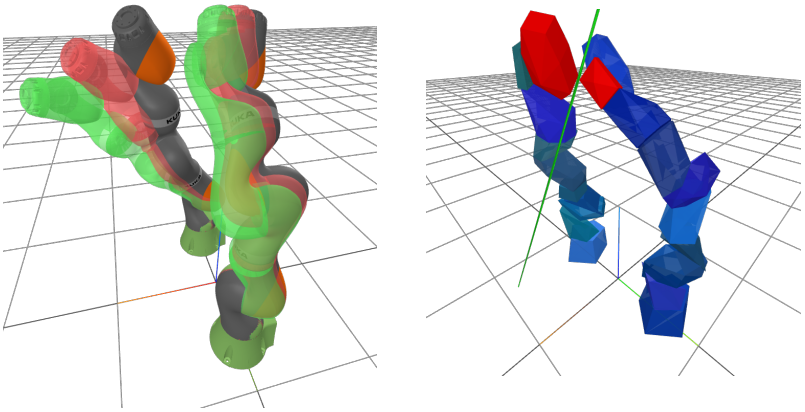

Fig. 1: Left: Multiple configurations sampled from one certified C-free region on a 12-DOF dual KUKA iiwa systems (with wrist joints fixed). Each configuration is visualized using one color. Right: Our certified C-free region is tight. At one of the sampled configurations, we draw the separating plane (green) as the non-collision certificate between the two highlighted polytopic collision geometries (red), with a distance of 7.3mm. Every link of the iiwa is approximated by one or several polytopes.

of obstacles in the Cartesian task space, a fundamental challenge is describing the collision-free C-space (C-free): the full range of configurations for which a robot is not in collision. Prior work has sought to describe a C-space obstacle from its task space description [2][3]. However it is recognized that a complete, mathematical description of C-space obstacles with high degree of freedom systems is intractable [4].

Despite this difficulty, describing C-free has been the subject of many works in robotics. Randomized, collision-free motion planners such as Rapidly Exploring Random Trees (RRT) [5], Probabilistic Road Maps (PRM) [6], and their variants can be seen as attempting to describe C-free via piecewise linear paths defined by the nodes and edges of the graph. Works such as [7][8][9] also seek to provide a description of C-free via randomized sampling. All of these methods guarantee that the configurations at the sample points are collision-free and attempt to provide a probabilistic certificate that nearby points contain no collisions. Unfortunately, such methods are inherently limited by the density of the samples used, which can become a barrier in higher dimensions. This work will overcome this barrier by providing a deterministic certificate that all of the infinitely many configurations in a given subset of C-space are collision-free. This certificate comes in the form of one parametric separating hyperplane for each pair of objects which can collide in a given environment such as the one seen in Fig 1. C-free will then be described as a union of these certified sets. Fig 2 shows an example of the complexity C-space obstacles for even a simple 3-DOF robot (the red mesh), as well as an example of several certified, collision-free polytopic regions produced by our method.

Describing C-free as a union of sets is not a new idea. In two and three dimensions and in the presence of polyhedral obstacles, this problem is equivalent to describing a non-convex polyhedron as a union of convex sets. It is known that finding a minimal such decomposition is NP-hard [10] to solve exactly and even APX-hard [11] to approximate[3]. Works such as [12] and [13] overcome these hardness results by finding decompositions that are unions of approximately convex sets.

A method of describing C-free in arbitrary dimensions is given by the IRIS algorithm in [14]. Under the assumption of known, convex obstacles in C-space, IRIS can rapidly grow convex, polytopic regions by alternating between two convex programs. The collision-free certificates of IRIS arise naturally due to the assumption of convexity of the obstacles. Unfortunately, it is often the case that obstacles are naturally described as convex sets in *task space* which are rarely convex in C-space.

Searching for certificates of non-collision in C-space when obstacles are specified as convex sets in the task space using convex programming (specifically Sum-Of-Squares (SOS) programming) is the primary technical contribution of this work. This is achieved by a novel formulation exploiting the well-known algebraic structure of the kinematics [15][16][17]. Similar to [14], we construct certified, collision-free polytopic regions by alternating between a pair of convex programs. Our method works in arbitrary dimensions and is the first to our knowledge to provide such certificates in this setting.

Describing C-free as a union of convex regions is particularly attractive as such descriptions have proven useful for optimization-based motion planning such as in [18][19]. Indeed, in our companion paper [20], the authors demonstrate great success in finding the globally-optimal collision-free trajectory for robot manipulators by planning through the types of regions generated by our method.

Our certified, convex regions also find natural use in randomized motion planners. By paying an upfront cost to describe C-free as a union of polytopic sets, piecewise-linear randomized motion planners can certify that both individual samples and edges connecting those samples are completely collision-free by simply checking membership in a finite number of polytopes.

We begin Section 2 by describing how non-collision of a single configuration can be certified via convex programming. In Section 3, we give essential background needed to describe the technical approach described in Section 4, where we extend the key ideas of the problem formulation to handle a range of configurations. Section 4 culminates in Algorithm 1 where we give our algorithm C-IRIS for growing certified regions. We begin in Section 5 by exploring a simple 3-DOF robot in which we can visualize the configuration space and follow by demonstrating the ability of our algorithm to certify a wide range of postures for a realistic 7-DOF robot. We conclude by showing our algorithm's ability to scale by exploring a 12-DOF, bimanual manipulator.

[3] A problem is said to be APX-hard if no polynomial time algorithm can achieve an approximation ratio of $1 + \delta$ for some $\delta > 0$ unless $P = NP$.

Notation: Throughout the paper, we will use calligraphic letters ($\mathcal{S}$) to denote sets, Roman capitals (X) to denote matrices and Roman lower case (x) to denote vectors. We use $[N] = \{1, \dots, N\}$ and we will denote the cone of Sums-of-Squares (SOS) polynomials as $\boldsymbol{\Sigma}$. Additionally, we will adopt the notation of [21] for rigid transforms.

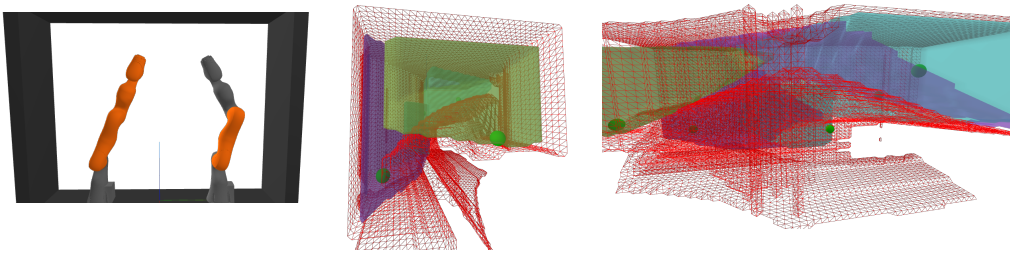

Fig. 2: The 3-DOF robotic system described in Section 5.1. In the right hand panels we visualize the C-space of this robot. The bright red mesh denotes the collision constraint of the robot with itself and with the outer box. The free space is given by the interior of the mesh and is filled by the polytopic regions certified by Algorithm 1 and grown from the seed points in bright green.

2 Problem Formulation

We assume a known environment in which both the robot and obstacles in the *task space* have been decomposed into a union of compact, convex, vertex-representation (V-rep) polytopes. Such polytopic collision geometries of task space are readily available through standard tools (e.g. V-HACD).

Our objective is to find large, convex regions of C-free regardless of the dimension of the configuration space. This objective is beyond the scope of current decomposition methods such as V-HACD due to the complexity of the non-linear kinematics of the robot and the dimensionality of interesting problems

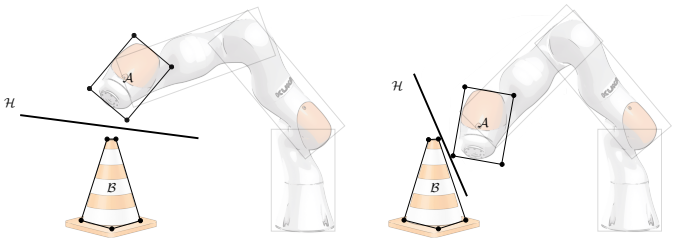

Fig. 3: The convex collision geometries $\mathcal{A}$ and $\mathcal{B}$ are collision-free if and only if there exists a separating plane $\mathcal{H}$. The plane acts as a certificate of non-collision.

We begin by considering the problem of certifying that two collision geometries $\mathcal{A}$ and $\mathcal{B}$ in task space are non-intersecting for a fixed configuration. Recall that two convex bodies are non-intersecting if and only if there exists a plane $\mathcal{H}$ separating the two bodies [22]. Moreover, any point in $\mathcal{A}$ and $\mathcal{B}$ can be written as a convex combination of the vertices of $\mathcal{A}$ and $\mathcal{B}$, respectively. Therefore, a finite certificate that $\mathcal{A}$ and $\mathcal{B}$ are not in collision is to find a plane $a^T x + b = 0$ such that all the vertices $\mathcal{A}_j$ lie on one side of the plane while the vertices $\mathcal{B}_k$ lie on the other side as in Fig 3. Adopting the convention established in [21], we denote the position of vertex $\mathcal{A}_j$ in a given frame F as ${}^F p^{\mathcal{A}_j}$. The non-collision condition is certified by the existence of a separating plane satisfying the following linear constraints on parameters a, b [23]:

$$\begin{aligned} a^T \left({}^F p^{\mathcal{A}_j}\right) + b &\geq 1 \ \forall \ j \\ a^T \left({}^F p^{\mathcal{B}_k}\right) + b &\leq -1 \ \forall \ k. \end{aligned} \tag{1}$$

This hyperplane proves that all the points in body $\mathcal{A}$ are separated from $\mathcal{B}$ by a distance of at least $\frac{2}{\|a\|}$. We emphasize that for each pair of collision geometries we need to search for a separating plane for that pair.

While such a formulation provides a certificate for a single configuration, it is not sufficient for an entire set of possible configurations, as different configurations might require different separating planes as seen in Fig 3. Therefore, we will search for a family of separating planes where $a(q), b(q)$ are parameterized functions of the robot configuration q. We search for these parameters through optimization such that the separating plane condition (1) holds for any configuration within a certain C-space region $\mathcal{P}$, namely

$$\forall q \in \mathcal{P} \implies \begin{cases} a(q)^T \left({}^F p^{\mathcal{A}_j}(q)\right) + b(q) \geq 1 \ \forall \ j \\ a(q)^T \left({}^F p^{\mathcal{B}_k}(q)\right) + b(q) \leq -1 \ \forall \ k \end{cases}, \tag{2}$$

where $\implies$ means the condition on the left implies the condition on the right.

The right hand side of (2) are non-negativity conditions $\left(a^T\right) {}^F p^{\mathcal{A}_j} + b - 1 \geq 0$ and $-1 - \left(a^T\right) {}^F p^{\mathcal{B}_k} - b \geq 0$. It is generally challenging if not intractable to verify a non-negativity condition for the infinitely many element in a set ($\forall q \in \mathcal{P}$ on the left hand side of (2)). But when the non-negativity condition and the set have certain structures, then the verification can become tractable. As we will see in Section 3, when both the set $\mathcal{P}$ and the non-negativity conditions on both side of $\implies$ are specified by polynomials, we can solve a convex optimization program (specifically a Sum-of-Squares program) to verify the condition. In Section 4 we will convert (2) to polynomial functions so as to verify the non-collision condition with the Sum-of-Squares technique.

3 Background

As hinted in Section 2 and as we shall show in Section 4, the problem of certifying a C-free region can be posed as a polynomial implication problem of the form

$$\forall x \in \mathcal{S}_g = \{x : g_i(x) \geq 0, i \in [n]\} \implies p(x) \geq 0, \tag{3}$$

where $g_i(x), p(x)$ are all polynomials of x. While checking the positivity of a polynomial is in general NP-hard [24], a sufficient condition is to check whether $p(x)$ can be expressed as a sum of squares $p(x) = \sum_i f_i^2(x)$, with $f_i(x)$ also being polynomials. Such a check can be performed using semidefinite programming (SDP) [24][25] and is known as Sums-of-Squares (SOS) programming. The SOS technique has been widely used in robotics, for example in stability verification [26][27][28], reachability analysis [29][30] and geometric modelling [31].

Moreover, polynomial implications of the type in (3) are well studied in algebraic geometry [25] and indeed algebraic certificates of the implication can be searched for using SOS programming [24]. Theorems concerning these implications are typically called Positivstellensatz (Psatz) theorems [32]. One of the strongest theorems concerning implications of the form in (3) is known as Putinar's Positivstellensatz:

Theorem 1 (Putinar's Positivstellensatz [33]). *Suppose $\mathcal{S}_g$ is Archimedean. Then $p(x) > 0$ for all $x \in \mathcal{S}_g$ if and only if there exists $\lambda_i(x)$ SOS such that:*

$$p(x) = \lambda_0(x) + \sum_i \lambda_i(x) g_i(x). \tag{C2}$$

The polynomials $\lambda_i(x)$ for $i \geq 0$ are referred to as multiplier polynomials.

where Archimedean is a property slightly stronger than compactness. It is defined in Appendix A.

In order to apply the SOS technique and Theorem 1 to our non-collision condition (2), we need to convert the point positions ${}^F p^{\mathcal{A}_j}, {}^F p^{\mathcal{B}_k}$ to polynomial functions. While the point positions are typically written as trigonometric functions of q using sin and cos, we will build upon a strong history in robotics of algebraic kinematics, e.g. [17][34], to write the kinematics function using polynomials in Section 4.1.

4 Technical Approach

Given an initial, collision-free posture specified as joint angles in the configuration space, our algorithm searches for two key objects: a C-free region $\mathcal{P}$ and a certificate $\mathcal{C}_\mathcal{P}$ that $\mathcal{P}$ contains only collision-free postures. In Section 4.1 we will introduce a rational parameterization of the robot configuration. This parameterization enables us to write the non-collision condition (2) as a polynomial implication problem, certifiable through solving a Sum-of-Squares program (Section 4.2). In Section 4.3, we build on the program from Section 4.2 to enable the search for regions increasing in size. This is outlined in Algorithm 1.

4.1 Rational Parametrization of the Forward Kinematics

In order to write the non-collision condition (2) as a polynomial implication problem (3), we need to convert both sides of $\implies$ in (2) to polynomials. To this

end, we will utilize a rational reparameterization of robot configurations, such that the position of a point on the robot, computed through forward kinematics, is converted to a rational function (with both the numerator and the denominator as polynomials of this reparameterization). This enables us to use Theorem 1 to verify the non-collision condition in Section 4.2 using SOS.

We consider a robot manipulator with N revolute joints[4]. The position of a point A expressed in the reference frame F as a function of the joint angles q generically assumes the form of the trigonometric polynomial

$$^F p_w^A(q) = \sum_j c_{jw} \prod_{i \in \mathcal{S}_{F,A}} \left(\cos^{n_{ij}}(q_i) \sin^{m_{ij}}(q_i)\right), \; w \in \{x, y, z\}, \tag{4}$$

where $\mathcal{S}_{F,A} \subseteq [N]$ is the set of joints lying on the kinematic chain between F and A, $m_{ij} + n_{ij} \leq 1$, and both m_{ij} and $n_{ij} \in \{0, 1\}$ for all i, j [36]. Namely, for each q_i, at most one of $\cos(q_i)$ or $\sin(q_i)$ can appear in $\prod(\cos^{n_{ij}}(q_i) \sin^{m_{ij}}(q_i)$; hence $\cos(q_1)\sin(q_2)\cos(q_3)$ and $\sin(q_1)\cos(q_3)$ are allowed, but $\cos(q_1)\sin(q_1)$ or $\cos^2(q_1)$ are not. c_{jx}, c_{jy}, c_{jz} are given constants computed from the D-H parameters of the kinematics chain. We choose to be explicit about the reference frame F at the risk of being pedantic, as the choice of reference frame F will have important consequences for the scalability of the approach described in Section 4.2 (see Appendix B.1 for a detailed discussion).

The function in (4) is known as a multi-linear trigonometric polynomial. It has many fortunate algebraic properties which we will exploit in subsequent sections. One particular property that we shall use is a change of variables known as the stereographic projection [37]:

$$s_i = \tan\left(\frac{q_i}{2}\right) \rightarrow \begin{cases} \cos(q_i) = \frac{1-s_i^2}{1+s_i^2} \\ \sin(q_i) = \frac{2s_i}{1+s_i^2} \end{cases}. \tag{5}$$

We will refer to the space defined by the variable q as the configuration space and the space defined by the variable s as the tangent-configuration space.

In this new variable s, the forward kinematics are given by the following rational function, where both the numerator and denominator are polynomial functions of s,

$$^F p_w^A(s) = \sum_j c_{jw} \prod_{i \in \mathcal{S}_{F,A}} \left(\frac{(1-s_i^2)^{n_{ij}}(2s_i)^{m_{ij}}}{(1+s_i^2)^{n_{ij}+m_{ij}}}\right), w \in \{x, y, z\}. \tag{6}$$

This parametrization and the theorems in Section 3 will be leveraged in Section 4.2 to transform the certification problem (2) into a SOS program.

Our objective will be to find large, polytopic regions $\mathcal{P} = \{s \mid Cs \leq d\}$ in the variables $s = \tan(q/2)$. We will assume that the joints of the robot cannot undergo complete rotation and are constrained to angles

$$-\pi < q_l \leq q \leq q_u < \pi. \tag{7}$$

[4] Our approach can be extened to robots with algebraic joints, including revolute, prismatic, cylindrical, plane and spherical joints [35].

This ensures that the mapping between the configuration and tangent-configuration spaces is bijective and so trajectories in the tangent-configuration space correspond unambiguously to trajectories in the original configuration space. Moreover, it allows us to restrict the polytope $\mathcal{P} \subseteq \mathcal{P}_{\text{lim}} = \{s \mid s_l \leq s \leq s_u\} = \{s \mid C_{\text{lim}} s \leq d_{\text{lim}}\}$ the polytope encoding the joint limits. This ensures that $\mathcal{P}$ is a compact polytope and hence we can invoke both directions of Theorem 1 due to [38, Corrollary 6.3.5].

4.2 The SOS Certification Problem

We certify the non-collision condition (2) through solving a SOS program which we describe in this section. The key idea is to generalize (1) and search for a polynomial family of separating planes parametrized by the tangent-configuration space variable s which separate two collision geometries $\mathcal{A}$ and $\mathcal{B}$ for all s in a given polyhedron $\mathcal{P}$. These separating planes serve as a certificate that $\mathcal{A}$ and $\mathcal{B}$ are not in collision for any configurations in $\mathcal{P}$. By simultaneously searching for a plane for each collision pair $(\mathcal{A}, \mathcal{B})$ in the environment, we are able to certify that no collision occurs between any two bodies for all $s \in \mathcal{P}$.

To begin, we denote the task-space-polytope vertex position from (6) as a vector of rational functions

$$ {}^{F}p^{\mathcal{A}_j}(s) = \frac{{}^{F}f^{\mathcal{A}_j}(s)}{{}^{F}g^{\mathcal{A}_j}(s)}. $$

Notice that ${}^{F}g^{\mathcal{A}}(s) = \prod_{i \in \mathcal{S}_{F,A}} (1+s_i^2)^{n_{ij}+m_{ij}}$ is a product of positive polynomials and therefore positive for all s. Moreover, this denominator is common for all x/y/z entries of ${}^{F}p^{\mathcal{A}}(s)$. By plugging ${}^{F}p^{\mathcal{A}}(s)$ into the constraints of (1) and multiplying the denominator through, we are able to write a pair of *polynomial* inequalities analogous to the constraints in (1)

$$ \underset{\forall \text{ pairs}(\mathcal{A},\mathcal{B})}{\text{find}} \; a_{\mathcal{A},\mathcal{B}}(s),\; b_{\mathcal{A},\mathcal{B}}(s) \;\text{ subject to} \tag{8a} $$

$$ \underbrace{a_{\mathcal{A},\mathcal{B}}^T(s)\left({}^{F}f^{\mathcal{A}_j}(s)\right) + (b_{\mathcal{A},\mathcal{B}}(s)-1)\left({}^{F}g^{\mathcal{A}_j}(s)\right)}_{\alpha^{F,\mathcal{A}_j}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s)} \geq 0 \;\forall\; j,\; s \in \mathcal{P} \tag{8b} $$

$$ \underbrace{-a_{\mathcal{A},\mathcal{B}}^T(s)\left({}^{F}f^{\mathcal{B}_k}(s)\right) - (b_{\mathcal{A},\mathcal{B}}(s)+1)\left({}^{F}g^{\mathcal{B}_k}(s)\right)}_{\beta^{F,\mathcal{B}_k}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s)} \geq 0 \;\forall\; k,\; s \in \mathcal{P}, \tag{8c} $$

where we denote the polynomials on the left hand side of (8b)(8c) as $\alpha^{F,\mathcal{A}_j}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s)$ and $\beta^{F,\mathcal{B}_k}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s)$ for convenience. $a_{\mathcal{A},\mathcal{B}}(s), b_{\mathcal{A},\mathcal{B}}(s)$ are also polynomials of s. The $\alpha^{F,\mathcal{A}_j}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s)$ and $\beta^{F,\mathcal{B}_k}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s)$ are linear in the coefficients of the polynomials $a_{\mathcal{A},\mathcal{B}}$ and $b_{\mathcal{A},\mathcal{B}}$ which are the decision variables in the problem. The variables s are known as the indeterminates of the polynomial.

At this point, we recognize that the non-collision condition (2) can be written as the following polynomial implication:

$$ s \in \mathcal{P} = \{s \mid Cs \leq d\} \implies \text{(8b) and (8c)}. $$

Using Theorem 1 and [38, Corrollary 6.3.5], a necessary and sufficient condition for this implication to hold is the existence of polynomials $\lambda_i^{\mathcal{A}_j,\mathcal{B}}(s), \nu_i^{\mathcal{A},\mathcal{B}_k}(s) \in \boldsymbol{\Sigma}$ (where $\boldsymbol{\Sigma}$ is the set of SOS polynomials) such that

$$\alpha^{F,\mathcal{A}_j}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s) = \lambda_0^{\mathcal{A}_j,\mathcal{B}}(s) + \sum_{i=1}^{m} \lambda_i^{\mathcal{A}_j,\mathcal{B}}(s)(d_i - c_i^T s) \tag{9a}$$

$$\beta^{F,\mathcal{B}_k}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s) = \nu_0^{\mathcal{A},\mathcal{B}_k}(s) + \sum_{i=1}^{m} \nu_i^{\mathcal{A},\mathcal{B}_k}(s)(d_i - c_i^T s), \tag{9b}$$

where c_i, d_i are the i-th row of C, d. The equality between the polynomials in (9a)(9b) means that the coefficients of the corresponding terms are equal, which adds linear equality constraints on the decision variables $a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, \lambda^{\mathcal{A}_j,\mathcal{B}}, \mu^{\mathcal{A},\mathcal{B}_k}$.

Therefore, a polytopic region in the tangent-configuration space $\mathcal{P} = \{s \mid Cs \leq d\}$ is collision-free if and only if the program

$$\begin{aligned}
&\underset{\forall \text{ pairs}(\mathcal{A},\mathcal{B})}{\text{find}} \; a_{\mathcal{A},\mathcal{B}}(s),\ b_{\mathcal{A},\mathcal{B}}(s),\ \lambda(s), \nu(s) \text{ subject to} \\
&\alpha^{F,\mathcal{A}_j}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s) = \lambda_0^{\mathcal{A}_j,\mathcal{B}}(s) + \sum_{i=1}^{m} \lambda_i^{\mathcal{A}_j,\mathcal{B}}(s)(d_i - c_i^T s) \\
&\beta^{F,\mathcal{B}_k}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s) = \nu_0^{\mathcal{A},\mathcal{B}_k}(s) + \sum_{i=1}^{m} \nu_i^{\mathcal{A},\mathcal{B}_k}(s)(d_i - c_i^T s) \\
&\lambda(s), \nu(s) \in \boldsymbol{\Sigma}
\end{aligned} \tag{10}$$

is feasible, where $\lambda(s)$ and $\nu(s)$ collect all the multipliers $\lambda_i^{\mathcal{A}_j,\mathcal{B}}$ and $\nu_i^{\mathcal{A},\mathcal{B}_k}$. This is a SOS program which searches for the coefficients of the polynomials $a_{\mathcal{A},\mathcal{B}}(s),\ b_{\mathcal{A},\mathcal{B}}(s),\ \lambda(s)$, and $\nu(s)$. We call a feasible solution

$$\mathcal{C}_{\mathcal{P}} = \bigcup_{\mathcal{A},\mathcal{B}} \{a_{\mathcal{A},\mathcal{B}}(s),\ b_{\mathcal{A},\mathcal{B}}(s),\ \lambda(s), \nu(s)\}$$

to (10) a collision-free certificate of $\mathcal{P}$.

Remark 1. If $\mathcal{P}$ is collision-free, then it can always be certified as such by (10) provided the degree of α and β are sufficiently large. We discuss practical considerations of basis selection for (10) in Appendix B.2

4.3 Growing Polytopic Regions in Tangent-Configuration Space

Describing C-free using a few large regions rather than many smaller ones is advantageous in almost all circumstances. In this section, we show how to extend the SOS feasibility program in (10) to enable not only certification, but certified *growth* of polytopic C-free regions.

We begin by discussing how we will measure the size of our polytope $\mathcal{P} = \{s \mid Cs \leq d\}$. While it may be attractive to measure the size of a polytope by its volume, it is known that computing the volume of a half-space representation (H-Rep) polytope is #P-hard[5] [40] and therefore intractable as an objective. A

[5] #P-hard problems are at least as hard as NP-complete problems [39]

useful surrogate for the volume of $\mathcal{P}$ used in [14] is the volume of the maximal inscribed ellipse of $\mathcal{P}$, the set $\mathcal{E}_\mathcal{P} = \{Qu + s_0 \mid \|u\|_2 \leq 1\}$ where Q describes the shape of the ellipsoid and s_0 its center. The problem of finding the maximal inscribed ellipsoid in a given polytope is a semidefinite program described in [22, Section 8.4.2].

Additionally, in order to cover diverse areas of C-free, we grow each polytope $\mathcal{P}$ around some nominal posture s_s we call the seed point. New seed points are typically chosen using rejection sampling to obtain a point outside of the existing certified regions and $\mathcal{P}$ is required to contain s_s as it grows.

Combining the objective, the constraints of the maximal inscribed ellipsoid program, and the seed point condition with the program in (10) yields the optimization program:

$$\max_{Q,s_0,C,d,\lambda,\nu,a_{\mathcal{A},\mathcal{B}},b_{\mathcal{A},\mathcal{B}}} \mathbf{logdet} Q \text{ subject to} \tag{11a}$$

$$\|Qc_i\|_2 \leq d_i - c_i^T s_0 \ \forall\ i \in [m] \tag{11b}$$

$$Cs_s \leq d \tag{11c}$$

$$\|c_i\|_2 \leq 1 \ \forall\ i \in [m] \tag{11d}$$

$$\alpha^{F,\mathcal{A}_j}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s) = \lambda_0^{\mathcal{A}_j,\mathcal{B}}(s) + \sum_{i=1}^{m} \lambda_i^{\mathcal{A}_j,\mathcal{B}}(s)(d_i - c_i^T s) \ \forall \text{ pairs } (\mathcal{A},\mathcal{B}) \tag{11e}$$

$$\beta^{F,\mathcal{B}_k}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s) = \nu_0^{\mathcal{A},\mathcal{B}_k}(s) + \sum_{i=1}^{m} \nu_i^{\mathcal{A},\mathcal{B}_k}(s)(d_i - c_i^T s) \ \forall \text{ pairs } (\mathcal{A},\mathcal{B}) \tag{11f}$$

$$Q \succeq 0,\ \lambda(s), \nu(s) \in \boldsymbol{\Sigma}. \tag{11g}$$

The condition $\mathcal{E}_\mathcal{P} \subset \mathcal{P}$ is given by the constraint (11b). (11c) enforces that $\mathcal{P}$ grows around s_s. (11e) and (11f) are the polynomial implication condition (8b) (8c): $s \in \mathcal{P} = \{s \mid Cs \leq d\} \implies \alpha \geq 0$ and $\beta \geq 0$, proving the existence of the separating planes between the robot and the obstacles in the task space. The added constraint (11d) prevents numerically undesirable scaling of our polytope. While this program is attractive as a specification, it is not convex due to the bilinearity terms Qc_i in (11b) and $\lambda_i d_i, \lambda_i c_i^T$ in (11e)(11f). This bilinearity precludes simultaneous search of $\mathcal{P}, \mathcal{E}_\mathcal{P}$ and the corresponding certificate.

However, the program *is* convex when either the polytope $\mathcal{P}$ is fixed or when the inscribed ellipse $\mathcal{E}_\mathcal{P}$ and the multiplier polynomials (save for λ_0 and ν_0) are fixed. In the former case, given a polytope such that (11) is feasible, we simultaneously obtain a collision-free certificate $\mathcal{C}_\mathcal{P}$, maximal inscribed ellipsoid $\mathcal{E}_\mathcal{P}$, and an accompanying estimate of the size of $\mathcal{P}$. In this situation, clearly (11d) becomes redundant. We remark that (11) is only marginally more complex than (10) as it introduces only one additional semidefinite decision variable Q and one vector decision variable s_0 with m constraints which are not coupled to the multiplier polynomials.

Conversely, from a solution of (11) we are also capable of searching for a new collision-free polytope with an accompanying certificate. To ensure growth of the polytopic region, our objective is to push the faces of the current polytope

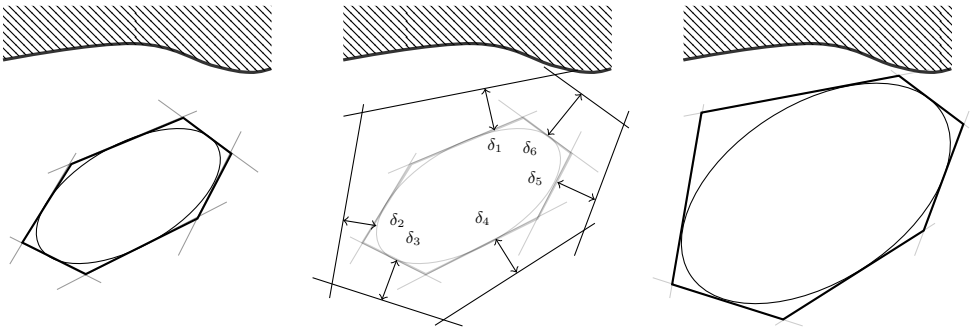

Fig. 4: In (12) we search for the maximum amount the polytopes faces can be pushed away from the current inscribed ellipse without violating the certificate found in the previous step.

away from the inscribed ellipsoid without violating our certificate as in Fig 4. To accomplish this, we present the following minor modification of (11)

$$\begin{aligned} \max_{C,d,\delta,\lambda_0,\nu_0,a_{\mathcal{A},\mathcal{B}},b_{\mathcal{A},\mathcal{B}}} \quad & \prod_{i=1}^{m}(\delta_i+\varepsilon_0) \text{ subject to} \\ \|Qc_i\|_2 \leq d_i - \delta_i - c_i^T s_0, \;\; & \delta_i \geq 0 \;\forall\; i \in [m] \\ & (11c), (11d), (11e), (11f) \\ & \lambda_0(s), \nu_0(s) \in \boldsymbol{\Sigma} \end{aligned} \tag{12}$$

where $\varepsilon_0 > 0$ is some positive constant ensuring that the objective is never 0. We emphasize that we have fixed the multipliers λ_i and ν_i for $i > 0$ and instead search for d_i and c_i.

In this program, we introduce the variable δ_i which measures the distance between the hyperplane $c_i^T s = d_i$ and the inscribed ellipse. Searching over the variable C allows the normal of the polytope to rotate in order to accomplish a greater change in margin. The remaining constraints enforce that the polytope continue to contain the ellipse $\mathcal{E}_\mathcal{P}$ while also generating a new set of separating planes within the collision-free certificate $\mathcal{C}_\mathcal{P}$. The alternation procedure is outlined in Algorithm 1 and can be seen as a generalization of the Iris algorithm from [14]. Notice that the volume of the inscribed ellipsoid $\mathcal{E}_\mathcal{P}$ is guaranteed to increase in each iteration of Algorithm 1.

Remark 2. To make Algorithm 1 numerically tractable, in Appendix B, we discuss practical aspects to scale the algorithm to realistic examples.

5 Results

In this section, we demonstrate the use of Algorithm 1 on three representative examples. In the first example, we consider a 3-DOF system for which we can

Algorithm 1: Given an initial polytopic region $\mathcal{P}_0$ for which (11) is feasible, return a new polytopic region $\mathcal{P}_i$ with a maximal inscribed ellipse $\mathcal{E}_{\mathcal{P}_i}$ with larger volume than $\mathcal{E}_{\mathcal{P}_0}$ and a collision-free certificate $\mathcal{C}_{\mathcal{P}_i}$. Notice that since $\mathcal{P}_0$ is assumed collision-free, programs (11) and (12) are always feasible. (11) and (12) are alternated until the change in the volume of the ellipse $\mathcal{E}_{\mathcal{P}}$ is less than some *tolerance*.

$i \leftarrow 0$
do
  $(\mathcal{E}_{\mathcal{P}_i}, \mathcal{C}_{\mathcal{P}_i}) \leftarrow$ Solution of (11) with data $(\mathcal{P}_i)$
  $(\mathcal{P}_{i+1}, \mathcal{C}_{\mathcal{P}_{i+1}}) \leftarrow$ Solution of (12) with data $(\mathcal{E}_{\mathcal{P}_i}, \mathcal{C}_{\mathcal{P}_i})$
  $i \leftarrow i + 1$
while $\left(\boldsymbol{vol}(\mathcal{E}_{\mathcal{P}_i}) - \boldsymbol{vol}(\mathcal{E}_{\mathcal{P}_{i-1}})\right) / \boldsymbol{vol}(\mathcal{E}_{\mathcal{P}_{i-1}}) \geq$ *tolerance*;
return $(\mathcal{P}_i, \mathcal{C}_{\mathcal{P}_i})$

visualize the entire configuration space. This enables us to plot the resulting regions of our algorithm. Next, we analyze the typical run times on the 7-DOF KUKA iiwa. Finally, we demonstrate the scalability of our algorithm on a 12-DOF system composed of two KUKA iiwas (with the wrist joint fixed as the wrist link is symmetric about the joint axis). All convex programs are solved using Mosek v9.2 [41] running on Ubuntu 20.04 with an Intel 11th generation i9 processor. Code is publicly available on Github[6].

5.1 3-DOF Flipper System

In this example, we consider the 3-DOF system in the left-most panel of Fig 2 consisting of two iiwas with all joints save those at the ends of the orange links fused. This system has few degrees of freedom, but maintains rich, realistic collision geometries. As the C-space is three-dimensional, we are able to visualize the collision constraint by using marching cubes [42]. This is plotted as the red mesh in the right two panels of Fig 2.

We run Algorithm 1 on 11 different regions initialized according to Algorithm 2 described in Appendix B.4. Our approach is able to grow regions whose union describes 90% of C-free. In Fig 2, we visualize a subset of these 11 regions demonstrating that each region covers a substantial amount of the C-free on their own. In step 3 of Algorithm 1, the largest SOS takes 0.008s to solve, and in step 4 of Algorithm 1, the SOS takes 0.647s to solve.

In Fig 5, we demonstrate the certification of a single, particularly large region. The green and teal plane certify that the end-effector of the left arm (highlighted in blue) does not collide with the middle link (highlighted in purple) and the top link (highlighted in red) respectively for all configurations in the purple region shown in right hand panes of 5.

[6] https://github.com/AlexandreAmice/drake/tree/C_Iris

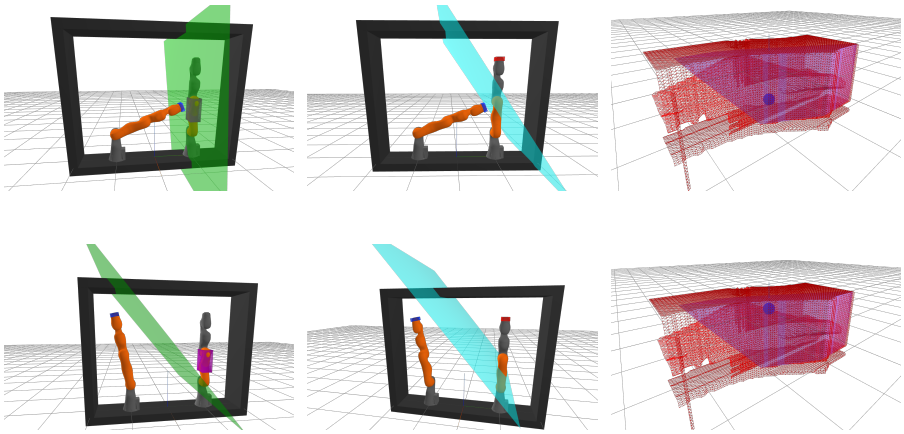

Fig. 5: Algorithm 1 generates a set of separating planes between each collision body that vary as the configuration moves (denoted by the blue sphere in the right most pane). As the configuration varies in the purple C-free region, the separating planes in green and teal can move to accommodate the relative shifts in the robot positions.

5.2 7-DOF KUKA with Shelf

We apply our approach to a 7-DOF KUKA iiwa arm to certify and search for C-free regions with a shelf, as shown in Fig 6. We also show the growth of volume for one certified C-free region in Fig 7. The volume increases by a factor of 10,000 in 11 iterations of Algorithm 1, and covers a range of configurations (Fig 7 left). In step 3 of Algorithm 1, the largest SOS take 54s to solve, and in step 4 of Algorithm 1, the SOS takes 8s to solve.

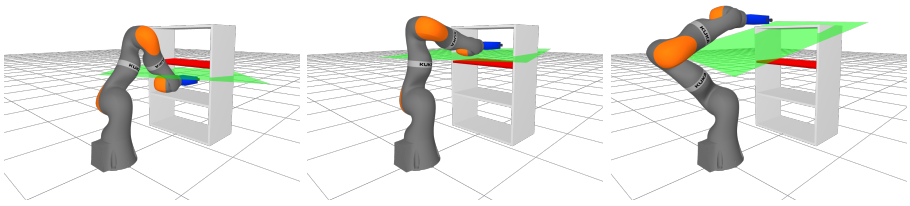

Fig. 6: 7-DOF iiwa example. We highlight one pair of collision geometries (blue on robot gripper and red on the shelf), together with their separating plane (green).

5.3 12-DOF Bimanual Example

Finally we demonstrate the scalability of our approach on a 12-DOF system of two KUKA iiwa arms (with fixed wrist joint) to find C-free regions avoiding self-

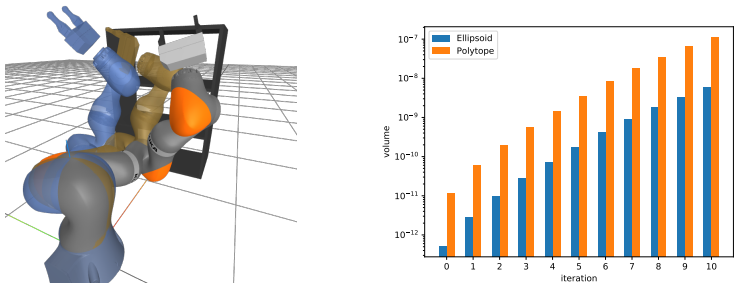

Fig. 7: Left: multiple sampled postures (in different colors) in a certified C-free region. Right: The growth of volume with Algorithm 1 in each iteration.

collision. In Fig 1, we visualize several postures in one certified C-free region. At one sampled posture, the closest distance between the collision geometries is 7.3mm (See Fig 1 right), indicating our certified C-free region is extremely tight. 12-DOF is quite high-dimensional for any sampling-based algorithm, and the solution times for generating our strong deterministic certificate increases significantly, too, compared to the single arm case. Solving the largest SOS program in line 3 of Algorithm 1 takes 105 minutes, and the SOS program in line 4 of Algorithm 1 take 4 minutes.

6 Conclusion and Future Work

In this work, we present an approach to find large certifiable C-free regions for robot manipulators. Our approach certifies a polytopic region in the tangent-configuration space to be collision-free through convex optimization. Moreover, we give a practical, iterative algorithm for finding and growing these C-free regions. Our method works in arbitrary dimensions and scales to reasonable, realistic examples in robotic manipulation. Such certified regions find practical use in both randomized and optimization-based collision-free motion planning algorithms.

While this paper assumed a system composed of only revolute joints and collision bodies defined as V-rep polytopes, we believe that the essential machinery can be applied to arbitrary algebraic joints and convex bodies. We intend to investigate such extensions in future work. Additionally, in Appendix C we have also given a non-linear optimization algorithm for very rapidly proposing regions when the cost of many alternations becomes prohibitive. Future work will explore a tighter integration of Algorithms 1 and 2.

References

1. T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computers*, vol. 100, no. 32, 1983.

2. L. E. Kavraki, "Computation of configuration-space obstacles using the fast fourier transform," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3.
3. M. Branicky and W. Newman, "Rapid computation of configuration space obstacles," in *Proceedings., IEEE International Conference on Robotics and Automation*, 1990.
4. J. Canny, *The complexity of robot motion planning*. MIT press, 1988.
5. S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.
6. L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
7. M. Verghese, N. Das, Y. Zhi, and M. Yip, "Configuration space decomposition for scalable proxy collision checking in robot planning and control," *arXiv preprint arXiv:2201.04314*, 2022.
8. Y. Han, W. Zhao, J. Pan, Z. Ye, R. Yi, and Y.-J. Liu, "A configuration-space decomposition scheme for learning-based collision checking," *arXiv preprint arXiv:1911.08581*, 2019.
9. T. H. Wong, G. Leach, and F. Zambetta, "An adaptive octree grid for gpu-based collision detection of deformable objects," *The Visual Computer*, vol. 30, no. 6.
10. A. Lingas, "The power of non-rectilinear holes," in *International Colloquium on Automata, Languages, and Programming*.
11. S. J. Eidenbenz and P. Widmayer, "An approximation algorithm for minimum convex cover with logarithmic performance guarantee," *SIAM Journal on Computing*, vol. 32, no. 3.
12. J.-M. Lien and N. M. Amato, "Approximate convex decomposition of polyhedra," in *Proceedings of the 2007 ACM symposium on Solid and physical modeling*.
13. M. Ghosh, N. M. Amato, Y. Lu, and J.-M. Lien, "Fast approximate convex decomposition using relative concavity," *Computer-Aided Design*, vol. 45, no. 2.
14. 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.
15. R. Diankov, "Automated construction of robotic manipulation programs," 2010.
16. P. Trutman, S. E. D. Mohab, D. Henrion, and T. Pajdla, "Globally optimal solution to inverse kinematics of 7dof serial manipulator," *arXiv preprint arXiv:2007.12550*, 2020.
17. A. J. Sommese, C. W. Wampler *et al.*, *The Numerical solution of systems of polynomials arising in engineering and science*. World Scientific, 2005.
18. R. Deits and R. Tedrake, "Efficient mixed-integer planning for uavs in cluttered environments," in *2015 IEEE international conference on robotics and automation (ICRA)*.
19. T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *2001 European control conference (ECC)*.
20. T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," 2022. [Online]. Available: https://arxiv.org/abs/2205.04422
21. R. Tedrake, *Robotic Manipulation*, 2021. [Online]. Available: https://manipulation.mit.edu/pick.html#monogram
22. S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
23. C. M. Bishop, "Pattern recognition," *Machine learning*, vol. 128, no. 9, 2006.

24. P. A. Parrilo, *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.
25. G. Blekherman, P. A. Parrilo, and R. R. Thomas, *Semidefinite optimization and convex algebraic geometry*. SIAM, 2012.
26. R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.
27. A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.
28. S. Shen and R. Tedrake, "Sampling quotient-ring sum-of-squares programs for scalable verification of nonlinear systems," in *2020 59th IEEE Conference on Decision and Control (CDC)*.
29. Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. Packard, "Some controls applications of sum of squares programming," in *42nd IEEE international conference on decision and control (IEEE Cat. No. 03CH37475)*, vol. 5. IEEE, 2003, pp. 4676–4681.
30. H. Yin, M. Arcak, A. Packard, and P. Seiler, "Backward reachability for polynomial systems on a finite horizon," *IEEE Transactions on Automatic Control*, vol. 66, no. 12, pp. 6025–6032, 2021.
31. A. A. Ahmadi, G. Hall, A. Makadia, and V. Sindhwani, "Geometry of 3d environments and sum of squares polynomials," *arXiv preprint arXiv:1611.07369*, 2016.
32. G. Stengle, "A nullstellensatz and a positivstellensatz in semialgebraic geometry," *Mathematische Annalen*, vol. 207, no. 2.
33. M. Putinar, "Positive polynomials on compact semi-algebraic sets," *Indiana University Mathematics Journal*, vol. 42, no. 3.
34. C. W. Wampler, A. Morgan, and A. Sommese, "Numerical continuation methods for solving polynomial systems arising in kinematics," 1990.
35. C. W. Wampler and A. J. Sommese, "Numerical algebraic geometry and algebraic kinematics," *Acta Numerica*, vol. 20.
36. J. J. Craig, *Introduction to robotics: mechanics and control*. Pearson Educacion, 2005.
37. M. Spivak, *Calculus Third Edition*. Cambridge University Press, 1994.
38. A. Prestel and C. Delzell, *Positive polynomials: from Hilbert's 17th problem to real algebra*. Springer Science & Business Media, 2013.
39. J. S. Provan and M. O. Ball, "The complexity of counting cuts and of computing the probability that a graph is connected," *SIAM Journal on Computing*, vol. 12, no. 4.
40. 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.
41. M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019. [Online]. Available: http://docs.mosek.com/9.0/toolbox/index.html
42. W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM siggraph computer graphics*, vol. 21, no. 4.

A Definition of Archimedean

In this section we formally define the Archimedean property that appears in Theorem 1.

Definition 1. *A semialgebraic set* $\mathcal{S}_g = \{x \mid g_i(x) \geq 0, i \in [n]\}$ *is Archimedean if there exists* $N \in \mathbb{N}$ *and* $\lambda_i(x) \in \boldsymbol{\Sigma}$ *such that:*

$$N - \sum_{i=1}^{n} x_i^2 = \lambda_0(x) + \sum_{i=1}^{n} \lambda_i(x) g_i(x)$$

B Practical Aspects

In this section we discuss important ways to scale Algorithm 1. In sections B.1 and B.2 we describe design choices which dramatically reduce the size of the SOS programs used in the alternations. Next, we discuss aspects of Algorithm 1 which can be parallelized, reducing solve time in the alternations. Finally, we describe an extension to the original Iris algorithm [1] which can be used to rapidly propose a very large region that is likely, but not certified, to be collision-free. Seeding Algorithm 1 with such a region can dramatically reduce the number of iterations required to obtain a satisfyingly large volume.

B.1 Choosing the Reference Frame

The polynomial implications upon which the programs (11) and (12) are based require choosing a coordinate frame between each collision pair $\mathcal{A}$ and $\mathcal{B}$. However, as the collision-free certificate between two different collision pairs can be computed independently of each other, we are free to choose a different coordinate frame to express the kinematics for each collision pair. This is important in light of (4) and (5) that indicate that the degree of the polynomials ${}^{F}f^{\mathcal{A}_j}$ and ${}^{F}g^{\mathcal{A}_j}$ are equal to two times the number of joints lying on the kinematic chain between frame F and the frame for $\mathcal{A}$. For example, the tangent-configuration space polynomial in the variable s describing the position of the end-effector of a 7-DOF robot is of total degree 14 when written in the coordinate frame of the robot base. However, when written in the frame of the third link, the polynomial describing the position of the end effector is only of total degree $(7-3) \times 2 = 8$. This observation is also used in [2] to reduce the size of the optimization program.

The size of the semidefinite variables in (11) and (12) scale as the square of the degree of the polynomial used to express the forward kinematics. Supposing there are n links in the kinematics chain between $\mathcal{A}$ and $\mathcal{B}$, then choosing the jth link along the kinematics chain as the reference frame F leads to scaling of order $j^2 + (n-j)^2$. Choosing the reference frame in the middle of the chain minimizes this complexity to scaling of order $\frac{n^2}{2}$ and we therefore adopt this convention in our experiments.

B.2 Basis Selection

The condition that a polynomial can be written as a sum of squares can be equivalently formulated as an equality constraint between the coefficients of the polynomial and an associated semidefinite variable known as the Gram matrix [3]. In general, a polynomial in k variables of total degree $2d$ has $\binom{k+2d}{2d}$ coefficients and requires a Gram matrix of size $\binom{k+d}{d}$ to represent which can quickly become prohibitively large. Fortunately, the polynomials in our programs contain substantially more structure which will allow us to drastically reduce the size of the Gram matrices.

We begin by noting from (6) that while both the numerator and denominator of the forward kinematics are of total degree $2n$, with n the number of links of the kinematics chain between frame A and F, both polynomials are of *coordinate* degree of at most two (i.e. the highest degree of s_i in any term is s_i^2). We will refer to this basis as $\mu(s)$ which is a vector containing terms of the form $\prod_{i=1}^{n} s_i^{d_i}$ with $d_i \in \{0, 1, 2\}$ for all n^3 possible permutations of the exponents d_i.

Next, we recall the form of $\alpha^{F,\mathcal{A}_j}(a_{\mathcal{A},\mathcal{B}}, b_{\mathcal{A},\mathcal{B}}, s)$ from (8b) and (8c). If $a_{\mathcal{A},\mathcal{B}}(s) = a_{\mathcal{A},\mathcal{B}}^T \eta(s)$ and $b_{\mathcal{A},\mathcal{B}}(s) = b_{\mathcal{A},\mathcal{B}}^T \eta(s)$ for some basis η in the variable s, then $\alpha^{F,\mathcal{A}_j}$ and $\beta^{F,\mathcal{A}_j}$ can be expressed as linear functions of the basis $\gamma(s) = \mathbf{vect}(\eta(s)\mu(s)^T)$ where we use **vect** to indicate the flattening of the matrix outer product. Concretely, if we choose to make $a_{\mathcal{A},\mathcal{B}}(s)$ and $b_{\mathcal{A},\mathcal{B}}(s)$ linear functions of the indeterminates s, then $\eta(s) = l(s) = \begin{bmatrix}1 & s_1 & \dots & s_n\end{bmatrix}$. Therefore $\alpha^{F,\mathcal{A}_j}$ and $\beta^{F,\mathcal{A}_j}$ can be expressed as linear functions of the basis

$$\gamma(s) = \begin{bmatrix}\mu(s) & s_1\mu(s) & \dots & s_n\mu(s)\end{bmatrix} \tag{13}$$

After choosing the basis $\eta(s)$, which determines the basis $\gamma(s)$, the equality constraints (9a) and (9b) constrain the necessary basis needed to express the multiplier polynomials $\lambda(s)$ and $\nu(s)$. The minimal such basis is related to an object known in computational algebra as the Newton polytope of a polynomial $\mathbf{New}(f)$ [4]. The exact condition is that the $\mathbf{New}(\eta(s)) + \mathbf{New}(\mu(s)) \subseteq \mathbf{New}(\rho(s)) + \mathbf{New}(l(s))$ where the sum in this case is the Minkowski sum.

If we choose $\eta(s)$ as the linear basis $l(s)$, then we obtain the condition that $\mathbf{New}(\rho(s)) = \mathbf{New}(\mu(s))$ and since $\mu(s)$ is a dense, even degree basis then we must take $\rho(s) = \mu(s)$. Choosing $\eta(s)$ as the constant basis would in fact result in the same condition, and therefore searching for separating planes which are linear functions of the tangent-configuration space does not increase the size of the semidefinite variables. As the complexity of (11) and (12) is dominated by the size of these semidefinite variables, separating planes which are linear functions changes does not substantially affect the solve time but can dramatically increase the size of the regions which we can certify.

Remark 3. In the case of certifying that the end-effector of a 7-DOF robot will not collide with the base using linearly parametrized hyperplanes, choosing to express conditions (9a) and (9b) in the world frame with naïvely chosen bases would result in semidefinite variables of size $\binom{7+7}{7} = 3432$. Choosing to express the conditions according to the discussion in Section B.1 and choosing the basis $\gamma(s)$ from (13) results in semidefinite matrices of rows at most $2^4 = 16$.

B.3 Parallelization

While it is attractive from a theoretical standpoint to write (11) as a single, large program it is worth noting that can in fact be viewed as $K+1$ individual SOS and SDP programs, where K is the number of collision pairs in the environment. Indeed, certifying whether pairs $(\mathcal{A}_1, \mathcal{A}_2)$ are collision-free for all s in the polytope $\mathcal{P}$ can be done completely independently of the certification of another pair $(\mathcal{A}_1, \mathcal{A}_3)$ as neither the constraints nor the cost couple the conditions of imposed on any pairs. Similarly, the search for the largest inscribed ellipsoid can be done independently of the search for the separating hyperplanes.

Solving the certification problem embedded in (11) as K individual SOS programs has several advantages. First, as written (11) has $2(m+1)K\sum_i |\mathcal{A}_i|$ semidefinite variables of various sizes. In the example from Section 5.1 this corresponds to 35,072 semidefinite variables. This can be prohibitively large to store in memory as a single program. Additionally, solving the problems independently enables us to determine which collision bodies cannot be certified as collision-free and allows us to terminate our search as soon as a single pair cannot be certified. Finally, decomposing the problems into subproblems enables us to increase computation speed by leveraging parallel processing.

We note that (12) cannot be similarly decomposed as on this step the variables c_i^T and d affect all of the constraints. However, this program is substantially smaller as we have fixed $2mK\sum_i |\mathcal{A}_i|$ semidefinite variables as constants and replaced them with $2m$ linear variables representing the polytope. This program is much more amenable to being solved as a single program.

B.4 Seeding the Algorithm

It is worth noting that the alternations in Algorithm 1 must be initialized with a polytope $\mathcal{P}_0$ for which (11) is feasible. In principle, the alternation proposed in Section 4.3 can be seeded with an arbitrarily small polytope around a collision-free seed point. This seed polytope is then allowed to grow using Algorithm 1. However, this may require running several dozens of iterations of Algorithm 1 for each seed point which can become prohibitive as the size of the problem grows. It is therefore advantageous to seed with as large a region as can be initially certified.

Here we discuss an extension of the IRIS algorithm in [1] which uses nonlinear optimization to rapidly generate large regions in C-space. These regions are not guaranteed to be collision-free and therefore they must still be passed to Algorithm 1 to be certified, but do provide good initial guesses. In this section, we will assume that the reader is familiar with IRIS and will only discuss the modification required to use it to grow C-space regions. Detailed pseudocode is available in Appendix C

IRIS grows regions in a given space by alternating between two subproblems: SEPARATINGHYPERPLANES and INSCRIBEDELLIPSOID. The INSCRIBEDELLIPSOID is exactly the program described in [5, Section 8.4.2] and we do not need

to modify it. The subproblem SeparatingHyperplanes finds a set of hyperplanes which separate the ellipse generated by InscribedEllipsoid from all of the obstacles. This subproblem is solved by calling two subroutines ClosestPointOnObstacle and TangentPlane. The former finds the closest point on a given obstacle to the ellipse, while the latter places a plane at the point found in ClosestPointOnObstacle that is tangent to the ellipsoid.

The original work in [1] assumes convex obstacles which enables ClosestPointOnObstacle to be solved as a convex program and for the output of TangentPlane to be globally separating plane between the obstacle and the ellipsoid of the previous step. Due to the non-convexity of the C-space obstacles in our problem formulation, finding the closest point on an obstacle exactly becomes a computationally difficult problem to solve exactly [6]. Additionally, placing a tangent plane at the nearest point will be only a locally separating plane, not a globally separating one.

To address the former difficulty, we formulate ClosestPointOnObstacle as a nonlinear program. Let the current ellipse be given as $\mathcal{E} = \{Qu + s_0 \mid \|u\|_2 \leq 1\}$ and suppose we have the constraint that $s \in \mathcal{P} = \{s \mid Cs \leq d\}$. Let $\mathcal{A}$ and $\mathcal{B}$ be two collision pairs and ${}^{\mathcal{A}}p_{\mathcal{A}}, {}^{\mathcal{B}}p_{\mathcal{B}}$ be some point in bodies $\mathcal{A}$ and $\mathcal{B}$ expressed in some frame attached to $\mathcal{A}$ and $\mathcal{B}$. Also, let ${}^{W}X^{\mathcal{A}}(s)$ and ${}^{W}X^{\mathcal{B}}(s)$ denote the rigid transforms from the reference frames $\mathcal{A}$ and $\mathcal{B}$ to the world frame respectively. We remind the reader that this notation is drawn from [7]. The closest point on the obstacle subject to being contained in $\mathcal{P}$ can be found by solving the program

$$\min_{s, {}^{\mathcal{A}}p_{\mathcal{A}}, {}^{\mathcal{B}}p_{\mathcal{B}}} (s - s_0)^T Q^T Q (s - s_0) \text{ subject to} \tag{14a}$$

$${}^{W}X^{\mathcal{A}}(s) {}^{\mathcal{A}}p_{\mathcal{A}} = {}^{W}X^{\mathcal{B}}(s) {}^{\mathcal{B}}p_{\mathcal{B}} \tag{14b}$$

$$Cs \leq d \tag{14c}$$

This program searches for the nearest configuration in the metric of the ellipse such that two points in the collision pair come into contact. We find a locally optimal solution $(s^\star, {}^{\mathcal{A}}p^\star_{\mathcal{A}}, {}^{\mathcal{B}}p^\star_{\mathcal{B}})$ to the program using a fast, general-purpose nonlinear solver such as SNOPT [8]. The tangent plane to the ellipse $\mathcal{E}$ at the point $s^\star$ is computed by calling TangentPlane then appended to the inequalities of $\mathcal{P}$ to form $\mathcal{P}'$. This routine is looped until (14) is infeasible at which point InscribedEllipse is called again.

Once a region $\mathcal{P} = \{s \mid Cs \leq d\}$ is found by Algorithm 2, it will typically contain some minor violations of the non-collision constraint. To find an initial, feasible polytope $\mathcal{P}_0$ to use in Algorithm 1, we search for a minimal uniform contraction δ of $\mathcal{P}$ such that $\mathcal{P}_\delta = \{s \mid Cs \leq d - \delta * 1\}$ is collision-free. This can be found by bisecting over the variable $\delta \in [0, \delta_{\max}]$ and solving repeated instances of (11).

Seeding Algorithm 1 with a $\mathcal{P}_0$ as above can dramatically reduce the number of alternations required to obtain a fairly large region and is frequently faster than seeding Algorithm 1 with an arbitrarily small polytope.

C Supplementary Algorithms

We present a pseudocode for the algorithm presented in Appendix B.4. A mature implementation of this algorithm can be found in Drake[7].

Algorithm 2: Given an initial tangent-configuration space point s_0 and a list of obstacles $\mathcal{O}$, return a polytopic region $\mathcal{P} = \{s \mid Cs \leq d\}$ and inscribed ellipsoid $\mathcal{E}_\mathcal{P} = \{s \mid Qu + s_c\}$ which contains a substantial portion of the free C-space (but is not guaranteed to contain no collisions)

$(C, d) \leftarrow$ plant joint limits
$\mathcal{P}_i \leftarrow \{s \mid Cs \leq d\}$
$\mathcal{E}_{\mathcal{P}_0} \leftarrow$ InscribedEllipsoid$(\mathcal{P}_0)$
$j \leftarrow$ number of rows of C
do
  do
    $(s^\star, {}^\mathcal{A}p^\star_\mathcal{A}, {}^\mathcal{B}p^\star_\mathcal{B}) \leftarrow$ FindClosestCollision$(\mathcal{P}_i, \mathcal{E}_{\mathcal{P}_i})$
    $(c^T_{j+1}, d_{j+1}) \leftarrow$ TangentHyperplane$(s^\star, \mathcal{E}_{\mathcal{P}_i})$
    $C \leftarrow$ **hstack**(C, c^T_{j+1})
    $d \leftarrow$ **hstack**(d, d_{j+1})
    $\mathcal{P}_i \leftarrow \{s \mid Cs \leq d\}$
    $j \leftarrow j + 1$
  while FindClosestCollision$(\mathcal{P}_i, \mathcal{E}_{\mathcal{P}_i})$ *is feasible*;
  $\mathcal{E}_{\mathcal{P}_i} \leftarrow$ InscribedEllipsoid$(\mathcal{P}_i)$ $i \leftarrow i + 1$
while $(\boldsymbol{vol}(\mathcal{E}_i) - \boldsymbol{vol}(\mathcal{E}_{i-1}))\,/\,\boldsymbol{vol}(\mathcal{E}_{i-1}) \geq$ *tolerance*;
return $(\mathcal{P}_i, \mathcal{E}_{\mathcal{P}_i})$

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. P. Trutman, S. E. D. Mohab, D. Henrion, and T. Pajdla, "Globally optimal solution to inverse kinematics of 7dof serial manipulator," *arXiv preprint arXiv:2007.12550*, 2020.
3. P. A. Parrilo, "Sum of squares programs and polynomial inequalities," in *SIAG/OPT Views-and-News: A Forum for the SIAM Activity Group on Optimization*, vol. 15, no. 2.
4. B. Sturmfels, "On the newton polytope of the resultant," *Journal of Algebraic Combinatorics*, vol. 3, no. 2.

[7] https://github.com/RobotLocomotion/drake/blob/2f75971b66ca59dc2c1dee4acd78952474936a79/geometry/optimization/iris.cc#L440

5. S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
6. C. Ferrier, "Computation of the distance to semi-algebraic sets," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 5.
7. R. Tedrake, *Robotic Manipulation*, 2021. [Online]. Available: https://manipulation.mit.edu/pick.html#monogram
8. P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1.